Energy-Efficient Resource Allocation for Edge Computing based on Models of Power Consumption

a thesis presented for the award of
Doctor of Philosophy

Pengcheng Liu

Supervisor: Dr. Martin Collier

School of Electronic Engineering
Dublin City University

Switching and Systems Laboratory September 2018

I would like to dedicate this thesis to my loving parents

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Pengcheng Liu

Pengcheng Liu

ID No.: 13210324

Acknowledgements

I hereby sincerely acknowledge the invaluable advice and tremendous assistance of my supervisor, Dr. Martin Collier, during the course of this research and the time in Switch and System Laboratory, Dublin City University, along with his knowledge and acute observation towards the new technologies, which motivated and impacted me thoroughly. I am very glad and fortunate to have his guidance in my research and the dissemination of the outcomes.

My gratitude also goes to Dr. Xiaojun Wang for his continuous motivation and supports. With special thanks to Mr. Probal Bose, my best friend, for his company all the time. Grateful thanks are extended to Mr. Erfu Zhao, for his meticulous care during the time I was writing this thesis.

Lastly, I would like to thank my family, for their constant encouragement and love.

This work was supported by the "In-Network Programability for next-generation personal cloUd service supporT" project, EU Horizon 2020 INPUT, grant agreement number 644672.

Table of Contents

List of Figures

List of Tables

Nomenclature

Acronyms / Abbreviations

ACPI Advanced Configuration and Power Interface

APN Active Programmable Network

AR Adaptive Rate

ASIC Application-Specific Integrated Circuit

CAM Content-Addressable Memory

CMOS Complementary Metal-Oxide-Semiconductor

DAQ Data Acquisition

DCG Dynamic Clock Generation

DCS Dynamic Clock Selection

DLL Delay Locked Loop

DRAM Dynamic Random-Access Memory

DVFS Dynamic Voltage and Frequency Scaling

EAS Energy-Aware State

ENDP Edge Network Device Placement

GAL Green Abstraction Layer

GeSI Global e-Sustainability Initiative

GHG Greenhouse Gas

GMII Gigabit Media-Independent Interface

IC Integrated Circuit

ICT Information and Communication Technologies

IDC Internet Data Centre

G-ENDP Greedy Edge Network Device Placement

I-ENDP Iterative Edge Network Device Placement

IIoT Industrial Internet of Things

ILP Integer Linear Program

InP Infrastructure Provider

IoE Internet of Everything

LDMC Logic Delay Measurement Circuit

LPF Low-Pass Filter

LPI Low Power Idle

LP Linear Programming

MIP Mixed Integer Program

MVFS Multi-level Voltage and Frequency Scaling

ND Network Device

NDPM Network Device Power Model

NFV Network Function Virtualisation

OERA Online Energy-efficient Resource Allocation

ON Overlay Network

PFD Phase Frequency Detector

PHY Physical

PLL Phase Locked Loop

QoE Quality of Experience

SDN Software Defined Networking

SP Service Provider

SRAM Static Random-Access Memory

VCO Voltage-Controlled Oscillator

VNE Virtual Network Embedding

VNM Virtual Network Mapping

VNP Virtual Network Provider

VNR Virtual Network Request

VN Virtual Network

Energy-Efficient Resource Allocation for Edge Computing based on Models of Power Consumption

Pengcheng Liu

Abstract

Computing services, when provided by Edge Networks rather than centralized clouds, are delivered close to the geographically extreme user edge. Edge computing enables functional offloading and improved scalability but suboptimal design of edge networks can result in needlessly high energy consumption and mismanagement of resources. Thus, how to effectively minimize the power dissipation of network resources at the edge is a significant problem as networks evolve.

This thesis investigates a complete suite of energy efficient solution for the edge network. A frequency scalable router architecture, based on the Software Defined Network (SDN) concept, has been proposed. Two new control policies have been integrated with the proposed green architecture and their performance has been analysed to evaluate the trade-offs between energy efficiency and performance in frequency-scaled Network Devices. A Network Device Power Model (NDPM) has been formulated to explore the power dissipation characteristics of frequency scalable CMOS devices (as measured using a NetFPGA testbed). An Online Energy-efficient Resource Allocation model (OERA) has been designed based on this model. This allocation model can map the resource requests onto a substrate network in the edge, with concurrent consideration of multiple factors including geographical location, resource availability and network-level energy cost, etc. The model features better support of virtual resource requests and lower power consumption than existing solutions.

Chapter 1

Introduction

Edge computing extends cloud computing to the network edge to form a decentralized infrastructure with application intelligence and storage closer to the clients. Edge computing shares some characteristics with volunteer computing [1], fog computing [2] and mobile cloud [3], which also provide resources to the end-users. However, edge computing differs in the resources' geographical location and their deployment resulting in improved response time for end users [4]. The trend of offloading services and applications from user devices to the cloud has increased, reducing the resource burden on the local device but increasing it in the cloud. Delivering these services from the edge network should be more **energy efficient** than using a centralized cloud.

1.1 Motivation

On recent years innovative Internet services and applications, such as cloud computing, social networking and e-commerce, etc., are experiencing prodigious development. For cost and environmental concerns, such as the requirement to reduce global Greenhouse Gas (GHG) emissions, energy efficient networks have become a vital research topic and attracted considerable attention from both academia and industry. The network devices (NDs) among

the networks have an efficiency trade-off between energy consumption and performance, this is because,

1. performing computationally intensive tasks on mobile devices increases power consumption;

2. sending data to and receiving from the edge increases bandwidth utilisation as well as network energy consumption.

Recently published statistical data shows an alarming trend of rising energy consumption because of new developments in network technologies. The Global e-Sustainability Initiative (GeSI) [5] predicted that without any further countermeasures, ICT will generate a 2.3% increase in global carbon footprint by 2020. Alcatel-Lucent has reported that the transport and core networks alone account for 30% of the overall network power dissipation [6]. In [7], the authors' assessment is that the relative share of the worldwide total electricity consumption of communication networks, personal computers, and data centres products and services has increased from about 3.9% in 2007 to 4.6% in 2012, and estimate that the yearly growth of all three individual ICT categories (10%, 5%, and 4%, respectively) is higher than the growth of worldwide electricity consumption in the same time frame (3%).

Nevertheless, a substantial part of this power consumption can be saved; for example, 40% of the network power dissipation can be reduced if the energy is in proportion to its carried traffic load [8]. Therefore the edge networks must:

- apply and adapt more power efficient techniques to the NDs, especially at the network edge;

- allocate NDs which are capable of actively or passively adjusting their power utilisation according to multi-factorial impacts such as traffic load, rate of packet loss etc.;

- deploy services and applications with energy efficient resource allocation schemes;

- apply and adapt multi-factorial power, bandwidth and placement cost minimisation techniques for NDs without compromising the user demands.

This thesis achieves these goals by carefully formulating a Network Device Power Model (NDPM) to represent the power dissipation characteristics of CMOS devices (as measured using NetFPGA platforms) and considers a suite of multi-factorial algorithms for power-aware resource allocation, which show better support of virtual resource requests but lower power consumption compared with other existing solutions.

1.2 Objectives

The contributions of the research presented in this thesis are as follows:

- to design and implement a frequency scaled ND to reduce overall energy consumption. Two approaches to providing multiple clocks, namely Dynamic Clock Generation (DCG) and Dynamic Clock Selection (DCS), are proposed. The performance impacts of using frequency scaling in a CMOS ND are explored through numerical and simulation studies.
- to derive a concise Network Device Power Model (NDPM) of a CMOS ND. Following careful measurement of the power consumption of a real CMOS ND, an NDPM has been developed for use in all later work presented in this thesis. The NDPM, developed from empirical measurements, will be a more realistic and accurate basis for modelling the actual hardware than theoretical, software and utilisation models.
- to design two energy control policies, namely the Escalator and the Hysteresis policies, and implement their algorithmic models for the frequency scaled ND.
- to develop an Online Energy-efficient Resource Allocation (OERA) scheme, which supports online virtual network requests (VNRs) for resource allocation in a substrate network. OERA maps the virtual network requests (VNRs) onto a substrate edge

network with concurrent consideration of dynamic requests, computation and link bandwidth availability, and node distances.

- to formulate and solve this multi-level requirement problem using linear programming (LP) solvers for solving resource mapping problems. This thesis considers high numbers of network requests, traffic flows and resources to formulate a realistic performance study. The performance of OERA is benchmarked against two existing well-known algorithms (R-ViNE and D-ViNE [9]).

1.3 Summary

The reminder of this thesis is organized as follows.

Chapter 2 provides background information about network virtualisation. It first gives the motivation for using virtual networks in the future Internet. An overview of current research progress in this field is then presented. The prevalent approaches to virtual network resource allocation are described next, followed by the evaluation metrics used in the following chapters. The concept of Software Defined Networking and existing power management techniques for increasing network flexibility are also presented in this chapter.

Chapter 3 describes the power consumption in CMOS circuits, giving detailed descriptions of the frequency scaling technique used to build a power adaptive network device. The evaluation testbed used for investigating frequency scaling switching schemes is introduced in this chapter. The existing reference design of an SDN switch based on NetFPGA platforms is described.

Chapter 4 contains an explanation of the multiple clock generation architecture. Multiple clocks are needed to support frequency scaling. This is followed by a detailed description of a new design of power efficient SDN switches, and its performance on both the NetFPGA-1Gbps and 10Gbps platforms is measured. This chapter also contains an explanation of

the power consumption model used and its evaluation. The proposed power management mechanism is then described.

Chapter 5 presents two frequency control policies. It gives details of how the frequency scaled ND system can be modelled, followed by an analysis of the state-dependent transition rate. Details of the simulation results are then given and discussed.

Chapter 6 presents a novel algorithm to address energy-aware resource allocation in edge networks. The energy minimisation problem is established as a linear programming one and the presented solutions show better support of virtual resource requests and lower power consumption than existing solutions.

Chapter 7 summarises the key achievements of the thesis and their significance. It also presents directions for future research.

Chapter 2

Issues in Designing Flexible Networks

2.1 The Internet is Ossified

The Internet is one of the most widely used technologies of our known world. It plays the most essential and critical roles in social development, economic growth and technology innovation. The Internet helps people in acquiring and updating their knowledge in unprecedented ways and its scope has been spread beyond people's expectation and imagination at the time of its invention over 40 years ago.

However, with soaring dependencies on the Internet, various novel applications have occurred to meet these growing expectations. Distinctive applications have been introduced for multifarious requirements of security, mobility and flexibility etc. None of these challenges were conceived when the Internet was invented, but have motivated the next generation evolution of the Internet. Focusing on the infrastructure of the present Internet, the cardinal reasons for its current ossification status are as follows [10]:

1. *Heavy Dependency on Hardware*

   Today's Internet consists of miscellaneous network devices, which are dedicated to the execution of network functionalities (such as Layer 2 switching or Layer 3 routing). Various *"network boxes"* need to be embedded into the infrastructure to perform a

certain service (like Deep Packet Inspection etc.). If the network topology needs to be adjusted, these hardware systems have to be physically rearranged to satisfy the dynamic requirements.

2. *Multi-Provider Nature*

   Multiple network providers have their dedicated policies to establish and maintain their exclusive network infrastructure. The Internet is maintained by disparate autonomous systems. Autonomous system providers focus on the quality of service and robustness of their operational domain only. This divide-and-conquer approach limits the scope for optimisation of the global network and hinders the introduction of new protocols and techniques into the global infrastructure [11].

3. *Hourglass Internet Protocols*

   The particular hourglass shape of the current Internet stack occurs because of the evolution of upper and lower layers being much faster than the layers in the middle. The top and bottom layer accommodate multiple protocols leaving the waist of the hourglass feebler. With the growing demands of the Internet along with huge amounts of data involved, the only operational protocol in the middle layer is IP (Internet Protocol); this leads to an urgent need to accommodate this layer with, for example, support of cross-layer features.

To fend off this ossification of the modern day Internet, several new concepts have been proposed and some have been deeply investigated, such as Cloud Computing, Software Defined Networking (SDN), Virtual Network (VN), and Edge Computing etc. This thesis will focus on **network virtualisation**.

2.2 Cloud Computing

Cloud Computing has achieved enormous popularity due to the low start-up costs for users, pay-as-you-use billing, and flexibility of resource virtualisation and automation, etc. A critical ongoing challenge is interfacing billions of devices (i.e. Internet of Everything (IoE) and Industrial Internet of Things (IIoT) concepts) with cloud services. While the cloud can offer high processing capabilities and storage capacity [12], rapid increases in the number of supported network devices (NDs) and supported cloud applications may degrade real-time services delivered from centralized data centres. The factors adversely affecting a quality of experience (QoE) include high bandwidth utilisation and network latency. One measure to mitigate these effects is to use edge computing and edge networks.

The original design goal of the Internet was to connect the dispersed computers together and share the information along with the resources. However, with the rapid development of the Internet, especially the occurrence of later enterprise networks, the Internet Data Centre (IDC) concept arose to address the difficulties in the management of Internet devices, including network equipment and servers. The industry began a process of moving all Internet services to the data centre, and the resulting centralized control and management have reduced both capital and operating costs.

The growth of new Internet services and particularly mobile device based Internet technologies has greatly increased the number and scale of the data centre. However, in contrast to the fast development of computing capacity (Moore's law), the utilisation of computing resources in data centres was often poor [12]. Virtualisation technology addresses this inefficiency and then naturally took centre stage in today's storage, computing and management infrastructures. The concept of virtualisation is to provide an abstraction of virtualised multiple logical computing units in a single physical device and enables full utilisation of the available resources in the device. Security and reliability are maintained

because different computing units work separately without interfering with others' contexts. Amazon took a step further and developed the well-known Amazon Web Service (AWS) to rent out the surplus resources to retail users. This successful model greatly promoted the development of Cloud Computing.

2.3 Software Defined Networking

Software Defined Networking (SDN) is intended to enable flexibility and innovation in the network and thus to improve the efficiency of network operation and service quality as well as the reduction of CAPEX and OPEX. This is facilitated by the removal of the network control plane from the distributed network devices to a logically-centralized control entity, which enables the introduction of new open interfaces between the application, the data plane, and the control plane [13]. With these interfaces, the network control plane can be realised as a freely programmable software, essentially an operating system for the network.

When a packet reaches an SDN-enabled node, it is buffered and the packet header, or other sub protocol information, is matched against some pre-defined rules which trigger switching decisions, additional processing activity, or other actions. In the case of a successful match, the action(s) specified in the rule are executed. If there is no matching rule, the packet is either dropped or a "packet-in" message containing the packet header is sent to the controller for further processing. The controller determines the action the network element should take and communicates its decision to the network element. This may involve the insertion of a new rule for subsequent packets belonging to the same flow. The management of these rules, and the task of enacting them as appropriate, is standardised in the *OpenFlow* protocol [14].

As stated by its first designers [15], the OpenFlow model was conceived to abstract the main functionalities provided by heterogeneous hardware platforms for network nodes, and

Table 2.1 Related Research

Project	Description			
	Research Domain	Network Techniques	Virtualisation Level	Reference
X-Bone	dynamically deploy and manage Internet overlays.	IP	Application	[16]
PlanetLab	deploy and manage testbeds.	IP	Application	[17]
NetScript	program intermediate network node functions.	IP	Network	[18]
UCLP	dynamic allocation of optical network resources.	SONET	Physical	[19]
Tempest	adaptive management in ATM control architectures.	ATM	Data Link	[20]
Emulab	deploy and manage testbeds.	IP	Network	[21]
4WARD	deploy and manage VN.	IP	Network	[22]

for this reason, it is much closer to the internal hardware organization of nodes than legacy network protocols.

2.4 Virtual Network Overview

2.4.1 Related Research

There are many incarnations that are closely related to the concept of network virtualisation, such as Virtual Private Network (VPN), Overlay Network (ON), Active Programmable Network (APN), Network Function Virtualisation (NFV) etc. Table 2.1 lists several well-known network virtualisation projects.

2.4.2 Network Virtualisation Architecture

Any virtual network has a provider of service and a user of service. The service providers are generally categorized with three different roles, which are Infrastructure Provider, Virtual Network Provider and Service Provider. In general, the Infrastructure Provider (InP) organises

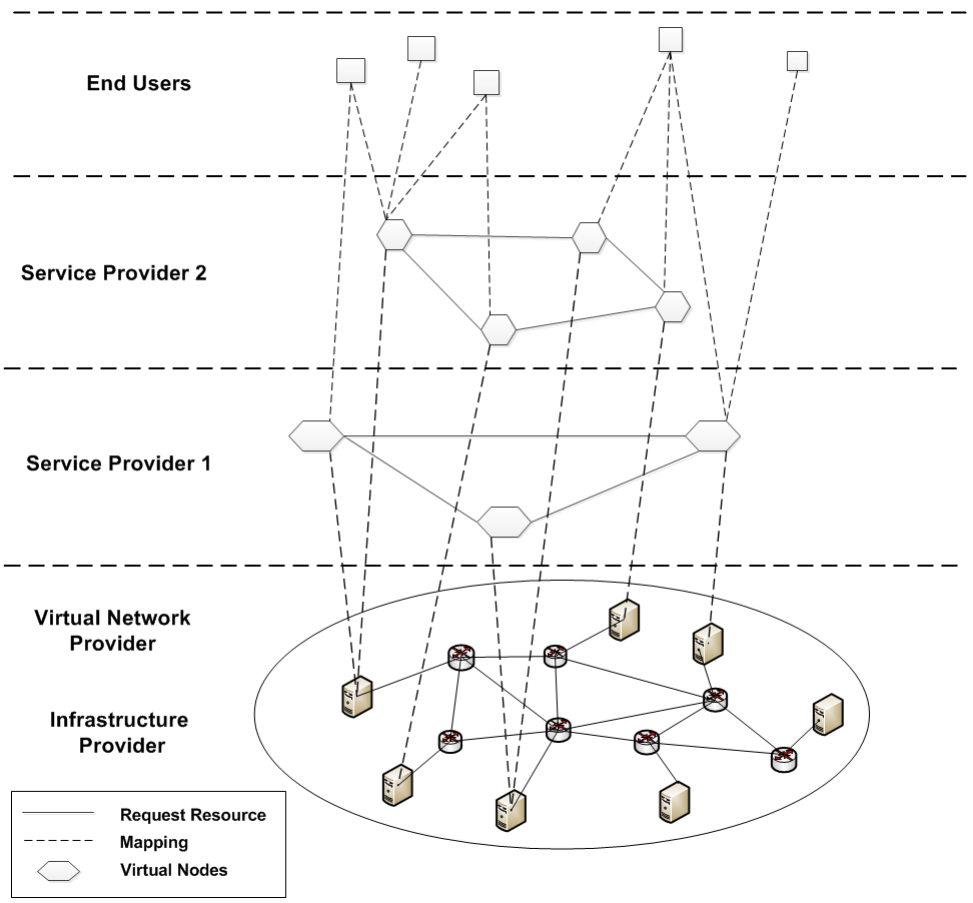


Fig. 2.1 A Typical Network Virtualisation Architecture

a certain number of physical resources needed to establish various virtual networks requested by the Service Provider (SP) and maintained by the Virtual Network Provider (VNP). The End User is always associated with the SP to receive the end service [23]. A typical network virtualisation architecture is shown in Fig. 2.1.

- **Infrastructure Provider (InP)**

  An Infrastructure Provider (InP) is responsible for the deployment, management and maintenance of the physical network resources. InP uses different virtualisation techniques to partition the resources into isolated slices and allocates these slices to the customers according to their requirements. The InP has knowledge of the allocated

resources but not of the inner protocols controlling the network. InPs can communicate with each other to provide inter-operative services.

- **Virtual Network Provider (VNP)**

  The Virtual Network Provider (VNP) is situated between the Service Provider and the Infrastructure Provider. It gathers virtual resources from one or more InPs according to the virtual network operators' request. VNP basically co-ordinates and sets up the infrastructure in which the operator can build the virtual network to satisfy network and end-user dynamic requirements.

- **Service Provider (SP)**

  The Service Provider (SP) is in charge of creating, operating and managing the virtual networks. SP provides end-to-end services to the end users by building and implementing VNs. It uses the resources leased from different InPs and then combines them to form single or multiple VNs. An SP can create VNs and act as a virtual InP to provide service to other SPs as well. Reacting to the different characteristics (e.g. delay, jitter, packet loss etc.) of the different network services and the requirements of particular applications (e.g. CPU, storage, bandwidth etc.), the SP generates the appropriate virtual network requests (VNR) which consist of resource requirements, topology, renting time (e.g. lifetime of the particular VN), and sends them to the VNP, which will establish the required infrastructure for the SP to build the required VNs. The virtual network could cross more than one VNP but it will present a single global view to the SP.

- **End User (EU)**

  End Users (EU) in a virtual network architecture are similar to those in the current Internet architecture. The EU is the terminal user who is allowed to access the services. From the viewpoint of the EU the ultimate advantage is the liberty to choose different

services provided by different SPs through multiple VNs, according to its interests, preferences, and service QoS requirements etc.

In summary, InP provides the physical layer, or more precisely the physical network infrastructure. The Service Providers (SP) receive the requirements from the User Equipment (UE) and generate resource requests. These resource requests are then sent to the Virtual Network Provider (VNP). VNP eventually does the mapping of the physical resource to the final user according to these resource requests from SP.

Chapter 4 and Chapter 5 have an explicit focus on InP and on the improvement of the power efficacy at the network device level, and Chapter 6 includes a discussion of VNP along with how to effectively allocate the network resources to the users in an optimized manner to minimize the entire network power consumption.

2.4.3 Advantages and Benefits

The advantages of network virtualisation include:

- **Flexibility:** The devices on the physical substrate layer are generalised as the resource pool. The SP can request resources according to their requirements and they will be allocated on demand, rather than being provisioned statically.
- **Decoupling:** In role of Internet service provider is decoupled into the interacting team of infrastructure provider, service provider and virtual network provider, decoupling service provision to end users from the maintenance of physical resources. The idea of decoupled network allows vendor independent infrastructure, which supports the deployment of various technologies and protocols within a single framework whether in the access or the core part of the network.
- **Isolation:** The virtual network concept allows multiple isometric virtual networks to run parallel to each other on top of the shared physical substrate infrastructure. The

owners of different virtual networks can deploy their own customized protocols and services by programming the shared resources.

- **Customization:** The fundamental design objective of virtualisation is to establish a customized networking stack. Network virtualisation masks the disparate nature of the substrate infrastructure and provides uniform programmable interfaces. The upper entities can independently deploy their own protocols running on their designated virtual networks without any interference from other virtual networks running in parallel.

The network virtualisation has the capacity to promote innovations for the network, and has immense potential challenges to be investigated. Hence, the network virtualisation has been widely recognised as one of the most critical trends for the future Internet technology.

2.4.4 Research Challenges

Several challenges related to network virtualisation still need to be explored and studied [23].

- *Interfacing*. As network virtualisation shares the common physical substrate infrastructure which is organised and maintained by various InPs, every InP must provide a standard interface to SPs for communicating with them to exchange their requirements. Similarly, the interfaces between EU and SPs, among multiple InPs, as well as among SPs must be identified and standardized.

- *Admission Control*. To provide guarantees of QoS and SLAs (service level agreements), InPs must be able to execute accurate accounting, implement admission control and perform distributed usage policing algorithms to ensure the physical resources allocated to the SPs are not overextended. Admission control must be developed for entire VNs rather than individual nodes or links.

- *Resource Allocation*. Resource Allocation, also called Virtual Network Mapping (VNM), or Virtual Network Embedding (VNE). Appropriate allocation and efficient scheduling of physical resources are extremely critical so as to maximise the number of coexisting VNs and thereby maximise revenue. At present, there are three different approaches to research on virtual network mapping, namely VN mapping in one InP, VN mapping across multiple InPs and VN mapping efficiency [1].

- *Isolation*. Isolation between coexisting VNs must be ensured to improve fault tolerance, security and privacy. The impact of misconfiguration or resource depletion in one particular VN must be contained within itself and not spread to other coexisting VNs.

- *Security*. Even though isolation between coexising VNs can provide a certain degree of security and privacy, it does not obviate the prevalent attacks and intrusions to the physical devices and VNs. Insecure programming models and interfaces can increase the vulnerability of VNs.

2.5 Virtual Network Resource Allocation

2.5.1 Resource Allocation Model

This subsection briefly presents the network virtualisation model and demonstrates an example afterwards. The concrete mathematical description of the resource allocation model will be defined in Chapter 6.

1. Substrate Network: The substrate network is modelled as a undirected connected graph G with specific weights, denoted by $G^S = (N^S, E^S, A_N^S, A_E^S)$, where N^S and E^S represent a set of substrate nodes and a set of substrate links respectively. A_N^S and A_E^S describe the

[1] Since the VNE problem is NP-hard, finding an optimal solution is computationally complex. For that reason, most existing researches focused on relaxing the original model and finding the sub-optimal solutions. However, even these heuristic approaches do not scale well for large networks [2].

attribute sets of the substrate nodes and links (for example; CPU capacity is a attribute of a node and bandwidth is of a link). $P^S(s,d)$ denotes the set of all loop-free paths in the substrate network from a source node s to a destination node d, $\forall s,d \in N^S$ and $s \neq d$.

2. Virtual Network Request: The VNR is an undirected connected graph denoted as $G^V = (N^V, E^V, C_N^V, C_E^V)$, where N^V is the set of requested nodes, and E^V is the set of requested links. C_N^V and C_E^V describe the constraints of the requested nodes and requested links. P^v denotes the set of all loop-free paths in the virtual network request graph.

3. Virtual Network Request Mapping: The mechanism of mapping virtual network requests to a subset of the substrate network, is modelled as $\mathscr{M} : G^V \mapsto (N', P', R_N, R_E)$, where $N' \in N^s$ and $P' \in P^s$. R_N and R_E represent the resources allocated to virtual nodes and virtual links.

An illustration of mapping two online requests VNR1 and VNR2 to a shared substrate network for resource allocation is presented in Fig. 2.2. Request VNR1 has the node mapping $\{a \mapsto E, b \mapsto C, c \mapsto D\}$ and VNR2 has the mapping $\{d \mapsto A, e \mapsto C, f \mapsto E, g \mapsto D\}$. Note that two virtual nodes from the same request cannot be mapped to the same substrate node (to enable fault tolerance), but two virtual nodes from two different VNRs can be mapped to the same substrate node (to maximise the resource utilisation). For example, in Fig. 2.2, $\{a, f\}$, $\{b, e\}$ and $\{c, g\}$ are mapped to the same substrate nodes as every pair of virtual nodes come from different VNRs.

2.5.2 Optimisation Objectives and Performance Evaluation Metrics

The objective of the virtual resource allocation is to maximise the revenue or minimise the cost. Several criteria have been defined in this section, which will be used as

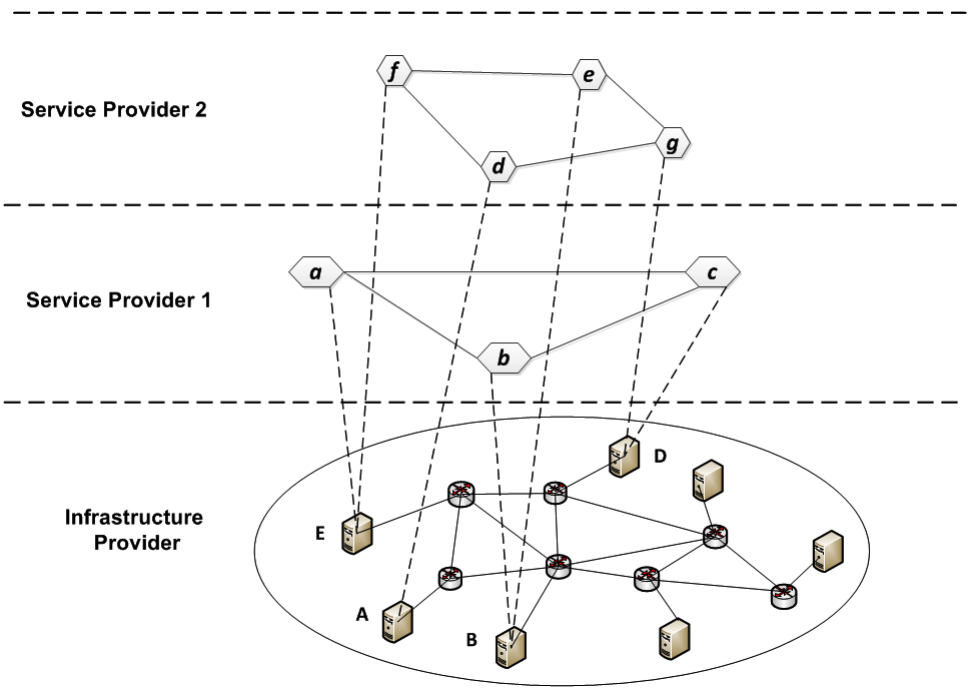


Fig. 2.2 VNR Mapping over a Substrate Network

the optimisation objectives or to evaluate the performance of different resource mapping algorithms.

1. **Cost**

   Cost represents the total amount of substrate resources used, comprising both CPU and bandwidth. There are different types of resources which can be considered and appropriately weighted, e.g. power consumption. Cost is defined as follows:

$$\mathbb{C}(G^V) = \alpha_c \sum_{e^V \in E^V} \sum_{e^S \in E^S} f_{e^S}^{e^V} + \beta_c \sum_{n^V \in N^V} c(n^V) \tag{2.1}$$

   where $f_{e^S}^{e^V}$ is the total amount of allocated bandwidth on the substrate link e^S. If a virtual link is mapped into a substrate path which consists of multiple physical links, the forwarding costs of the intermediate nodes are not considered. α_c and β_c denote the unit cost of the network resource (bandwidth) and the computing resource (CPU capacity).

2. **Revenue**

   The sum of virtual resources requested by the virtual entities is generally referred to as revenue. The computation scheme of revenue is similar to the scheme applied to determine the cost. Evaluation of the benefit generated using an VNM algorithm can be performed by calculating revenue. Similar to [24] and [25], the Revenue of a VN request is defined as:

$$\mathbb{R}(G^V) = \alpha_r \sum_{e^V \in E^V} b(e^V) + \beta_r \sum_{n^V \in N^V} c(n^V) \tag{2.2}$$

   where $b(e^V)$ is defined as the requested bandwidth of a virtual link e^V and $c(n^V)$ is the requested CPU capacity of a virtual node n^V. α_r and β_r are weights and defined by the service provider to reflect their respective cost of bandwidth and CPU. In [24] and [25], $\alpha_r = \beta_r = 1$.

3. **Acceptance Ratio**

   Acceptance Ratio is the ratio of the successful requests to total requests of the virtual network. It is defined as follows:

$$Acceptance\ Ration = \frac{Number\ of\ accepted\ requests}{Number\ of\ all\ requests} \tag{2.3}$$

4. **Stress Balance**

   Each substrate node can be mapped onto several virtual nodes. Similarly, a substrate link can also incorporate multiple virtual links. The stress level describes the number of virtual entities that are mapped onto a substrate entity. The stress ratio at time t, for node ($\mathbb{R}_{\mathbb{N}}$) and link ($\mathbb{R}_{\mathbb{L}}$), have been defined as the ratio of the maximum stress and the average stress [25].

$$\mathbb{R}^N(t) = \frac{\max\limits_{n \in N^S} S^N(t,n)}{[\sum\limits_{n \in N^S} S^N(t,n)]/|S^N|} \tag{2.4}$$

$$\mathbb{R}^L(t) = \frac{\max\limits_{e \in E^S} S^L(t,e)}{[\sum\limits_{e \in E^S} S^L(t,e)]/|E^S|} \tag{2.5}$$

where $S^N(t,n)$ and $S^L(t,e)$ express the stress level of the substrate node n and the substrate e at time t. $|S^N|$ and $|E^S|$ represent the number of the substrate nodes and the number of substrate links respectively.

Various existing mapping approach have been proposed to equally distribute resource load across the substrate network by minimizing the stress, as shown in Appendix A.

2.5.3 Algorithm Taxonomy

The virtual network resource allocation algorithms can be categorized, as follows.

- *Offline vs. Online*

  The VNE algorithms initially can be classified into two forms: offline algorithms and online algorithms. Offline algorithms assume that all VNRs are already known before they are processed. These algorithms do not consider the dynamic arrival VNRs, in other words, the offline algorithms ignore the distributed arrival time of different VNRs. Online algrithms assume that the VN requests are not known in advance. A VNR may arrive dynamically and stay in the network for an arbitrary period of time before departing. Online algorithms have higher computing complexity and are seen as *NP-hard* [26].

- *Static vs. Dynamic*

  This taxonomy depends on whether the mapped resources can be reconfigured in order to optimize the substrate resource utilisation and allocation. Every VNR is associated with an arrival time and an expiry time. The resources allocated to a VNR will be withdrawn after the specific expiry time has passed. Dynamic algorithms try to reconfigure the mapped resources over time to accommodate new VNRs. For

example, dynamic algorithms will re-compute the optimal solution to reallocate the resources once a VNR has expired and releases its resources. For the purpose of improving the global performance of embedding in the substrate network, the resources for successfully mapped VNRs may or may not undergo re-mapping. Static VNE algorithms do not attempt remapping of existing VNRs to optimize the performance of the embedding in the substrate network.

- *One Stage vs. Two Stages*

  Virtual network embedding algorithms deal with the dual allocation of virtual resources onto the nodes and links associated with the VNR. A one-stage algorithm implements the mapping of virtual nodes and virtual links to the substrate infrastructure simultaneously. In two-stages algorithms, the node mapping and link mapping are separated from each other. Generally, node mapping will be executed before link mapping in two-stages algorithms.

- *Centralized vs. Distributed*

  These approaches consider the situation that the substrate network is being run by multiple InPs. Centralized approach is based on using a single entity from the substrate network to perform the embedding. The distributed approach will utilize multiple substrate entities to compute the specific embedding. A significant advantage of the distributed approach is that the VN mapping problem is subdivided across InPs. This reduces the complexity of computing, at the cost of finding a suboptimal solution.

2.5.4 Existing Research on VNE

Several related works [9, 24, 25, 27, 28] address Virtual Network Embedding (VNE) onto substrate networks. Yu *et al.* [24] developed a hybrid approach to consider computing and link capacity abilities to measure the node rank. VNE mapping for nodes is achieved using a greedy algorithm for a number of small virtual networks. Chowdhury *et al.* ([9],

[27]) proposed an online algorithm to solve the VNE problem using a new node mapping stage (introducing Metanodes and Metaedges). Such node mapping using an augmented graph is also used in the new algorithms described in this thesis. Cheng *et al.* [28], inspired by Google's page ranking algorithm use a topology aware node ranking algorithm for virtual network mapping. Zhu and Ammar [25] use a stress algorithm to develop and maintain a balanced node and link stress for network resource allocation. A heuristic approach is used for clustering of lightly stressed substrate nodes and links. Table 2.2 lists several published algorithms for the allocation of virtual network resources.

Table 2.2 Existing Resource Allocation Algorithms

Reference	Allocation [Static(S)/ Dynamic(D)]	Computing [Centralized(C)/ Distributed(D)]	Coordination [One Stage(1)/ Two Stage(2)]	Contribution
[24]	S	C	2	Introduces the multi-path for mapping virtual links.
[29]	S	C	2	Introduces random BW demand to the VNE.
[30]	S	C	2	Proposes shared on-demand and pre-allocation backup schemes.
[31]	S	C	2	Propose opportunistic bandwidth sharing algorithm
[27]	S	C	2	Multi-path for mapping virtual links
[28]	S	C	2	Applies Markov Random Walk model to rank a network node.
[32]	S	C	1	Formulates and solves VNE as a mixed integer program.
[33]	S	C	1	Proposes a mixed integer program which provides optimal energy efficient embeddings.
[34]	S	C	1	Proposes an algorithm based on subgraph isomorphism detection.
[35]	S	C	2	Reoptimises and re-embeds initially-rejected VN requests after fixing bottleneck requirements.
[36]	S	C	1	Jointly considers node mapping and link mapping.
[37]	S	C	2	Propose a Proposes a new algorithm bases on the proximity principle.
[25]	S	C	2	Provides a balanced node and link stress in substrate network.
[38]	S	C	2	Allows a virtual node to be mapped into multiple substrate nodes.
[39]	S	D	2	Finds the a reconfiguration policy which can deduce the overall cost.
[40]	D	D	2	Propose a Dal self-organizing model to effectively manage the substrate network resources.
[41]	S	C	1	Proposes a VNE algorithm bases on traffic-matrix and a mixed integer programming formulation.
[42]	S	C	1	Proposes a heuristic solution to survivable virtual infrastructure mapping for failure-dependent protection.
[43]	S	C	2	Proposes a heuristic-based algorithm of mapping virtual link and a coordinated improvement approach in [27].
[44]	S	C	2	Uses physical resource migration to better utilize resources on substrate nodes.
[45]	S	C	2	Focuses on reconfiguration with critical virtual nodes by linear programming.

In future virtual network, using multiple resources within the same physical resources can be an enabler for energy optimization. There are multiple strategies on which researchers are working on; Chang and Wu [46] propose a heuristic approach that finds the minimization of the computing and communications power of applications in cloud computing environments. A Green Power Management (GPM) method has been proposed by Yang *et al.* [47] to find the optimized load balancing solution amongst virtual machines in a cloud substrate. Chapter 6 will address a novel online energy-efficient resource allocation approach with the objective of minimizing the edge network energy consumption.

2.6 Power Management in Networks

Depending on the hardware technologies of components, different power management techniques can be applied at the silicon level (e.g., clock scaling [48], voltage scaling [49], voltage gating [50], etc.), in order to reduce the resulting energy consumption in an effective way. Power management techniques applied to network nodes include:

1. *Adaptive Rate (AR)*: the operating speed of the hardware component is reduced in response to load.

2. *Low Power Idle (LPI)*: the hardware block or some of its components are turned off during idle periods. When the block receives new work, a wake up time must elapse before returning to a fully operate state.

3. *Standby*: this resembles LPI, but hardware blocks in standby mode require much longer times to wake up. During Standby modes, the functionalities provided by the blocks should be considered unavailable.

In systems compliant with the Advanced Configuration and Power Interface (ACPI) standard [51], the number of such configurations is limited to a few stable power management configurations for each hardware block.

2.6.1 Power Control Policies

The prevailing technologies to reduce network energy consumption can be placed in two broad categories. The first approach is to put network elements into a sleep mode during idle computational periods. [52] proposed green OSPF protocol which can detect links with low traffic (which can comprise up to 60% of total links in an actual IP network) and power them off to save energy. The second approach is so-called Rate Adaption which is an effective way of achieving network energy proportionality by adaptively scaling the link data rate in response to the link utilisation. [53] first described the ALR (Adaptive Link Rate) and intended it primarily for use in edge links. Nedevschi *et al.* [54] studied both forms of power management schemes and showed a substantial saving of network energy when using even simple schemes for sleeping or rate-adaption. Policies and rules to determine when to enter sleep mode or to change the link rate have been widely studied. Gunaratne *et al.* [55] investigated queue length based policies to determine the appropriate conditions to trigger link rate changes according to a dual-threshold policy. Researchers in [56] proposed two state-setting policies which considered including queuing length, service rate, rate adaption interval etc. and showed a deterministic bound on network delay. The work in [57] introduced network stability into the scheduling algorithms and studied two scenarios, namely temporary sessions and permanent sessions and showed how delay bounds are affected by rate adaption.

Gunaratne *et al.* [55] considered only two hypothetical service rates (High and Low) which may not be effective enough in practice considering the power saving. Although [58] extended the dual-threshold policy to consider various service rates, their study did not model rate transitions which will be of concern in practical systems. [59] proposed several local control algorithms and demonstrated their experimental results on NetFPGA platforms, but the authors did not provide its queuing policies or any performance model. A model addressing these deficiencies will be developed in Chapter 5.

2.6.2 Green Abstraction Layer

The Green Abstraction Layer or GAL is a novel interface/middleware architecture that enables network nodes to exchange power consumption data and locally apply control strategies for energy saving [60]. The European Telecommunications Standard Institute (ETSI) approved GAL as an official standard (ETSI Standard 204-237) in March 2014 [61]. GAL was conceived with the purpose of exchanging information about capabilities and parameter settings between energy-aware networking devices and their network management primitives. At the same time, it defines a suitable hierarchical structure, in order to propagate a similar abstract representation throughout the subsystems of devices (chassis, electronic boards, etc.), at an appropriate level of detail and granularity. GAL also defines a standard interface for exchanging energy consumption information among the network devices. GAL consists of two segments: the Energy Aware States (EASes) and the Green Standard Interface (GSI). EASes describe different power levels which are supported by the client devices; GSI is a set of primitives which are used for controlling the power consumption of the equipment by making transitions between different EASs. GAL had been successfully applied in a number of middle and large-scale projects such as DROP (the Distributed Router Open Platform) [62] [63] and is supported in commercial hardware such as Mellanox routers [64]. In this thesis, GAL has been integrated with the power-aware network devices as a power management standard in conjunction with the OpenFlow protocol which handles the forwarding control information.

2.7 Summary

The constant evolution and expansion of the Internet and Internet-related technologies has exposed the limitations of the current networking infrastructures, which exhibit unsustainable power consumption and low level of scalability. In fact, these infrastructures are

still based on the typical, ossified architecture of the TCP/IP paradigm. In order to cope with the requirements of the Future Internet, recent contributions envisage an evolution towards more programmable and efficient networks, such as cloud computing and SDN, which have been described in the chapter. Network virtualisation is one new approach promising the flexible and autonomic network management. This chapter gives detailed descriptions of current research on network virtualisation. How to effectively map virtual resources to physical resources is of great significant with this approach. The metrics used to evaluating the network resource allocation algorithms and the notation used to represent them have also been presented.

Chapter 3

Hardware Considerations in Energy-efficient Network Design

High performance switches and routers enabled the rapid growth of the next generation networks. However, the power consumption and the resulting thermal effects become one of the biggest challenges in modern network device designs [65]. The hardware designers can no longer focus on creating the highest performance chips because the highest performance circuit they can create is likely to dissipate too much power. Instead, researchers must now pay more attention to power efficiency in order to achieve high performance while satisfying power constraints considering the impact of critical factors raising the power consumption problems.

In this respect, this chapter describes the basic issues, the technical approaches, and the methodologies for the implementation of energy efficient network devices.

3.1 Power Consumption Analysis of CMOS Circuits

A wide range of computational approaches of modelling power consumption have been introduced for CMOS based integrated circuits (ICs). These approaches are described by

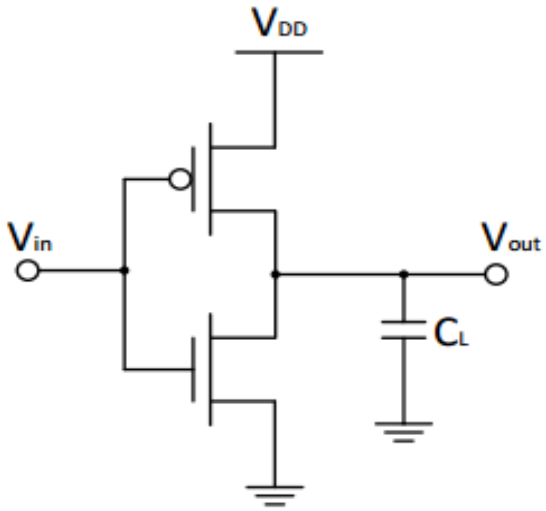


Fig. 3.1 CMOS Inverter

different mathematical equations and considered various levels of granularity. For example, the dissipation due to direct-path currents is considered as part of dynamic power consumption in [66], but it is not taken into account in [67]. The calculation methods described in [67] and [68] are comprehensively adopted in this thesis (with minor changes to the mathematical notation) because the mathematical expressions are concise and a level of granularity is considered.

The CMOS inverter is the basic component of all digital designs. The power consumption properties and characteristics of an IC can be quite accurately derived by extrapolating the investigation results obtained from the CMOS inverters. Fig. 3.1 depicts a static CMOS inverter which has two different states. The total power consumed by a CMOS inverter can be divided into three components, which are its **dynamic consumption**, **short-circuit consumption** and **static consumption**, as defined by Eq. (3.1).

$$P_{tot} = P_{dyn} + P_{dp} + P_{stat} \tag{3.1}$$

1. Dynamic Power Consumption

   The dynamic power consumption is caused by capacitor C_L which is charged/discharged through the PMOS/NMOS respectively. A measurement for one direction of dissipation can be precisely derived, as follows (taking the charging procedure as an example),

$$E_{vdd} = \int_0^\infty i_{VDD}(t) \cdot V_{DD} dt = V_{DD} \int_0^\infty C_L \frac{dv_{out}}{dt} dt = C_L V_{DD} \int_0^{V_{DD}} dv_{out} = C_L V_{DD}^2 \quad (3.2)$$

$$E_C = \int_0^\infty i_{VDD}(t) \cdot v_{out} dt = \int_0^\infty C_L \frac{dv_{out}}{dt} v_{out} dt = C_L \int_0^{V_{DD}} v_{out} dv_{out} = \frac{C_L V_{DD}^2}{2} \quad (3.3)$$

   where E_{VDD} denotes the energy taken from the supply during the transition, and E_C is the energy stored on the capacitor at the end of the transition [67].

   Eq. (3.2)-(3.3) show that during the low-to-high transition, the capacitor C_L stores energy equal to $\frac{C_L V_{DD}^2}{2}$, and another half of this energy has been consumed by the PMOS transistor. During the discharging procedure, the energy of C_L is dissipated in the NMOS transistor. In summary, each switching cycle (consisting of an $L \rightarrow H$ and an $H \rightarrow L$ transition) takes a fixed amount of energy, equal to $C_L V_{DD}^2$. In order to obtain the total power consumption, the switching frequency has to be taken into account. If the gate is switched $f_{0\rightarrow 1}$ times per second, the power consumption is given by

$$P_{dyn} = C_L V_{DD}^2 f_{0\rightarrow 1} \quad (3.4)$$

2. Short-circuit Power Consumption

   The short-circuit power consumption is caused by a direct current path existing between V_{DD} and GND for a short period of time during switching, while the NMOS and the PMOS devices are conducting simultaneously. Similar to the capacitive power dissipation, the short-circuit power dissipation is proportional to the switching activity.

The short-circuit power consumption can be calculated as follows

$$P_{dp} = t_{sc} V_{DD} I_{peak} f \tag{3.5}$$

where t_{sc} represents the time both devices are conducting and I_{peak} is determined by the sizes of the transistors[68].

3. Static Consumption

   Ideally, there should be zero static current through the CMOS inverter in steady state. This is because the PMOS and NMOS are never simultaneously on. However, a leakage current always exists and flows through the reverse-biased diode junctions of the transistors. The static consumption is composed of the $P-N$ Junction Reverse-Bias Current, the Subthreshold Leakage Current, and the Tunneling Current into and through the Gate Oxide [69]. In [66], the static dissipation is simply given by

$$P_{stat} = I_{leak} V_{DD} \tag{3.6}$$

   and this formula will be used in this thesis.

4. Total Power Consumption

   The total power consumption of a CMOS inverter can be expressed as the sum of the three components above

$$P_{tot} = P_{dyn} + P_{dp} + P_{stat} = (C_L V_{DD}^2 + V_{DD} I_{peak} t_{sc}) f_{0\to 1} + V_{DD} I_{leak} \tag{3.7}$$

Based on these formulas, and with parameter values typical of CMOS circuits, it may be concluded that:

- the capacitive dissipation is the dominant factor in a typical CMOS circuit;

- the supply voltage cannot be reduced without restriction; when V_{DD} approaches twice of voltage threshold that will lead to a serious performance penalty [66];
- the short-circuit power consumption can be kept within bounds by careful circuit design, and this leakage can be ignored [66];
- voltage and frequency scaling is an effective approach to reducing the power consumption;
- either a lower voltage or lower temperature can significantly reduce the lower leakage current [70].

3.2 Hardware Level Power Saving Techniques

The energy efficiency of network devices can be improved using **sleep mode** and **power scaling** [71]. Sleep mode refers to putting electronic devices into the lowest power status. This mode significantly reduces the power consumption compared to leaving the device fully operational continuously and constantly. In sleep mode, the devices stay in a minimum power state by turning off unnecessary subsystems and the RAMs [72].

Power scaling is another energy saving technique. It refers to dynamically scaling the power dissipation of the network devices by placing them into different processing states in response to various external factors such as traffic load.

3.2.1 Sleep Mode

Sleep mode seeks to disable unused hardware, or parts of its components, such as ports of a ND, putting them into a mode where they consume the lowest power, and waking them up as needed. Significant power can be saved when the hardware stays in sleep mode.

Sleep mode was first introduced in laptop computers for the purpose of extending battery life for mobile users [73]. After a certain period of time with no application activity detected by the operating system, a specific firmware, located between the operating system and the hardware, will initiate power management and trigger a signal to the hardware (e.g., the processor, hard drive, network card, etc.), so that the corresponding hardware components can be placed into a low power sleeping mode. A wake-up interrupt will then lead such sleeping hardware components to return to an active operating mode once the operating system detects demand from applications. Similarly, a customized set of power management primitives in green networking allows NDs to turn their functionalities almost completely off and stay in a sleep mode.

There are challenges to introducing sleep mode in network devices as time and power are needed to transition between *sleep* and *active* mode. How to predict the appropriate state in advance is still a topic of ongoing research.

3.2.2 Power Scaling

Power scaling can also be used to improve the power efficiency of the network hardware. Various techniques of power scaling exist such as the following:

1. *Dynamic Voltage and Frequency Scaling (DVFS)*

   Eq. (3.4) in Section 3.1 shows the quadratic relationship between the dynamic power consumption and the supply voltage. Therefore, decreasing the voltage can significantly reduce the dynamic power. However, reducing the supply voltage comes at the expense of increasing circuit delay and results in decreasing the circuit performance. *Dynamic Voltage Scaling* (DVS) refers to adjusting the voltage supply dynamically instead of using a constant value to maintain circuit performance while reducing the power consumption to its maximum possible extent. In [49], a novel *logic delay measurement circuit* (LDMC) is used to adjust the supply voltage of an FPGA chip in a closed loop

fashion based on the operating temperature. [49] reports a 4% to 54% power saving on a Xilinx Virtex XCV300E FPGA by introducing LDMC. *Dynamic frequency scaling* (DFS) [48] [74] is another power scaling technique that refers to dynamically scaling the clock speed of the processor depending on the switching load.

2. *Multi-level Voltage and Frequency Scaling (MVFS)*

   Use of $dual - V_{dd}$ has been explored and already successfully applied in ASICs within various fields (e.g. [75]). Li *et al.* has published a series of papers regarding this technique [50, 76–78]. The concept of V_{dd} programmability for FPGA was first introduced in [50], and [76] and [77] present analysis and experimental results based on this concept. [78] shows that introducing the field-programmable $dual - V_{dd}$ on a 100-nm FPGA platform can result in an overall power saving of more than 45%. [58] proposed a multi-frequency scaling scheme where a clock adapter is designed to dynamically scale the internal frequency by examining the buffer usage inside a network device.

3. *Glitch Removal*

   A glitch is an unexpected transition that dissipates undesired dynamic power in digital circuits and may cause logic errors. The glitch is often a result of a fault or design mistake that occurs where the input signals to logical cells have different arrival times. [79] investigates the effect of glitches on power consumption and reports that between 20% and 70% of total power dissipation in ASICs is due to glitches. [80] proposes to reduce glitching power in FPGA circuits by using negative edge triggered flip-flops. Lamoureux *et al.* [81] introduce a programmable delay element to the logic blocks which aligns the arrival times of the input signals to prevent glitches and reports removal of 87% of glitches and a reduction of up to 17% of overall power dissipation.

4. *Clock Gating and Power Gating*

   Safeen *et al.* [82] described a novel clock gating architecture, in which a network switch is integrated inside the built-in clock tree. A customizable enabling pin is used to control the switching so that partial logical blocks which are not used for a function will be disabled to decrease the power consumption of the clock network. The clock gating approach is used in a Xilinx Virtex-5 FPGA by Wang *et al.* [83] who reported a 28% reduction in power consumption due to the reduced toggling on the clock interconnect. Multiple mode power gating was proposed in [84]; it can achieve a further reduction of 17% in power dissipation. The power gating technique is also an effective approach to decreasing the static power consumption.

5. *Partial Reconfiguration*

   Partial reconfiguration allows a portion of a circuit to be reconfigured while the remain parts continue to work without any interference. The primary benefits of using this technique lie in hardware reuse and flexibility [85]. It potentially allows the implementation of a large-scale system using relatively small-scale hardware logical resources. Partial reconfigration also has the advantage of reducing the power consumption, as has been verified and investigated by Liu *et al.* [86]. The inactive hardware modules inside an FPGA can be unloaded using partial reconfiguration while the chip is operating, saving power consumed in the leakage and clock distribution power.

3.3 FPGA as a Hardware Accelerator

The terminology of hardware acceleration commonly refers to the usage of specific hardware platform, such as FPGA, GPU, DSP, etc., to perform some functions more efficiently than is possible in software running on a more general-purpose CPU. A comparison

between four different hardware accelerator platforms, say multi-core CPUs, GPUs, MPPAs (Massively Parallel Processor Arrays) and FPGAs has been performed in [87]. The target application is Random Number Generation and the results reveals that for getting the best performance from each platform different implementation approaches to random number generation should be devised. In other words, methods which are highly efficient in scalar oriented CPUs may not work well neither in case of GPUs, nor in memory limited MPPAs.

Within the emergence of virtualisation approaches, the FPGA platforms as potential virtual computation resources found a new role. In [88] a systematic solution, termed as pvFPGA, for virtualising an FPGA-based hardware accelerator based on Xilinx Virtex-6 FPGA and the x86 platform has been proposed. The authors have shown that the time overhead caused by the virtualisation in the proposed solution is very close to zero. To cope with the intensive processing data mining tasks in genomic applications, an FPGA based solution has been addressed in [89]. Due to the intrinsic computational intensiveness of Video streaming applications, they are offered as suitable candidates for hardware acceleration engines. A real time implementation of image analysis algorithms based on Xilinx Virtex 5 FPGA family was proposed in [90], where a test prototype for four video HD_SDI streams has been also developed.

In [91] a fair comparison between the performance of GPUs and FPGAs, both as accelerators, has been done. The paper evaluates the technology for both GPU and FPGA devices, and performs a qualitative and quantitative comparison. The brief result is that FPGAs are more power efficient while GPUs are more cost efficient. Because of their flexible architecture and deep-down pipelining and parallelism capability, FPGAs have attracted both research and industrial community's interest to be customized as hardware accelerators [92].

3.4 Evaluation Testbed

In order to perform experiments on energy efficiency in network devices, a suitable testbed is required. The ideal testbed will allow hardware to be modified, and will support recent advances in network software that increase network flexibility, such as SDN and network virtualisation. The NetFPGA [93] architecture provides a suitable platform on which to design such a testbed, and is described below.

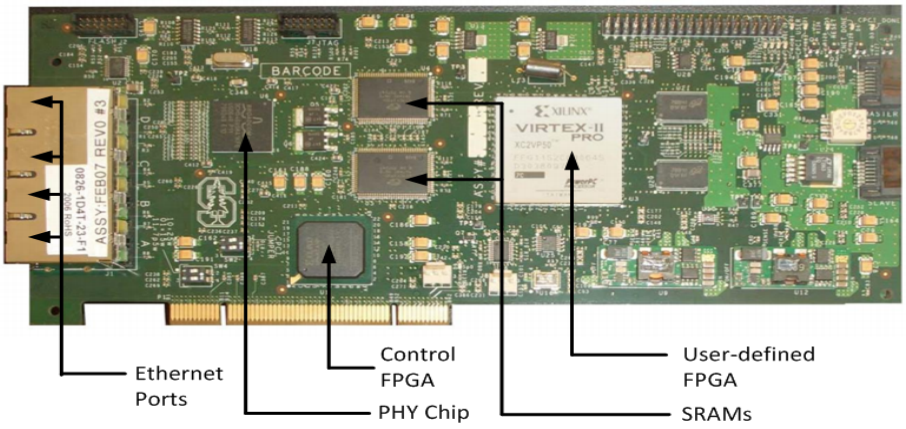
The NetFPGA platform is a line-rate programmable and open platform for network scientists and researchers. It is widely used both by academic institutions and industries around the world. It provides a fast way to design and investigate power efficient mechanisms with reconfiguration features.

3.4.1 The NetFPGA Platforms

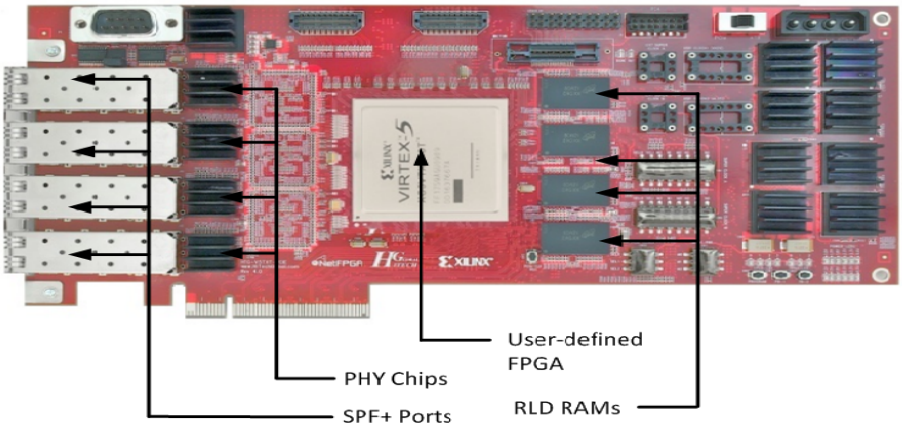
The NetFPGA-1G [94], shown in Fig. 3.2a, is a low-cost reconfigurable networking hardware platform with four Gigabit Ethernet ports attached. It contains a Xilinx Virtex-2 Pro FPGA that is used for user-defined logic (UFPGA) and another Xilinx Spartan-2 FPGA used as the PCI controller (CFPGA). The 1G platform has 4.5 Mbytes of SRAM and 64 Mbytes of DRAM. It communicates with the host PC through a standard peripheral communication interconnect (PCI) bus.

The Ten Gigabit version of the NetFPGA (NetFPGA-10G) [95], shown in Fig. 3.2b, was released in 2012. It is equipped with four 10G SFP+ interfaces and the FPGA has a Xilinx Virtex-5 Tx240T in its core. The NetFPGA-10G contains up to 27 Mbytes of SRAM and 288 Mbytes of DRAM as well as an eight channels of PCI Express interface supporting 5 GBPS/lane.

Table 3.1 compares the hardware configuration of the two boards.



(a) NetFPGA-1G board



(b) NetFPGA-10G board

Fig. 3.2 NetFPGA Platforms

3.4.2 Reference Design of SDN Switch

The design of the NetFPGA reference SDN Switch [94] [95] is modular. New features, such as the frequency scaling mechanisms discussed here, can be implemented by adding new custom modules or by adjusting the existing ones. Fig. 3.3 shows the main blocks of an SDN switch on a NetFPGA platform.

The entire reference system can be separated into two parts: the software portion and the hardware portion. The software portion resides on an X86 based host, exchanging OpenFlow Protocol messages with an external controller, collecting various statistical data,

Table 3.1 NetFPGA Board Comparison

	NetFPGA-1G	NetFPGA-10G
Port	4 x 1Gbps Ethernet Ports	4 x 10Gbps SFP+
Memory	4.5 MB ZBT SRAM 64 MB DDR2 SDRAM	27 MB QDRII-SRAM 288 MB RLDRAM-II
Interface	PCI	PCI Express X8
FPGA	Virtex II-Pro 50	Virtex 5 TX240T
Logic Cell	53,136	239,616
Block RAMs	4176kbits	11664kbits

as well as communicating with the hardware to set up flow entries etc. The main function of the hardware portion is to switch packets between different physical ports according to internal flow tables fabricated in the hardware. The hardware portion and the software portion communicate with each other via a DMA/CPCI interface. The software portion generates a software flow table after communicating with the controller, and this software flow table will be copied into a hardware flow table which is stored in CAM (Content-Addressable Memory) inside the hardware. The user traffic comes from one of the physical ports and its header processing looks up the CAM to determine the output port. If there is no record on CAM, these packets will be forwarded to the controller through the software portion. The controller will decide a new flow rule for these packets, and this specific rule will be added into the hardware for future reference. Hence recently used rules are cached in hardware.

The hardware implementation of the reference SDN switch on NetFPGA-10G platform is demonstrated in Fig. 3.4. Packets from each external interface and packets from the host interface go through a dedicated pipeline. Other substantial components of the switch are the SDN Data Path, the Input Arbiter and the Output Queues.

1. OpenFlow Data Path

   This is the core component of the OpenFlow switch. All activities in the data plane associated with OpenFlow, including flow table lookup, action processing and packet header handling have been implemented inside this module. After passing through this module, packets will be switched to the appropriate port(s) by the Input Arbiter

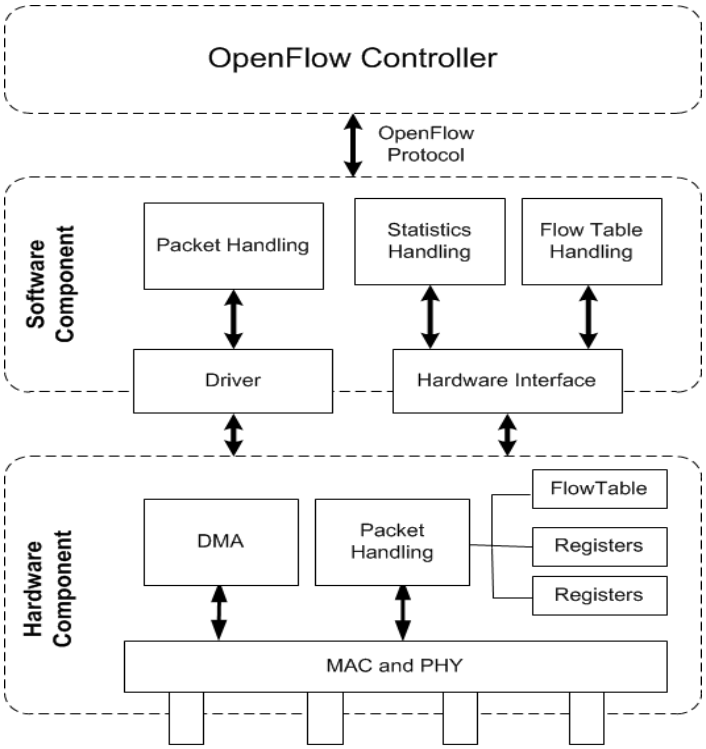


Fig. 3.3 Block Diagram of NetFPGA SDN Switch

component in conjunction with the Output Queues component, unless the packet is to be dropped.

2. Input Arbiter

   This module is in charge of arbitrating five streams of data (one from each port and one from the CPU) and producing a single multiplexed internal bus to feed the Output Queues. The arbitration uses a round robin algorithm to schedule packets.

3. Output Queues

   This module inserts each packet into the outgoing queue associated with its destination port. After some format conversion, packets are transferred to output interfaces.

The SDN switch design on the NetFPGA reference 1G platform is somewhat different and is shown in Fig. 3.5. Unlike NetFPGA-10G, NetFPGA-1G implementation has eight

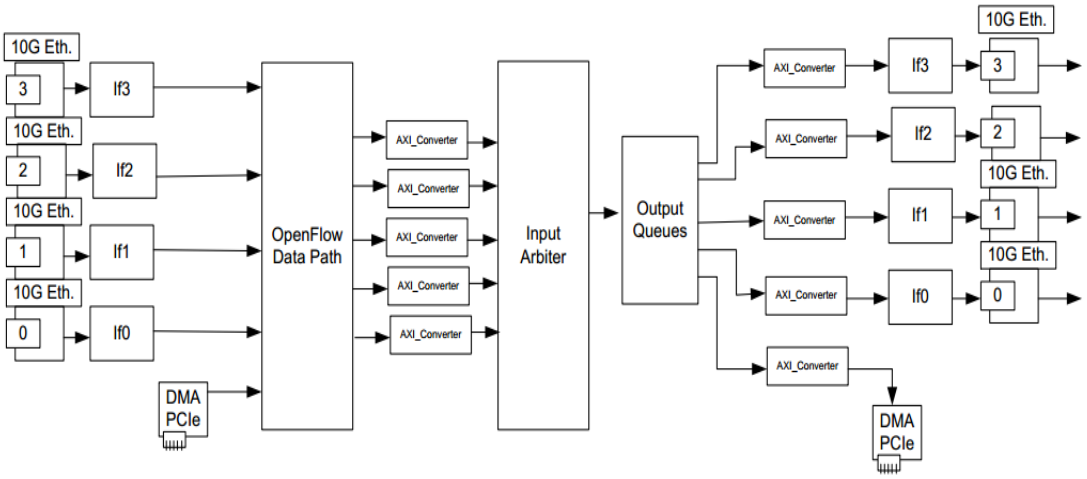


Fig. 3.4 NetFPGA Reference 10G SDN Switch, Internal Hardware Data Path

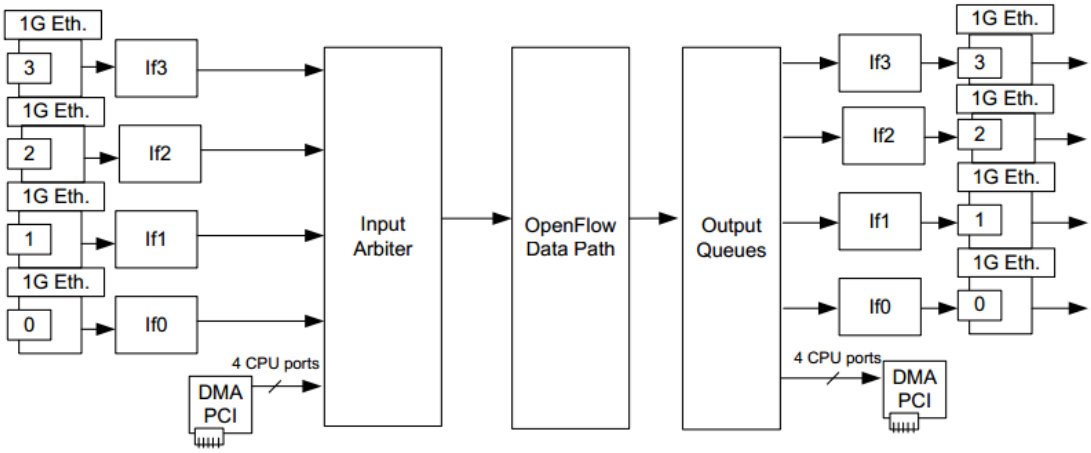


Fig. 3.5 NetFPGA Reference 1G SDN Switch, Internal Hardware Data Path

ports (four MAC ports and four corresponding host ports) and only one data path pipeline for data from all the eight ports.

Two NetFPGA platforms are used because they have different line rates. This gives more versatility to the experiments showing that the proposed NDPM is generic for any CMOS based network devices. The power dissipation results for both platforms show an excellent match to the model, as will be discussed in next chapter.

3.5 Summary

This chapter describes the power consumption in CMOS circuits, giving detailed descriptions of the frequency scaling technique used to build a power adaptive network device. The evaluation testbed used for investigating frequency scaling switching schemes is introduced in this chapter. The existing reference design of an SDN switch based on NetFPGA platforms is described.

Chapter 4

Energy Efficient Network Device Design and Evaluation

Eq. (3.7) shows that the power dissipation on a CMOS device is proportional to the switching frequency f_{clk}[1][2] and has a quadratic relation with the supply voltage (V): $P_{total} \propto f_{clk}V^2$. Appropriately modulating the system clock frequency is thus a way to reduce the power consumption. In this chapter, a frequency scaling *Software Defined Networking* (SDN) switch is designed and validated on both the NetFPGA-1G [94] and NetFPGA-10G [95] platforms. This new frequency scaling SDN switch can be used to study and investigate multiple factors impeding energy saving.

[1] It assumes that the device is designed on a synchronized circuit in which $f_{0\to1}$ can be expressed with respect to the clock f_{clk}.

[2] Switching activity, also called "toggle rate", is defined by Xilinx as "Toggle Rate is the rate at which a net or logic element switches compared to its input(s). For synchronous elements, the Toggle Rate reflects how often an output changes relative to a given clock input and can be modeled as percentage between 0-200%" [96].

4.1 Multiple Frequency Clocks to Support Frequency Scaling

To use frequency scaling, two novel approaches implementing multiple clock signals are dynamic clock generation (DCG) and dynamic clock selection (DCS), and these are described in this section.

4.1.1 Dynamic Clock Generation

Dynamic Clock Generation (DCG) uses frequency synthesis techniques [67] adopted from radio systems to generate a system clock f_{OPS} from a reference clock f_{REF} such that

$$f_{OPS} = M \cdot f_{REF} + \phi_0 \tag{4.1}$$

where ϕ_0 is a constant and M is a programmable integer. In a packet processing engine, the value of M may be set in response to buffer occupancies, so that the processing is accelerated when buffers start to fill. Frequency synthesis is achieved using a digital Phase Locked Loop (PLL) or Delay Locked Loop (DLL) [67], as shown in Fig. 4.1, which also illustrates a threshold mechanism used to select the value of M. When the value of M changes, the PLL loses lock, and a period of time which depends on the loop dynamics [97] elapses before lock is re-established.

The value of ϕ_0 maybe changed after lock is re-established, and so DCG is not synchronous. Numerous clock glitches will occur whilst lock is being established.

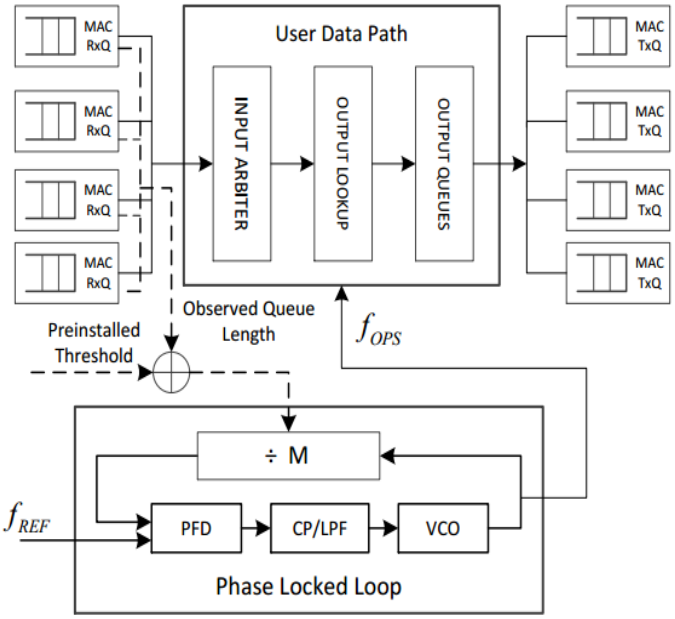


Fig. 4.1 Dynamic Clock Generation (DCG)

4.1.2 Dynamic Clock Selection

Dynamic Clock Selection (DCS) approach is depicted in Fig. 4.2. DCS is able to choose the best clock frequency from a set of pre-generated clocks, using techniques such as multiple crystal oscillators or frequency synthesis, to optimize the device power consumption.

Since there is no convergence time associated with PLL or DLL dynamics, unlike in DCG, transitions in clock rate do not result in glitches. However, the supporting clock rates of DCS are pre-defined so that the new required frequency can not be dynamically added using DCG, the flexibility and usability are less. Besides, more supported clock rates in DCS consume more hardware and it may result in failure of implementation due to the lack of resources.

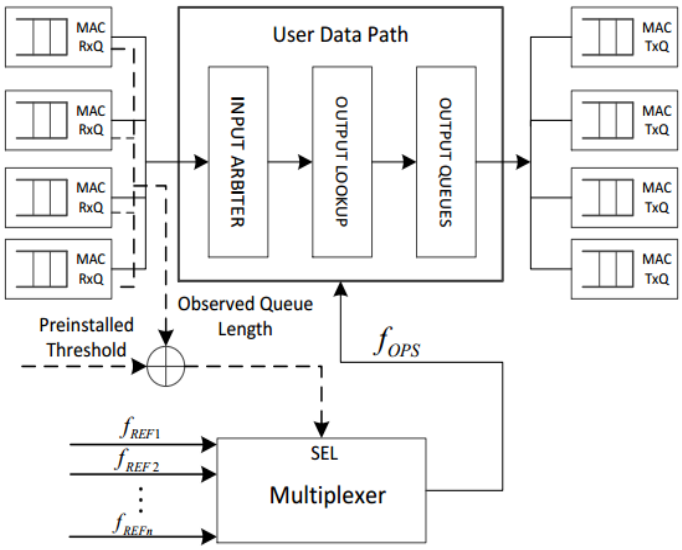


Fig. 4.2 Dynamic Clock Selection (DCS)

4.2 New Frequency Scaling Network Device Design

4.2.1 Adjustable Scheduler

The energy efficiency of the packet processing engine is the focus in this thesis as it is widely believed to be the most energy-consuming component in network devices [98]. It can be done by dynamically tuning the core clock frequencies using in packet processing engine (i.e., frequency scaling) as shown in Fig. 4.3. Frequency scaling allows the performance of digital hardware to be modulated in response to demand, so as to consume no more power than is needed to provide the necessary quality of operation. The goal of saving energy consumption can be achieved by reducing the frequency of the scheduler, effectively increasing the service time, or by decreasing the offered bandwidth.

Energy adaptation feature has been implemented based on the reference design of the NetFPGA-1G/10G SDN switch, presented in Section 3.4.2. To save energy, a new switch needs to be able to dynamically tune its operating frequency without seriously degrading

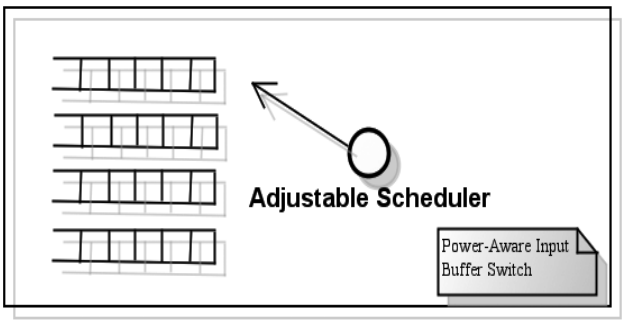


Fig. 4.3 The Adjustable Scheduler

the quality of service. In accomplishing this, a DCG clock engine has been designed for the NetFPGA-10G platform and a DCS engine was designed for the NetFPGA-1G platform.

4.2.2 Frequency Scaling on NetFPGA-10G Platform

The existing clock generator (provided by Xilinx IP [96]) has been used to generate the reference frequency, from which circuitry encoded in Verilog generates new clocks which are fed to the user logic, thus implemeting the power scaling. The NetFPGA-10G platform provides flexible and efficient clock distributions for the core FPGA and other onboard peripheral chips, in which, one differential and three single-ended clocks directly connect to the FPGA, as shown in Fig. 4.4:

- one 100MHz clock coming from the host PC generates a pair of differential clocks which are used for controlling the PCI data communication;
- four independent SPF+ (10Gbps) interfaces are supported by an independent clock circuit that either provides a 125MHz or a 156.25MHz clock;
- a 100MHz clock provided by an onboard crystal is used as a core clock in the FPGA. One PLL resource has been used to produce five different clock frequencies, which are 160MHz, 125MHz, 100MHz, 80MHz, and 50MHz.

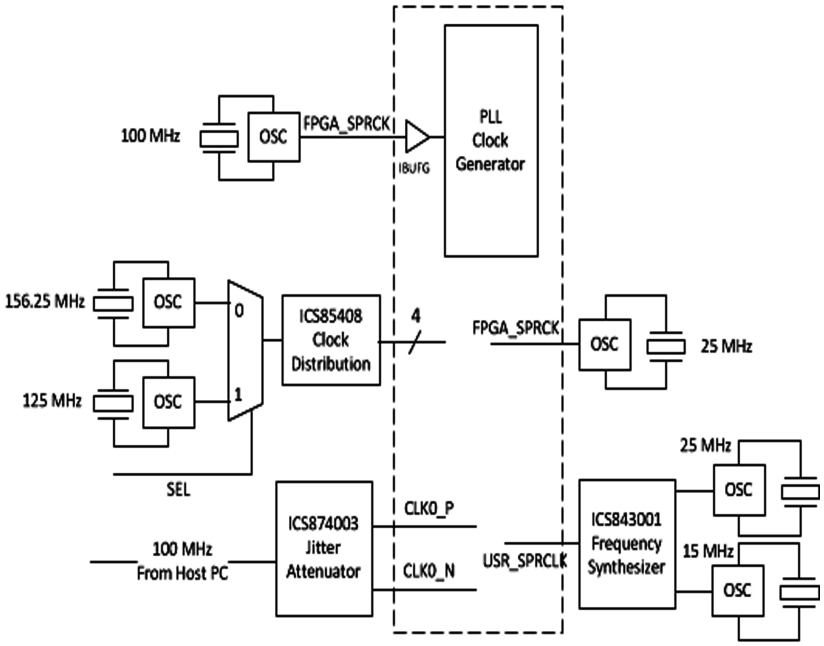


Fig. 4.4 Clock Distribution on the NetFPGA-10G Platform

- a frequency synthesizer (ICS843001) is used to scale the clock in the system and the clock that feeds to the PHY chip is not changed so that the probability of losing packets in input port becomes higher when the processing engine runs at a lower frequency.

A frequency setting module to scale the operating clock has been implemented. This module is connected to the DMA engine via a AXI Lite slave interface so that the internal registers can be accessed from the host over PCIe. This module will be controlled by Local Control Policies (LCP) which is to be dicussed in Section 4.4.2. The ports of the frequency setting module are assigned to GPIO pins (an onboard UART connector used for debugging) and a LogicPort Logic Analyzer [99] is used to verify the design. The generated clock signals are shown in Fig. 4.5.

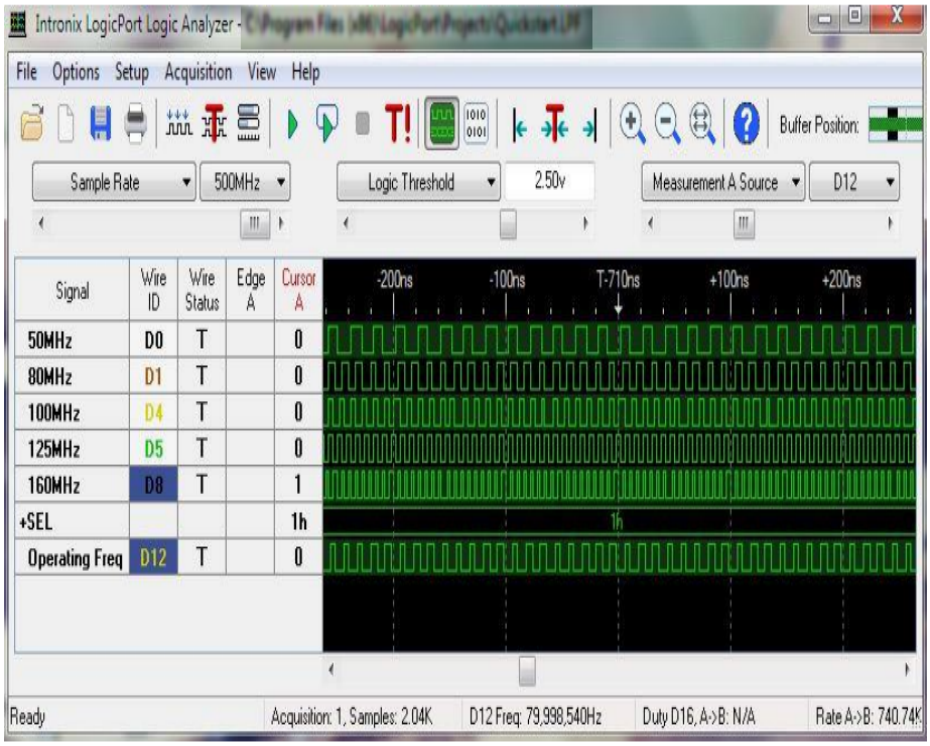


Fig. 4.5 Captured Clock Signals for the NetFPGA-10G Platform

4.2.3 Frequency Scaling on NetFPGA-1G Platform

The frequency scaling procedure has also been implemented on the NetFPGA-1G platform. A novel clock dividing module has been designed and inserted into the scheduler module of the reference implementation on the core Virtex II FPGA chip (known as UFPGA) of 1G platform, without changing the SRAM module and PHY (such as GMII in NetFPGA-1G platform). This energy proportional design generates a set of pre-defined clocks according to $f_n = 125MHz/2^n$, where n is a positive integer and 125MHz clock is provided by an onboard crystal. This design eventually provides five scalable frequency options: 125MHz, 62.5MHz, 31.3MHz, 15.6MHz and 7.8MHz. Fig. 4.6 shows the clock design at the top level. A setting module has also been implemented to dynamically select an appropriate clock from the pre-generated clocks.

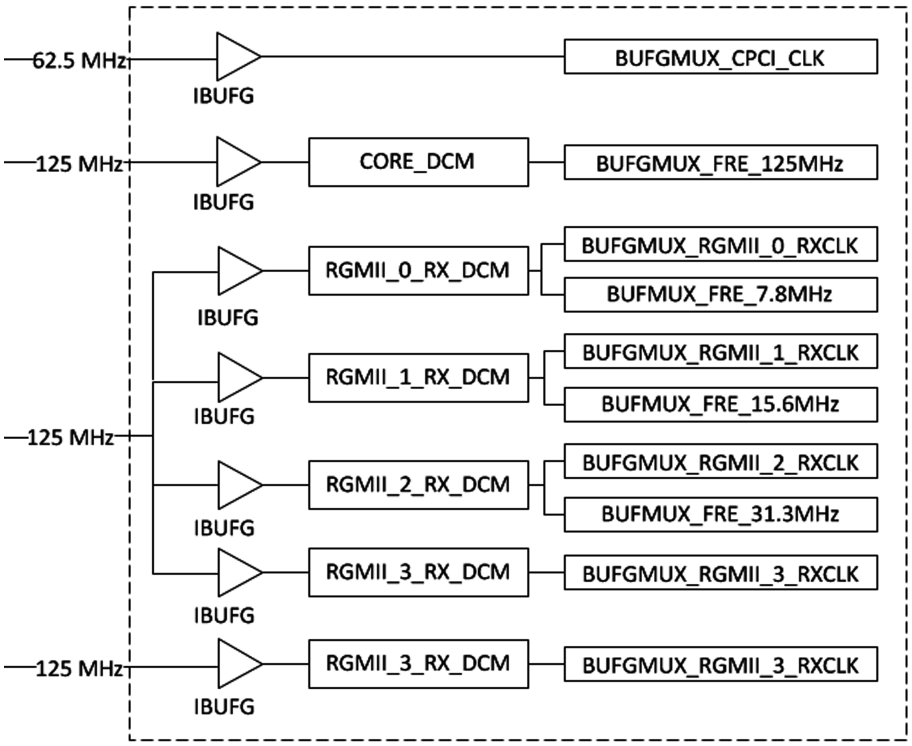


Fig. 4.6 Top Level View of the Scaling Clock on the NetFPGA-1G

A different frequency scaling design on NetFPGA-1G has been reported in [100], in which, the power scaling mechanism is implemented by adding two functional blocks, which are a Clock Controller and a Clock Divider. The Divider block generates the operating frequencies from the core clock (125MHz) and the PCI clock (62.5MHz) in accordance with the feedback from the Clock Controller block. This approach reuses the DCM (Digital Clock Manager) to generate the new frequencies and consumes more hardware resources than our design.

4.3 Evaluation of Power Consumption

Various experiments have been designed and conducted to evaluate the energy consumption of the frequency scaling NetFPGA-1G and 10G SDN switches. These are described below.

4.3.1 Experimental Setup

4.3.1.1 The NetFPGA-10G Platform

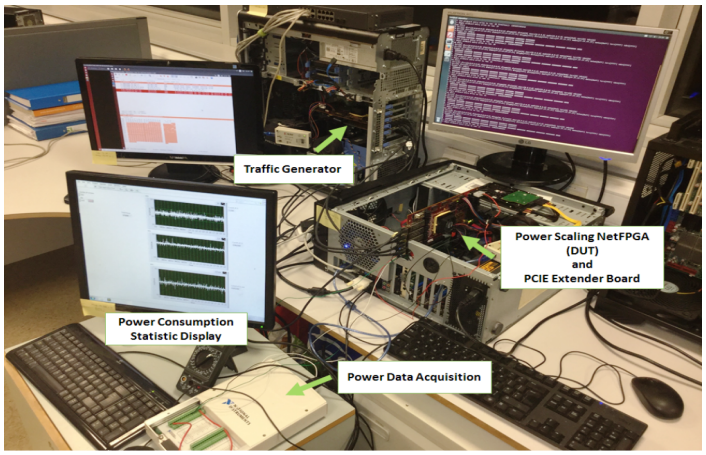
The NetFGPA-10G platform can be operated in two modes. The board is solely driven by an external 12V ATX supply powered for Stand-alone Mode, but it needs to be powered by both External 12V ATX interface and via the PCI Express interface in PCIe Mode. The external ATX power supply powers the DRAM, SRAM and all four PHY chips (Broadcom AEL2005) in PCIe Mode. The PCIe power supply powers the FPGA. The host PC's power supply, from which the ATX connector is powered, and which powers the host motherboard and thus the PCIe BUS, is a CORSAIR Vx550W. For a concurrent and comparative analysis of power consumption of the device, the power utilization needs to be measured on both PCIe and the ATX supply (using the PCIeEXT-16 Smart PCIe Bus Extender [101]). NI USB-6251 [102] has been used as the DAQ (Data Acquisition). Fig. 4.7 depicts the measurement environment schema as well as the real-life testing environment.

4.3.1.2 The NetFPGA-1G Platform

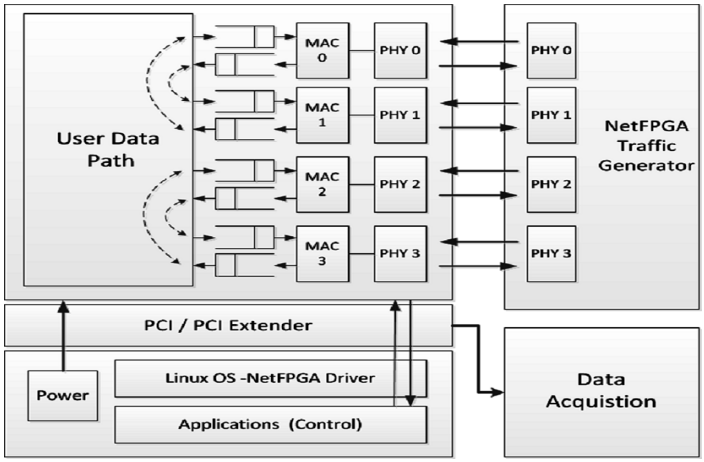
The test environment for NetFPGA-1G Platform is on another aspect as same as for the 10G. In testbed, an Ultraview PCI extender, PCIEXT-64U [101], is used together with a data acquisition (NI USB-6251, same as used for NetFPGA-10G testbed) for data acquisition to measure the NetFPGA-1G power consumption with a sampling rate of 764KHz on the DAQ, and all experiments are undertaken with a room temperature of 23 degrees centigrade.

4.3.1.3 Packet Generator

As a commercial packet generator was unavalable in the lab, a second NetFPGA-10G board was used as a packet generator. In comparison with the OSNT (Open Source Network Tester [103]) project, the design is simplified to directly generate various traffic (for example,



(a) Experimental Testbed



(b) 1G/10G Testbed Schema

Fig. 4.7 Experimental Testbed Setup using NetFPGA-1G/10G

different packet length and different load) in NetFPGA rather than accessing RAM and replaying the PCAPed packets. The packet generator is able to generate up to 8 Gbit/s traffic on each port.

A flow table is used to configure paths on the board to interface it to the packet generator as in Table 4.1.

Table 4.1 Ports Forward to Traffic Generator

Port of Device Under Test	Connected Port of Packet Generator
1	2
2	1
3	4
4	3

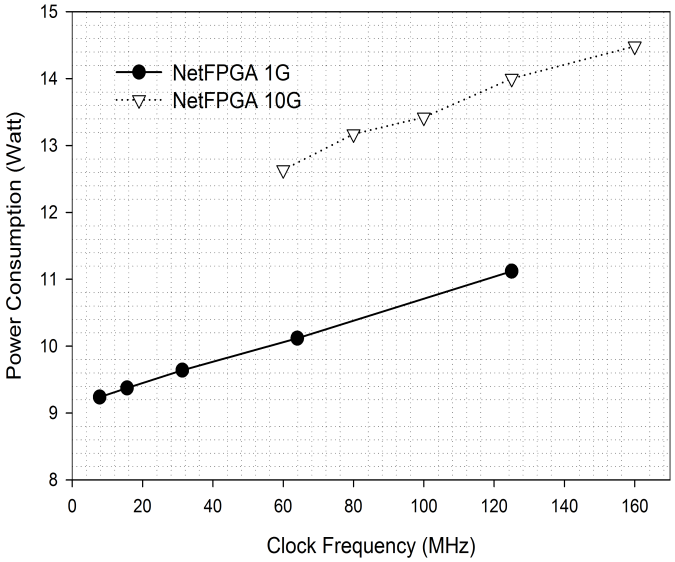


Fig. 4.8 Power Consumption v. Operating Frequency

4.3.2 Performance Evaluation

4.3.2.1 Quiescent Power Consumption

An experiment was undertaken to measure the *Quiescent Power Consumption*[3], to determine the relationship between the clock frequency and power consumption in the absence of any input traffic.

Fig. 4.8 depicts the results for both NetFPGA-1G and 10G platforms. It can be concluded that the relationship between quiescent power consumption and operating frequency is, to a

[3]Quiescent Power Consumption is defined by Xilinx as "the power drawn by the device when it is powered up, configured with user logic and there is no switching activity." [104]

very close approximation, a linear one. This result is used in Section 4.5 to develop a model of quiescent power consumption for all NDs using CMOS circuitry.

4.3.2.2 Variation with Traffic Rate

Fig. 4.9 shows the energy consumption of the NetFGPA-1G and 10G platforms for various levels of traffic load. Evidently, the relationship between power consumption and traffic rate is, to a very good approximation, a linear one. Furthermore the slopes of the best-fit lines through the data are very similar for each system clock rate. It follows that the variation in power consumption as traffic rate changes is largely independent of the system clock rate.

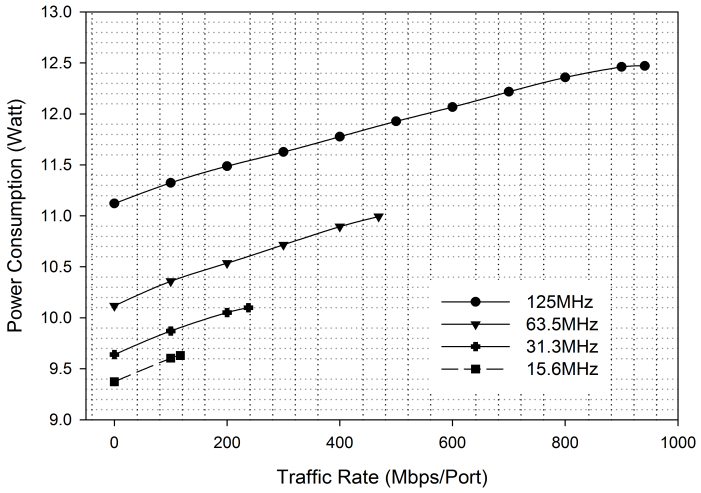
4.3.2.3 Variation with Traffic Shape

The impact on power consumption of varying characteristics of the incoming traffic, such as packet length and inter-arrival time, has also been studied on the 1G and 10G platforms. A series of experiments has been conducted to explore the power consumption distributions in various scenarios. The packet generator was configured to generate traffic with the following properties.

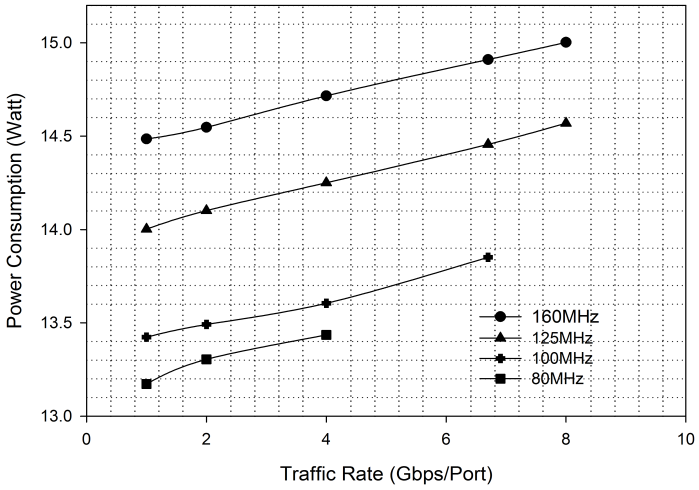
1. Each burst generated has a duration of 100 seconds (a value chosen to ensure that sufficient packets are in each burst for the mean of measurements to be statistically valid);

2. The interval between each burst is 40 seconds (sufficient to ensure independence of the results for successive bursts);

3. Within each burst, the packet length is drawn from a uniform distribution. The mean is common within a burst, but varies between bursts;



(a) Power Consumption v. Traffic Rate on NetFPGA-1G



(b) Power Consumption v. Traffic Rate on NetFPGA-10G

Fig. 4.9 Power Consumption v. Traffic Rate

4. Within each burst, the interval between successive packets has a exponential distribution;

5. Ten bursts are generated in succession, with progressively longer mean packet lengths. The mean inter-departure time is adjusted so that the offered load is the same for all bursts in the sequence;

6. The above sequence of bursts is repeated, with successively higher levels of offered load.

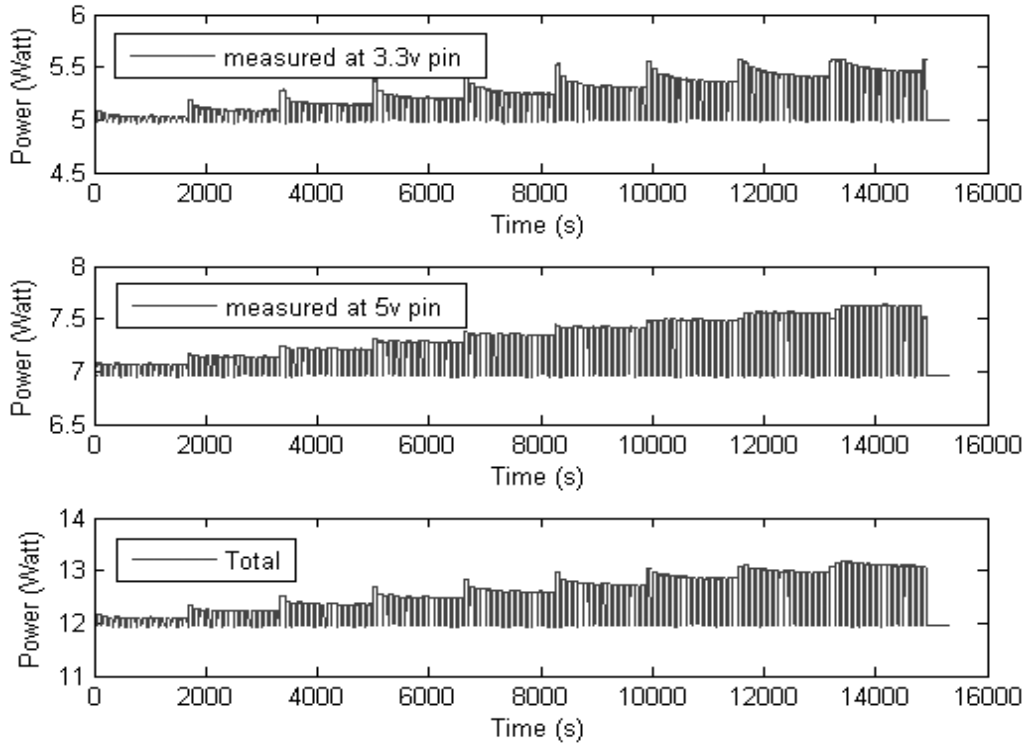


Fig. 4.10 Experimental Process

The Figure 4.10 shows the measured power consumption results of the experiments collected from the 3.3V and 5V pins respectively and the total power consumption which is the sum of the 3.3V power consumption and the 5V power consumption. The experimental results are presented in Fig. 4.11.

The power consumption in Fig. 4.11 falls with increasing inter-arrival time, because of the reduced number of packet headers to be processed. Fig. 4.13a shows the power consumed by the NetFPGA-10G board at various operating frequencies and incoming packet length when the traffic load is 1 Gbit/s. Clearly power consumption decreases as the frequency is reduced. Fig. 4.13b and 4.13c show the same trend at higher traffic load. As shown in Fig. 4.12, if traffic load is same, smaller packets consume more power, as more packet headers need to be processed. It can be concluded that the traffic load has a higher impact on power consumption than the variable packet length.

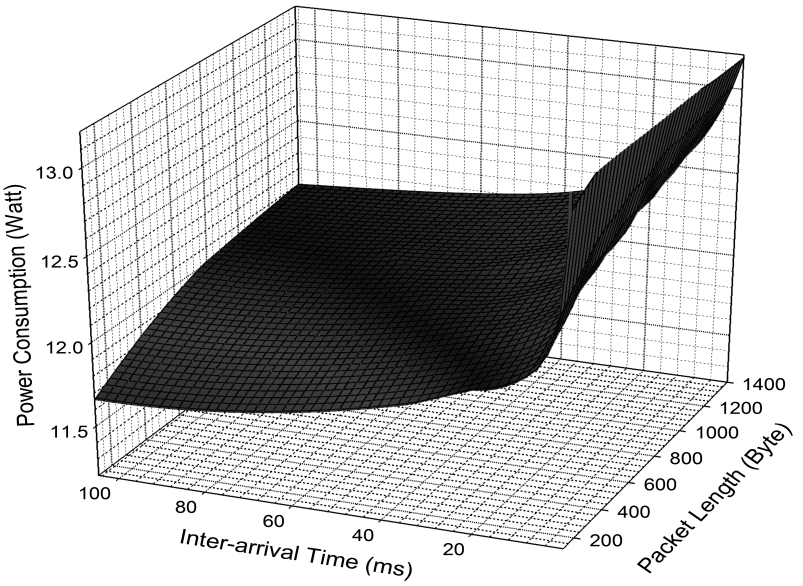


Fig. 4.11 Power Consumption: Packet Length v. Inter-arrival Time

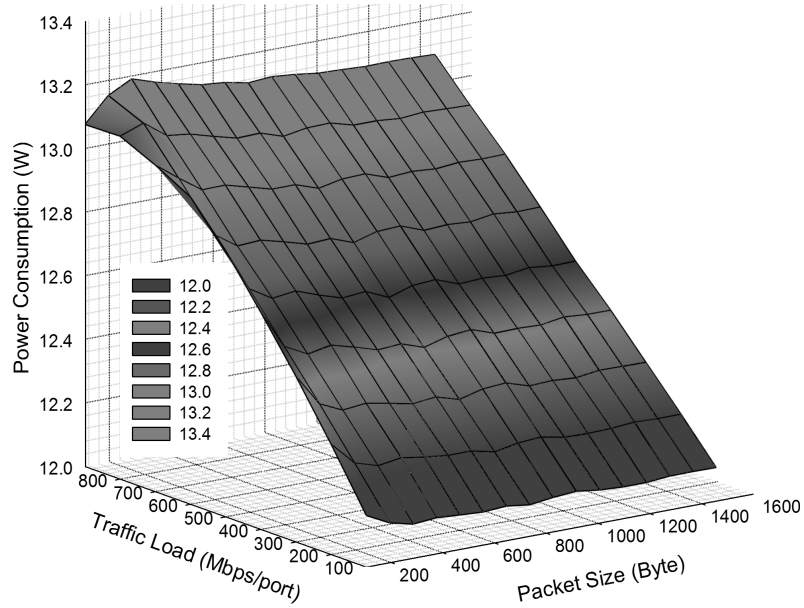
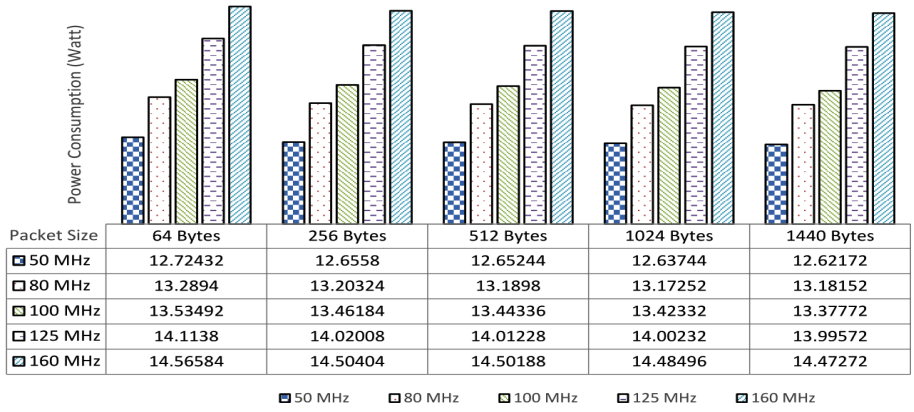


Fig. 4.12 Power Consumption: Packet Length v. Traffic Load

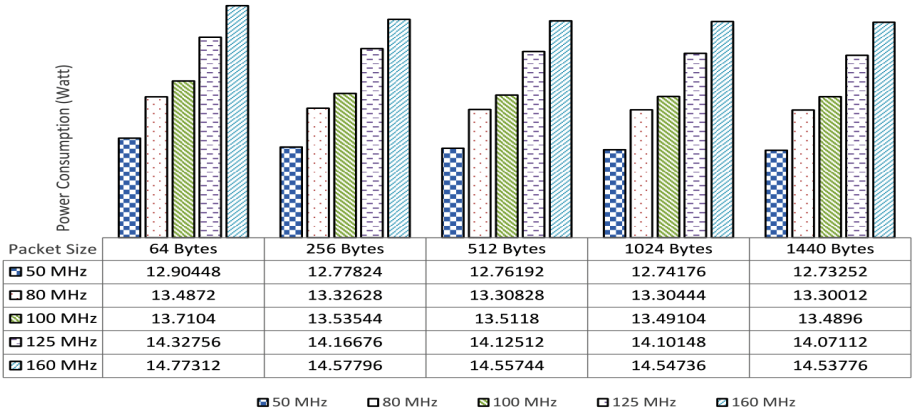
4.3.2.4 Scaling Transition Time

The transition time is defined as the time taken to switch the system between two operating frequencies. A long transition time can negatively impact QoS, due to packet loss during the transition, especially in the input buffer. Quantifying this transition time



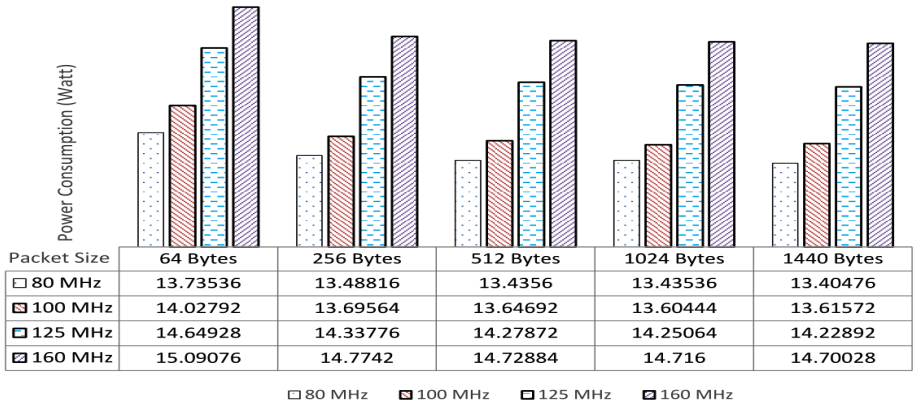
Packet Size	64 Bytes	256 Bytes	512 Bytes	1024 Bytes	1440 Bytes
50 MHz	12.72432	12.6558	12.65244	12.63744	12.62172
80 MHz	13.2894	13.20324	13.1898	13.17252	13.18152
100 MHz	13.53492	13.46184	13.44336	13.42332	13.37772
125 MHz	14.1138	14.02008	14.01228	14.00232	13.99572
160 MHz	14.56584	14.50404	14.50188	14.48496	14.47272

(a) Load is 1 Gbit/s



Packet Size	64 Bytes	256 Bytes	512 Bytes	1024 Bytes	1440 Bytes
50 MHz	12.90448	12.77824	12.76192	12.74176	12.73252
80 MHz	13.4872	13.32628	13.30828	13.30444	13.30012
100 MHz	13.7104	13.53544	13.5118	13.49104	13.4896
125 MHz	14.32756	14.16676	14.12512	14.10148	14.07112
160 MHz	14.77312	14.57796	14.55744	14.54736	14.53776

(b) Load is 2 Gbit/s

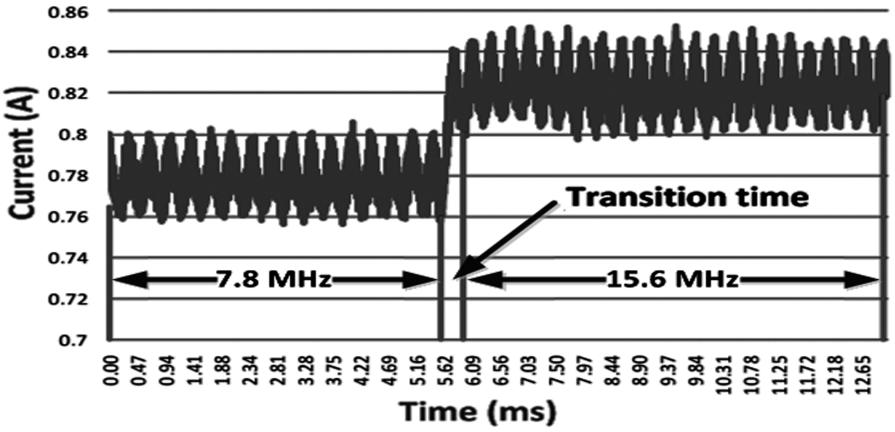


Packet Size	64 Bytes	256 Bytes	512 Bytes	1024 Bytes	1440 Bytes
80 MHz	13.73536	13.48816	13.4356	13.43536	13.40476
100 MHz	14.02792	13.69564	13.64692	13.60444	13.61572
125 MHz	14.64928	14.33776	14.27872	14.25064	14.22892
160 MHz	15.09076	14.7742	14.72884	14.716	14.70028

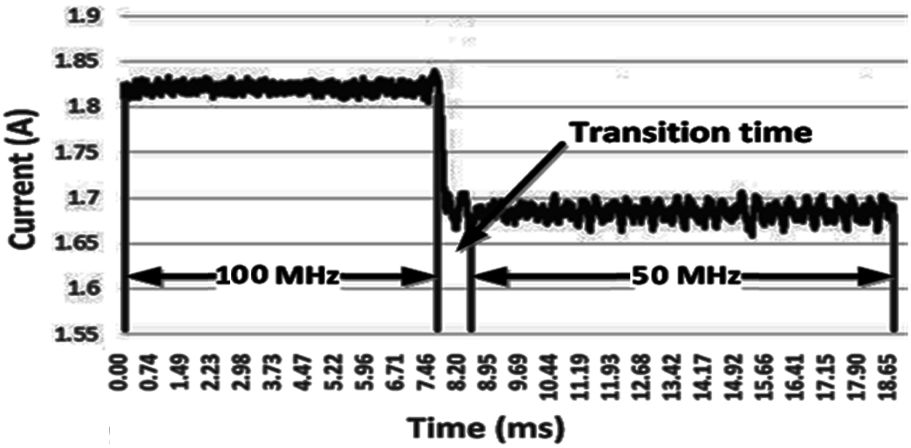
(c) Load is 4 Gbit/s

Fig. 4.13 Power Consumption v. Operating Frequency

is important in order to know the penalty incurred by triggering a clock transition. The transition time is inferred from the power consumption measurements taken in an interval around the instant the transition is triggered, as shown in Fig. 4.14. The transition time is largely independent of the current traffic load and also of the operating frequency. The average transition time of NetFPGA-1G board is around 0.3 millisecond, but this value is up to 0.6 millisecond for the 10G board. This is because the DCG scheme is used in the 10G switch and the PLL needs more time to lock the generated frequency.



(a) Transition Time Measurement on NetFPGA-1G



(b) Transition Time Measurement on NetFPGA-10G

Fig. 4.14 Transition Time Measurement

4.4 Power Management Mechanism

The power measurements obtained show that energy consumption increases with increasing the clock frequency. However, lowering the clock frequency can only safely be done if traffic conditions permit. The goal is to assure that, at any instant, the device performance capabilities exactly match the traffic requirements without excess capability (which needlessly consumes power) or insufficient capability (which results in poor QoS). Any algorithm to achieve this must be supported by appropriate primitives in the hardware.

Power management primitives provide the possibility of dynamically adapting the trade-off between the energy consumption and the processing capacity of a hardware block (e.g., one or more hardware components) in this way.

4.4.1 Two-Layer Architecture for Power Management

The simplest way to manage the power consumption of devices is to directly manipulate the physical resources. In the case of the frequency scaled SDN switch, this would be the system clock rate. However, such an approach would require a complete redesign for every new piece of hardware. Instead, it will be more effective to decompose the power management into two layers.

As Fig. 4.15 shows, the architecture is decomposed into a physical layer and a logical layer. Extended SDN primitives can be used to pass physical layer properties through the logical layer to the controller, but commands to the physical layer, must pass via the logical layer or the Local Control Policy (LCP) to reach the physical layer. The LCP, as its name suggests, makes decisions based only on local information. The controller, on the other hand, can potentially take a system-wide or even network-wide view, making decisions relating to, and informed by data from, multiple logical resources.

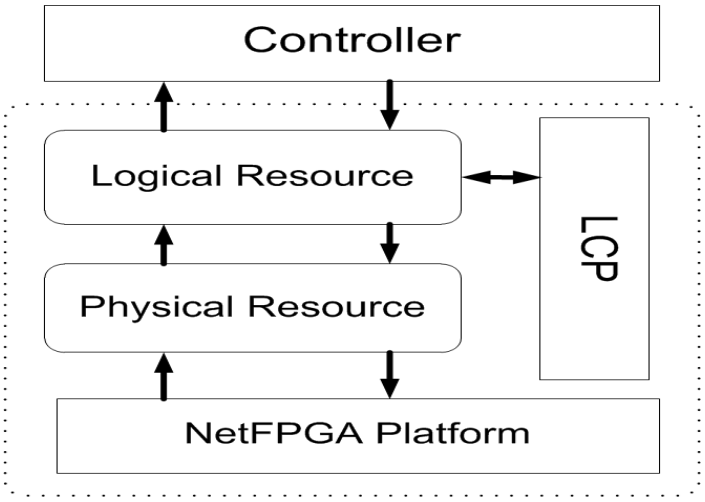


Fig. 4.15 NetFPGA based Two-Layer Architecture

Table 4.2 EAS and Multi-Frequencies

EAS	Frequency in NetFPGA-1G	Frequency in NetFPGA-10G
0	125	160
1	62.5	125
2	31.3	100
3	15.6	80
4	7.8	50
5	Adaptive Scaling	Adaptive Scaling

The logical layer uses the abstraction of "Energy Aware States", described in the ETSI standard describing the green abstraction layer (GAL) in energy efficient network equipment. Table 4.2 lists the relationship between these energy aware states and the frequency options available at the physical layer.

4.4.2 Local Control Policies

As discussed in the previous section, the power surveillance and setting commands from the Controller can only be sent to the logical level. The Local Control Policy (LCP) has the capacity to look up the mapping relationship between logical and physical resources and interpret the higher level commands for the corresponding lower level, so that the appropriate

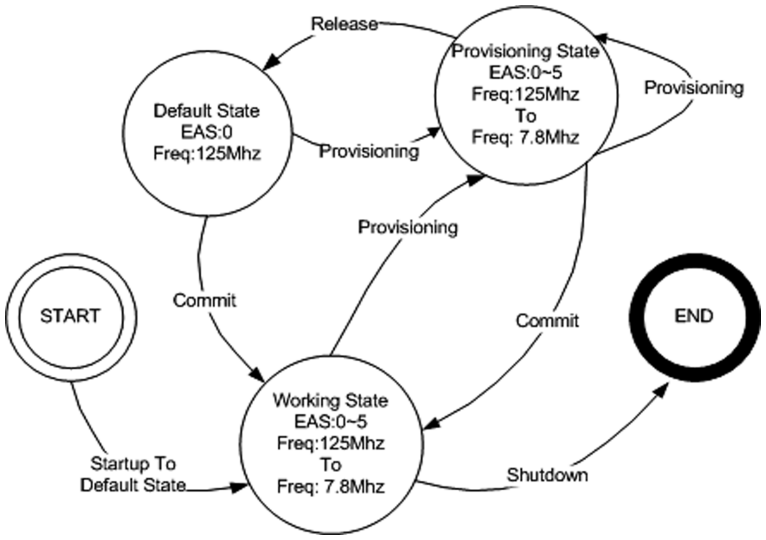


Fig. 4.16 EAS Transition by LCP for the NetFPGA-1G Platform

EAS (in this case, the operating frequency) of the associated hardware modules can be set. In this way, the LCP controls each entity's power consumption and provides a simplified way for the Controller to interact with hardware. The state diagram showing the energy aware states (EAS) and state transitions for the LCP of the NetFPGA-1G as shown in Fig.4.16 (the state transition diagram for the NetFPGA-10G is the same, except for the EAS values). Each EAS is assigned an associated threshold, which responds to either traffic load or packet loss. The LCP monitors the MAC statistic data of each Ethernet port in a particular time interval. If the measured value exceeds the threshold of the current EAS, the switch will automatically shift to the next EAS with higher frequency. If measurement value goes below the threshold of the adjacent EAS, the switch will move to the next lower frequency.

4.4.3 OpenFlow Extension to Support Power Management

OpenFlow exchanges routing configuration data between the controller and switches through a secure channel. Given that power management techniques can be applied to single hardware blocks, and that different hardware blocks can enter different power settings, the OpenFlow abstraction model natively offers the opportunity of mapping logical network

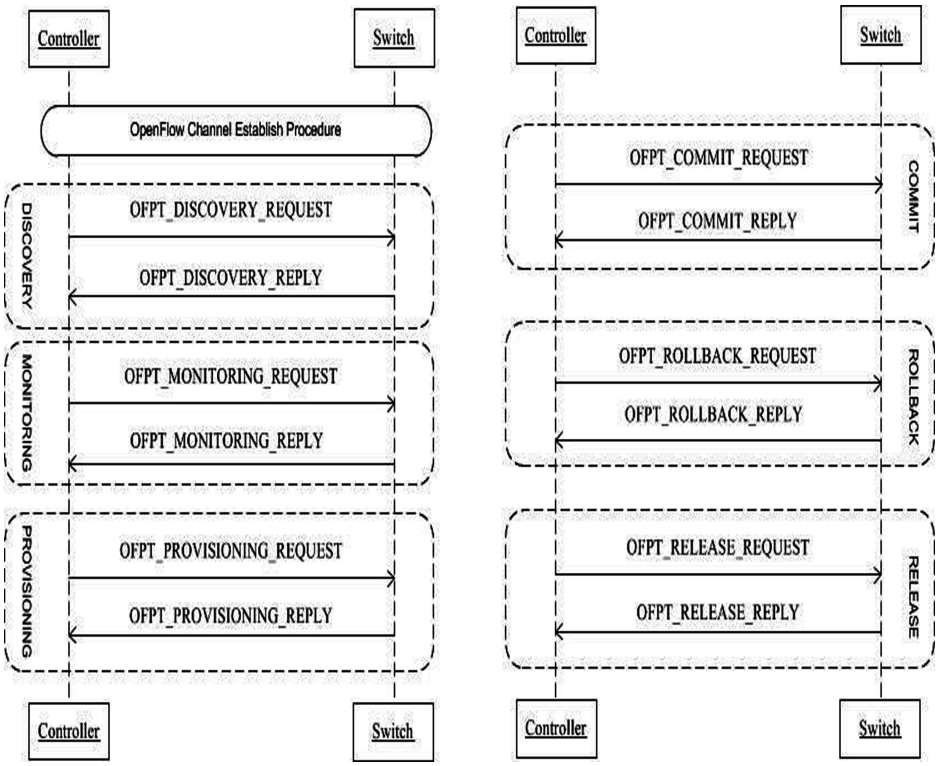


Fig. 4.17 Extended Procedures between Controller and Switch

functions with the power settings of hardware components realising them. Twelve new messages are introduced for enhancing power efficiency among data plane elements. Table 4.3 lists the new messages and their functions; Fig. 4.17 describes the extended procedures required between controller and switches.

4.5 Network Device Power Model

A CMOS device level power model is considered here in order to establish an empirical power consumption model for a real ND. As discussed in Section 3.1, the power dissipation in CMOS circuits comprises dynamic and static power consumption. The dynamic power consumption is due to the switching activities of circuit nodes (the charging and discharging of capacitances) and the static power consumption is caused by the logical states of the

Table 4.3 Extended OpenFlow Primitives

Primitive	Description
OFPT_DISCOVERY_REQUEST and OFPT_DISCOVERY_REPLY	Use for Controller to retrieve information about available power states and other entity's descriptive information such as correspondent IP address, the power gain in terms of the maximum power consumption etc.
OFPT_PROVISIONING _REQUEST and OFPT_PROVISIONING_REPLY	Allows the Controller to configure an entity into a specified power states, but this change is provisional, needs to be effective by using Commit command.
OFPT_MONITORING _REQUEST and OFPT_MONITORING_REPLY	Controller uses Monitoring to request the data of current EAS and power consumption of an entity.
OFPT_COMMIT_REQUEST and OFPT_COMMIT_REPLY	Make the provisional power state changes of entity effective.
OFPT_ROLLBACK_REQUEST and OFPT_ROLLBACK_REPLY	Discard the provisional state changes and rollback to the last commit point.
OFPT_RELEASE_REQUEST and OFPT_RELEASE_REPLY	Restore the state of an entity back to the default configuration.

circuits.

$$\begin{aligned} P_{total} &= P_{dyn} + P_{stat} \\ &= \sum_{i \in net} (C_i V_i^2 f_{0\to 1}) + P_{stat} \end{aligned} \tag{4.2}$$

where C_i denotes the capacitance of the capacitor i, V_i is the power supply voltage and $f_{0\to 1}$ is the switching activity, which represents the frequency of energy-consuming transitions, which include both L to H and H to L transitions.

In a synchronized circuit, we may re-write the switching activity $f_{0\to1} = P_{0\to1} \cdot f_{clk}$, where $P_{0\to1}$ denotes the probability of an input signal transition during a clock cycle. The switching activity is known as toggle-rate as defined by Xilinx. Toggle-rate is the rate at which a network or logical element switches compared to its input. Thus, toggle-rate reflects how often an output changes relative to a given clock input and can be modeled as a percentage [96]. Evidently, this probability depends on the packet arrival rate. We describe this relation as $P_{0\to1} = \zeta(r)$, in which r denotes the packet arrival rate. We assume that $P_{0\to1}$ is identical for the entire circuit[4]. Thus, Eq. (4.2) becomes:

$$\begin{aligned} P_{total}(f_{clk}, r) &= f_{clk} P_{0\to1} \sum_{i \in net} (C_i V_i^2) + P_{stat} \\ &= f_{clk} \zeta(r) \sum_{i \in net} (C_i V_i^2) + P_{stat} \\ &= f_{clk} \zeta(r) C_{total} V_{swing}^2 + P_{stat} \end{aligned} \tag{4.3}$$

where C_{total} is the sum of all capacitances and V_{swing} is the voltage swing. Eq. (4.3) can be represented as $P_{total} = f(clk, r) + P_{stat}$, where clk denotes the clock frequency. Expanding this as a Taylor Series about $r = 0$, we get:

$$\begin{aligned} P_{total}(c, r) &= f(c, r) + P_{stat} \\ &= f_0(c) + f_1(c) r + f_2(c) r^2 + \cdots + f_n(c) r^n + P_{stat} \\ &= \sum_{n=0}^{\infty} f_n(c) r^n + P_{stat} \end{aligned}$$

In practice, $f_n(c) \approx 0 \quad for \quad n > 1$, resulting in:

$$P_{total}(c, r) \approx f_0(c) + f_1(c) r + P_{stat} \tag{4.4}$$

[4]this assumption has been widely used, for example in [105].

For a given clock rate C, the quiescent power consumption (i.e. that consumed when no packets are arriving) can be measured to determine the value of $f_0(c) + P_{stat}$, and the $f_1(c)r$ can be estimated by fitting the measured power to Eq. (4.4).

The maximum throughput supported by the hardware network devices can be approximately calculated as:

$$R_{max} = C \cdot W \tag{4.5}$$

where R_{max} is maximum supported throughput in bits/second. C is the clock frequency used in circuit, which is measured in Hz. W denotes the width of internal bus in bit. For example, the reference NetFPGA-1G design supports 125 MHz $\times$ 64 bist=8 Gbits/sec, and each input port can achieve 1 Gbits/sec. NetFPGA-10G board supports 160 MHz $\times$ 256 bits=40 Gbits/sec and each input port can achieve 10 Gbits/sec. Considering the proposed NDPM (Eq. (4.4)), applying Eq. (4.5) will give the upper bound of power consumption of a network device. Fig. 4.20 shows the comparison of measured maximum throughput on NetFPGA-1G evaluation board and its theoretical value.

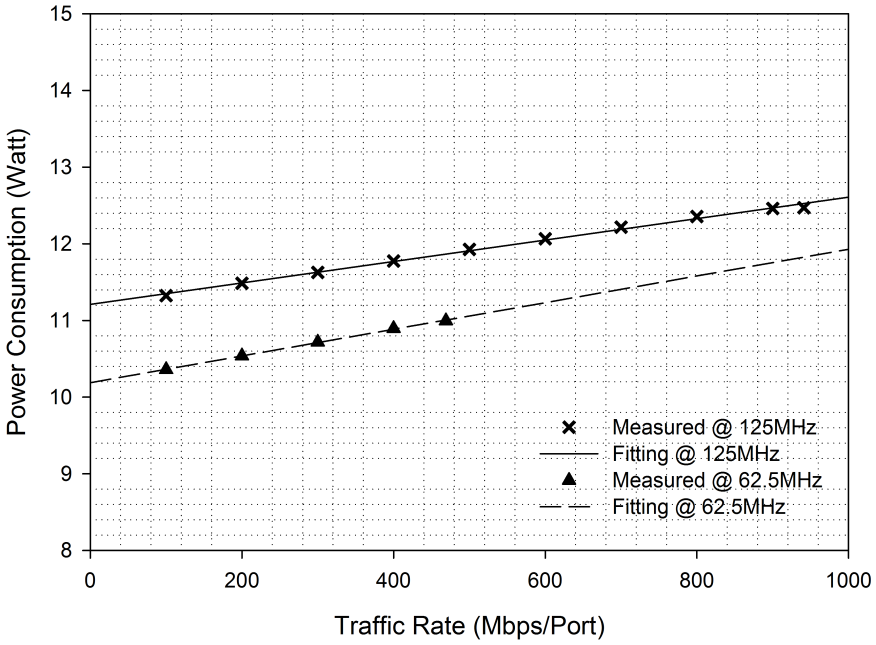
4.6 Model Evaluation

A testbed is developed, as shown in Section 4.3.1, to evaluate the NDPM which is derived in Section 4.5.

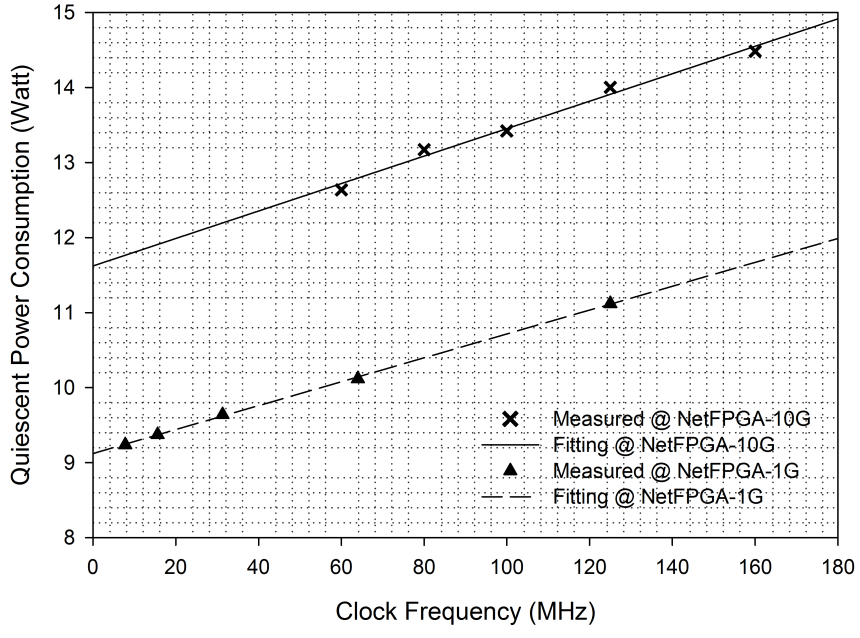
4.6.1 NDPM Validation

NDPM (Eq. (4.4)) can be rewritten as

$$P_{total}(c,r) \approx P_{dynamic}(c,r) + P_{quiescent}(c) \tag{4.6}$$

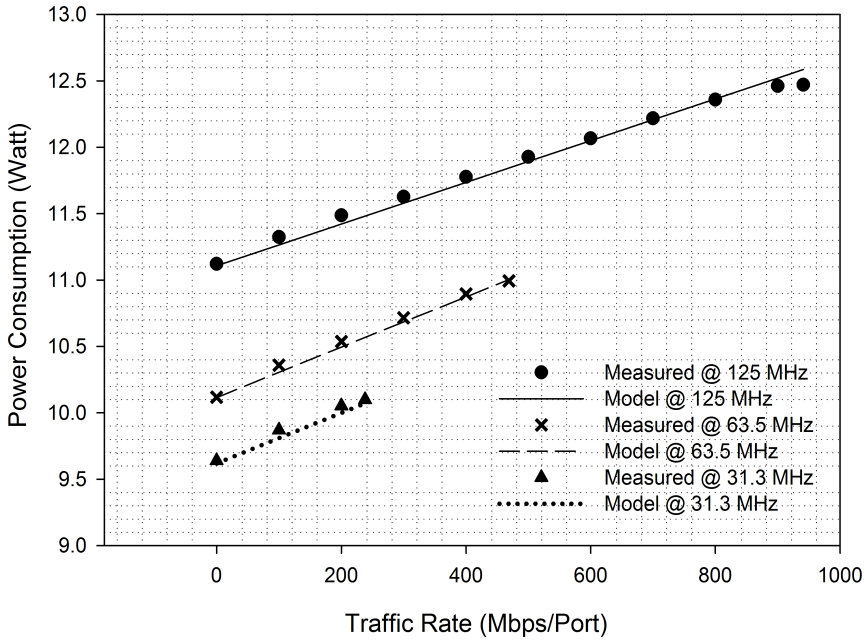


(a) Slope Fitting

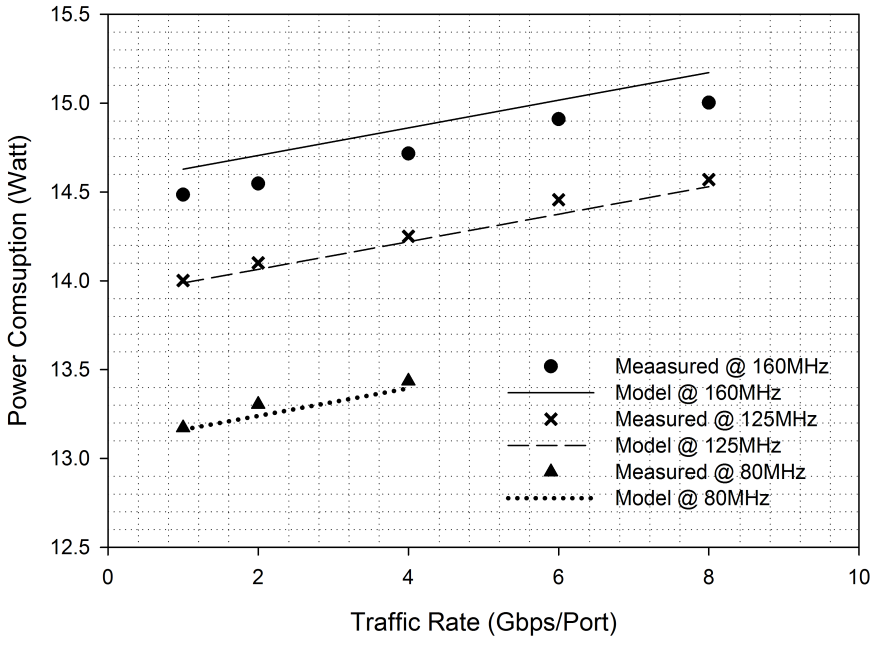


(b) Intercept Fitting

Fig. 4.18 NetFPGA 1G/10G Power Model Measurements: Fittings



(a) NetFPGA 1G Model Validation



(b) NetFPGA 10G Model Validation

Fig. 4.19 NetFPGA 1G/10G Power Model Measurements: Validations

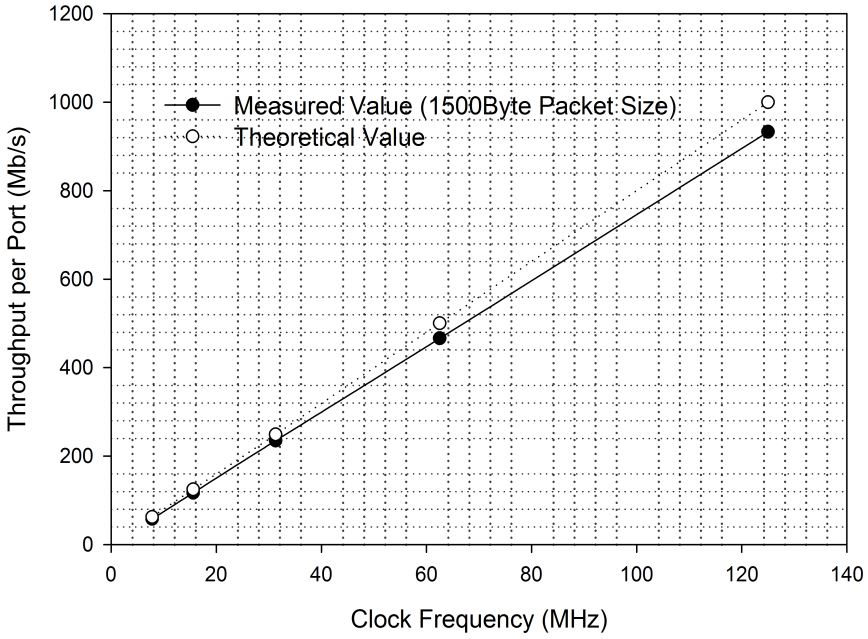


Fig. 4.20 Maximum Throughput on NetFPGA-1G

where $P_{quiescent}$ is the quiescent power consumption with a clock frequency of c, and $P_{dynamic}$ denotes the dynamic power consumption arising from the processing of incoming packets arriving at a rate of r bits/s. The quiescent power consumption is defined as the power consumption with no arriving traffic (i.e, when $r = 0$), it is an inherent characteristic of a ND, which can be measured and modelled using the linear least squares fitting technique [106]. Fig. 4.18b shows the close match obtained by fitting the measured quiescent power consumption to this model on both NetFPGA-1G and 10G platforms.

Fig. 4.18a depicts the fitting of the measured dynamic power consumption to the formula $P_{dynamic} = f(c,r)$ with varying traffic rate r for two values of c. This is assumed to the linearly dependent on the arrival rate and thus we may write that:

$$P_{dynamic}(c,r) = f(c)r$$

For a fixed clock rate, $f(c)$ will be a constant. Since the slope of both fitted lines is similar, it is clear that $f(125MHz) \approx f(62.5MHz)$. The model can be further simplified by expending

Table 4.4 NDPM Evaluation on NetFPGA-1G/10G

Device Under Test	Model	Average Error (Watt)	Variation (Watt)
NetFPGA-1G	$0.0019r + 0.0159c + 9.1204$	-0.01979	0.051879
NetFPGA-10G	$0.0824r + 0.0183c + 11.6230$	0.031462	0.022216

$f(c)$ as a Taylor Series in c:

$$f(c) = f(0) + c\frac{\partial f}{\partial c}\Big|_{c=0} + c^2\frac{\partial^2 f}{\partial c^2}\Big|_{c=0} + \cdots$$

If we assume that $\frac{\partial f}{\partial c} = 0,\ \forall\, c$, we can approximate $f(c)$ as a constant. The total power consumption is then approximately of the form:

$$\begin{aligned} P_{total}(c,r) &\approx P_{dynamic}(r) + P_{quiescent}(c) \\ &\approx ar + bc + d \end{aligned} \tag{4.7}$$

where r is the traffic rate, c is the clock frequency, and a, b and d are constants. This simplified NDPM has been evaluated on both NetFPGA-1G and 10G platforms and the results are shown in Fig. 4.19 and Table. 4.4.

4.7 Summary

This chapter described a green SDN network which is composed of OpenFlow NetFPGA-1G and 10G switches configured using a power scaling mechanism. The operating clock frequencies of these green switches can be dynamically adjusted to balance performance and energy consumption. The power scaling mechanism can be managed effectively by the Green Abstraction Layer or via SDN extensions.

It follows a description of a linear power consumption model of frequency scaled NetFPGA 1G/10G platforms, chosen as representative CMOS network devices, based on careful measurements of actual power consumption. This linear power consumption model (Network

Device Power Model or NDPM) can produce a good approximation of the experimental testbed results.

Chapter 5

Frequency Scaled Control Models and their Performance Evaluation

5.1 Introduction

The power consumption of network devices depends on factors such as the arrival and service rates, queue length etc., a control model will choose the appropriate energy aware state by processing one or more of these factors. Ideally, this processing will exactly model the dependence of power consumption on all these factors, so that the chosen state is always optimal. In this chapter, such a model is developed based on probabilistic assumptions about packet arrival rate and queue length. The model is complicated by the dependence of service time not only on packet length, but on the clock state of the frequency scaled switch. This behaviour is modelled by representing the service rate by a continuous time Markov chain. The resulting queuing model is too complicated to model analytically or numerically and will instead be emulated using computer simulation.

Two control policies will be described and their dynamics will be explored using the simulation model.

Table 5.1 Mathematical Notations in Models

Symbols	Description
L	the threshold step size.
M	the number of threshold steps, the system provides $M+1$ different service rates from μ_0 to μ_M.
$X(t)$	a stochastic process with the state space S.
S	the state space of a continuous-time Markov Chain, $S = 0, 1, 2, \cdots$
i	any state $i \in S$
$o(\Delta t)$	any function goes to zero with Δt faster than Δt itself, namely, $\lim_{\Delta t \to 0} \frac{o(\Delta t)}{\Delta t} = 0$.
λ	the mean arrival rate of the Poisson arrival process.
μ_k	the mean service rate. where $0 \leq k \leq M$, and $\mu_0 < \mu_1 < \cdots < \mu_M$.
ρ_k	the traffic intensity $\rho_k = \frac{\lambda}{\mu_k}$ in one server system.
$p_i^{\mu_k}(t)$	the probability that the service rate is μ_k, and the system in state i, at time instant t, i.e., $p_i^{\mu_k}(t) = P\{\mu(t) = \mu_k, N(t) = i\}$.
$p_i^{\mu_k}$	the probability that the service rate is μ_k, and the system in state i while the system is in equilibrium. $p_i^{\mu_k} = \lim_{t \to \infty} p_i^{\mu_k}(t)$.
$r_{i,j}$	average rate of transitions $\mu_k \to \mu_{k-1}$.
Θ	service process generator matrix.
Υ	service rate matrix.

5.2 System Modelling and Control Policies

In this section, the model to abstract the operating behavior of a frequency scaling system is first described. Next, an **Escalator Policy** is presented which extends the single-threshold ALR model described in [55]. This policy can potentially give rise to an excessive number of clock rate transitions, and a **Hysteresis Policy** is proposed to dampen this behaviour. All the notation to be used in this chapter is summarized in Table 5.1.

5.2.1 Frequency Scaling System Model

The following assumptions are made in modelling the frequency scaling system:

- the arrival process has an exponentially distributed inter-arrival time with a constant arrival rate (λ);
- the service time is exponentially distributed with a mean duration of $(1/\mu)$ but the service rate μ is dynamically adjustable across a range of values μ_k for $0 \leq k \leq M$ to model the frequency scaling;
- a single packet processing engine forwards the packets on the user data path;
- the packets in the queue are served in a First Come First Serve (FCFS) manner;
- the size of the FIFO input queue is infinite;
- the arrival process and the service process are independent of each other.

In this model, the service rate is controlled by a Continuous Time Markov Chain (CTMC). There are $M+1$ phases (states) in the modulated chain and if the system is in the *i-th* phase, the service rate is adjusted to $\mu_i, i = 0, 1, 2, \cdots, M$ by using the proposed frequency scaling technique. Therefore the service process is uniquely determined by an infinitesimal generator matrix Θ and a service rate matrix Υ, which are defined as:

$$\Theta = \begin{pmatrix} -r_0 & r_{01} & \dots & r_{0M} \\ r_{10} & -r_1 & \dots & r_{1M} \\ \vdots & \vdots & \ddots & \vdots \\ r_{M0} & r_{M1} & \dots & -r_M \end{pmatrix} \tag{5.1}$$

where $r_i = \sum_{j=0, j!=i}^{M} r_{ij}$. The service rate associated with each phase is a diagonal matrix Υ,

$$\begin{aligned}\Upsilon =& diag(\mu_0, \mu_1, \cdots, \mu_M) \\ =& \begin{pmatrix} \mu_0 & 0 & \dots & 0 \\ 0 & \mu_1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \mu_M \end{pmatrix}\end{aligned} \tag{5.2}$$

The entries in the generator matrix Θ depend on the control policy in place. The following assumptions are made about control polices:

- An increase (resp. decrease) in queue length, if it triggers a change in service rate, will cause an increase (resp. decrease) in service rate;
- A service rate transition can only occur immediately following when a packet is served and leaves the system;
- There are $M+1$ service rates, μ_0 to μ_M, where $\mu_0 < \mu_1 < \cdots < \mu_M$;
- The system equilibrium condition (ergodicity) is met, $\lambda < \mu_M$.

5.2.2 Escalator Policy

The Escalator Policy operates as follows.

- If the queue length reaches the threshold $kL (0 < k \leq M)$ from $kL-1$, and the current service rate is μ_{k-1}, the service rate is increased to μ_k,
- If the queue length falls to $kL-1 (0 < k \leq M)$ from kL, and the current service rate is μ_k, the service rate is decreased to μ_{k-1},

- If the queue length exceeds the threshold ML, and the service rate achieves the maximum value μ_M, the service rate will remain at μ_M until the queue length drops below ML.

Fig 5.1 depicts the state transition-rate diagram of the Escalator Policy according to the definitions above.

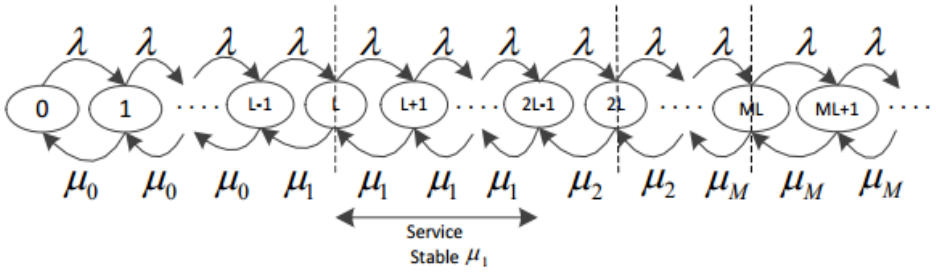


Fig. 5.1 State Transition-Rate Diagram (Escalator Policy)

The entire process can be viewed as an infinite Birth-Death process.

In state $\{i = kL-1, 0 < k \leq M\}$, it has

$$\begin{aligned} p_{kL-1}(t+\Delta t) =& p_{kL}(t)[\mu_k \Delta t + o(\Delta t)] + p_{kL-2}(t)[\lambda \Delta t + o(\Delta t)] \\ &+ p_{kL-1}(t)[1-(\lambda+\mu_{k-1})\Delta t + o(\Delta t)] + o(\Delta t) \end{aligned}$$

or

$$\begin{aligned} \frac{p_{kL-1}(t+\Delta t) - p_{kL-1}(t)}{\Delta t} =& \mu_k p_{kL}(t) + \lambda p_{kL-2}(t) \\ &- (\lambda+\mu_{k-1}) p_{kL-1}(t) + \frac{o(\Delta t)}{\Delta t} \end{aligned}$$

Letting $\Delta t \to 0$ and considering the general equilibrium solution ($t \to \infty, \lim_{t\to\infty} \frac{d}{dt} p_i(t) = 0$), we have

$$\begin{aligned} \mu_k p_{kL} + \lambda p_{kL-2} &= (\lambda+\mu_{k-1}) p_{kL-1} \\ &(0 < k \leq M) \end{aligned} \tag{5.3}$$

Due to the sub-process $\{X(t) = i | kL-1 < i \leq (k+1)L-1, 0 \leq k < M\}$ has a constant service rate μ_k. When the system is in steady states:

$$\begin{aligned} \mu_k p_{i+1} + \lambda p_{i-1} &= (\lambda + \mu_k) p_i \\ (kL-1 < i \leq (k+1)L-1, \quad 0 < k < M) & \end{aligned} \tag{5.4}$$

From Eqs. (5.3) and (5.4), it has

$$\begin{aligned} \lambda p_{kL-2} - \mu_{k-1} p_{kL-1} &= \lambda p_i - \mu_k p_{i+1} \\ (kL-1 \leq i < (k+1)L-1, \quad 0 < k < M) & \end{aligned} \tag{5.5}$$

Considering the equilibrium solution on sub-process $\{X(t) = i | 0 \leq i < L-1\}$,

$$\begin{cases} \lambda p_0 - \mu_0 p_1 = 0, \\ p_{i+1} = (\dfrac{\lambda}{\mu_0}) p_i \quad (0 \leq i < L-1) \end{cases} \tag{5.6}$$

Using Eq. (5.5), it follows that

$$\begin{cases} p_i = (\dfrac{\lambda}{\mu_0})^i p_0, \quad (0 \leq i \leq L-1) \\ p_i = (\dfrac{\lambda}{\mu_k})^{(i-kL+1)} p_{kL-1}, \\ \qquad (kL \leq i < (k+1)L-1, \quad 0 < k < M) \\ p_i = (\dfrac{\lambda}{\mu_M})^{(i-ML+1)} p_{ML-1}, \quad (i \geq ML-1) \\ p_{kL-1} = (\dfrac{\lambda}{\mu_0})^{L-1} \prod_{j=1}^{k-1} (\dfrac{\lambda}{\mu_j})^L \cdot p_0 \quad (0 < k \leq M) \end{cases} \tag{5.7}$$

or

$$\begin{cases} p_i = (\rho_0)^i p_0, \quad (0 \le i \le L-1) \\ p_i = (\rho_k)^{(i-kL+1)} (\rho_0)^{L-1} \prod_{j=1}^{k-1} (\rho_j)^L \cdot p_0, \\ \qquad (kL \le i < (k+1)L, \quad 0 < k < M) \\ p_i = (\rho_M)^{i-ML+1} (\rho_0)^{L-1} \prod_{j=1}^{M-1} (\rho_j)^L \cdot p_0, \quad (i \ge ML) \end{cases} \tag{5.8}$$

where, if $k = 1$, $\prod_{j=1}^{0} (\rho_j)^L = 1$. Calculating p_0 using the normalization equation,

$$p_0 = \{ \sum_{i=0}^{L-1} (\rho_0)^i + \sum_{k=1}^{M-1} \sum_{i=kL}^{(k+1)L-1} (\rho_k)^{i-kL+1} (\rho_0)^{L-1} \prod_{j=1}^{k-1} (\rho_j)^L + \sum_{i=ML}^{\infty} (\rho_M)^{i-ML+1} (\rho_0)^{L-1} \prod_{j=1}^{M-1} (\rho_j)^L \}^{-1}$$

or

$$p_0 = \{ \frac{1-(\rho_0)^L}{1-\rho_0} + (\rho_0)^{L-1} \sum_{k=1}^{M-1} \frac{\rho_k (1-(\rho_k)^L)}{1-\rho_k} \prod_{j=1}^{k-1} (\rho_j)^L + (\rho_0)^{L-1} \frac{\rho_M}{1-\rho_M} \prod_{j=1}^{M-1} (\rho_j)^L \}^{-1} \tag{5.9}$$

Service rate transitions have an equilibrium property in this escalator policy.

Lemma 5.1. *In the Escalator Policy, the transition rate between adjacent service rates are balanced (equilibrium).*

Proof. Consider transitions between the states i and $i-1$, where $i = kL, 0 < k \le M$, such a state transition will cause the service rate to make a transition described as $\mu_k \leftrightarrow \mu_{k-1}$. Using Eq. (5.3) and (5.6), it may be observed that,

$$\mu_k p_{kL} = \lambda p_{kL-1}$$

This indicates that the average rate of transitions $\mu_k \to \mu_{k-1}$ is equal to the transition rate $\mu_{k-1} \to \mu_k$. It follows that the infinitesimal generator matrix Θ of the modulated service process is symmetric.

Consider the state $i = kL-1$ and $i = (k+1)L-1$, $0 < k < M$, it has

$$\mu_k p_{kL} + \lambda p_{(k+1)L-1} = \lambda p_{kL-1} + \mu_{k+1} p_{(k+1)L}$$

Hence the average rate of transition from $\mu_k \to (\mu_{k-1} \cup \mu_{k+1})$ is equal to the transition rate $(\mu_{k-1} \cup \mu_{k+1}) \to \mu_k$. □

Consider the average rate of transitions from $\mu_{k-1} \to \mu_k$ is equal to

$$r_{k-1,k} = \lambda p_{kL-1}, \quad (0 < k \leq M)$$

Applying Eq. (5.7) and (5.9) recursively, we obtain

$$r_{k-1,k} = \lambda (\rho_0)^{L-1} \prod_{j=1}^{k-1} (\rho_j)^L \cdot p_0, \quad (0 < k \leq M).$$

Applying again Eq. (5.9), gives

$$\begin{aligned} & r_{k-1,k} \\ &= \frac{\lambda \prod_{j=1}^{k-1} (\rho_j)^L}{\frac{1-(\rho_0)^L}{(\rho_0)^{L-1}(1-\rho_0)} + \sum_{i=1}^{M-1} \frac{\rho_i(1-(\rho_i)^L)}{1-\rho_i} \prod_{j=1}^{i-1} (\rho_j)^L + \frac{\rho_M}{1-\rho_M} \prod_{j=1}^{M-1} (\rho_j)^L}, \quad (0 < k \leq M). \end{aligned} \tag{5.10}$$

According to Lemma (5.1), we can obtain the average rate of transitions between adjacent service rates as

$$R = 2 \sum_{k=1}^{M} r_{k-1,k} \tag{5.11}$$

The average number in the system can be calculated by applying Eq. (5.8) and (5.9) to be

$$\bar{N} = \sum_{i=0}^{\infty} i p_i \tag{5.12}$$

Service rate transitions can result in packet loss. When using a DCG clock, once the control module decides to adjust the service rate of the system, the PLL circuit is turned on. Before the loop lock occurs and the new service rate has been established, the clock output will be chaotic as the transition of this loop from its unlocked to locked state is a highly nonlinear procedure [107]. This may result in packet loss. Even though this lock time is not an issue in DCS, higher frequency transition rates consume more power as shown in Table. 5.2, which shows empirical results captured from our NetFPGA-1G testbed. The experiment has been designed as 50% of time at 125MHz clock rate and 50% of time at 62.5MHz clock rate without any incoming traffic load. Thus, excessively frequent rate transitions can adversely affect power consumption. For these reasons, unnecessary transitions should be avoided, by introducing some form of damping to the algorithm. An alternative approach, called **Hysteresis Policy**, addresses this issue.

Table 5.2 Power Consumption caused by Clock Rate Transition

Transition Rate (times/second)	Power Consumption (Watt)
10	6.4900
100	6.5300
1000	6.6000

5.2.3 Hysteresis Policy

One way to introduce such damping is to use hysteresis. The thresholds at which transitions occur depend on the dynamics of traffic. Specifically, the queue length thresholds at which transition occurs depend on whether the queue is falling or emptying. Additional state information is not required since the queue length and the service rate are related

monotonically, and the latter data is already being recorded. The Hysteresis Policy can be described as follows:

- Following a packet arrival, if the queue length reaches the threshold kL $(where\ 0 < k \leq M)$, and the current service rate is μ_{k-1}, and thus the corresponding queue length was previously lower, the service rate is increased to μ_k,
- Following a packet departure, if the queue length reaches the threshold $(k-1)L$ $(where\ 0 < k \leq M)$, and the current service rate is μ_k, and thus the corresponding queue length was previously higher, the service rate is decreased to μ_{k-1},
- If the queue length exceeds the threshold ML, and the service rate is the maximum value μ_M, then the service rate will maintain at μ_M until the queue length falls below $(M-1)L$.

The initial service rate μ_0 can be set to 0 to fully utilize the rates and achieve the optimized power saving. The system state transition process when the system is in a stable state is presented in Fig. 5.2.

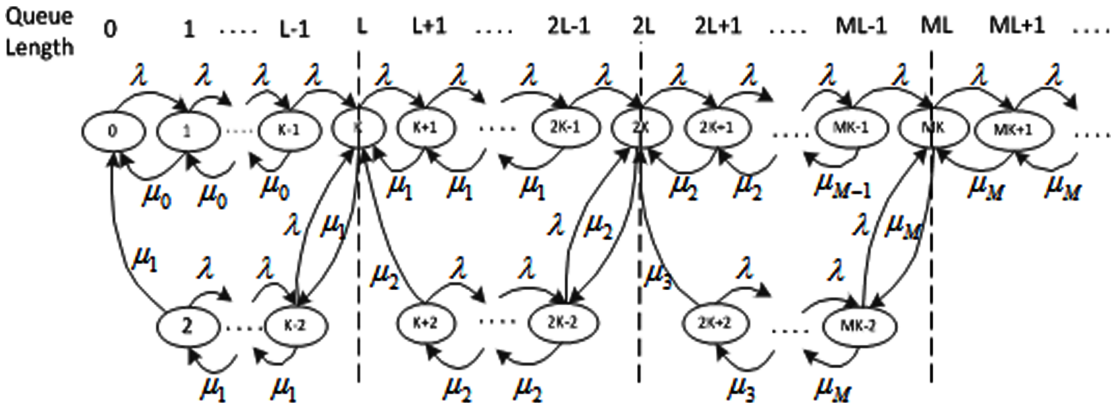


Fig. 5.2 State Transition-Rate Diagram (Hysteresis Policy)

The fundamental idea of the hysteresis control is to provide a dead zone where momentary fluctuations in queue length do not trigger rate transitions which results in the reduction

of rate oscillation and thereby reduce power consumption. There are three independent Birth-Death processes involved in this enhanced Hysteresis Policy control, which are as:

(a) the sub-process $\{X(t) = i | (k-1)L+1 \leq i \leq kL, 0 < k \leq M\}$ is a Birth-Death process with a constant service rate μ_k as load falls,

(b) the sub-process $\{X(t) = i | kL \leq i < (k+1)L-1, 0 < k < M\}$ is a Birth-Death process with a constant service rate μ_k as load increases,

(c) the sub-process $\{X(t) = i | i > ML\}$ is a Birth-Death process with a constant service rate μ_M.

Analogously with Eq. (5.5), for sub-process (a), we have an equilibrium solution

$$\begin{aligned} \mu_k p_{i+1}^{\mu_k} &= \lambda p_i^{\mu_k} + \mu_k p_{(k-1)L+1}^{\mu_k} \\ for\ (k-1)L+1 \leq i < kL, &\quad where\ 0 < k \leq M \end{aligned} \tag{5.13}$$

For sub-process (b), we have

$$\begin{aligned} \mu_k p_{i+1}^{\mu_k} &= \lambda p_i^{\mu_k} - \lambda p_{(k+1)L-1}^{\mu_k} \\ for\ kL \leq i < (k+1)L-1, &\quad where\ 0 < k < M \end{aligned} \tag{5.14}$$

The equilibrium equation for sub-process (c) is simply

$$\mu_M p_{i+1}^{\mu_M} = \lambda p_i^{\mu_M} \tag{5.15}$$

After simplification, we have

$$\begin{cases} p_{i+1}^{\mu_k} = \rho_k p_i^{\mu_k} + p_{(k-1)L+1}^{\mu_k}, \\ \qquad ((k-1)L+1 \le i < kL, \quad 0 < k \le M) \\ p_{i+1}^{\mu_k} = \rho_k p_i^{\mu_k} + (\rho_k)^L p_{(k-1)L+1}^{\mu_k}, \\ \qquad (kL \le i < (k+1)L-1, \quad 0 < k < M), \\ p_{i+1}^{\mu_M} = \rho_M p_i^{\mu_M}, \quad (i > ML) \\ p_{(k-1)L+1}^{\mu_k} = \rho_k \prod_{j=1}^{k-1} (\rho_j)^L \cdot p_0^{\mu_0}, \quad (0 < k \le M) \end{cases} \tag{5.16}$$

where, if $k = 1$, $\prod_{j=1}^{0} (\rho_j)^L = 1$.

Conditioning the entire state space (S) on the value of service rate (μ_k), according to the law of total probability and Eq. (5.16), it follows that

$$p_S^{\mu_k} = \begin{cases} \dfrac{L(1-(\rho_k)^L\rho_k)}{1-\rho_k} \prod_{j=1}^{k-1} (\rho_j)^L \cdot p_0^{\mu_0}, & (0 \le k < M) \\ \dfrac{L\rho_k}{1-\rho_k} \prod_{j=1}^{k-1} (\rho_j)^L \cdot p_0^{\mu_0}, & (k = M) \end{cases}$$

Calculating $p_0^{\mu_0}$ using the normalization equation $(\sum_{k=0}^{M} p_S^{\mu_k} = 1)$, we obtain

$$p_0^{\mu_0} = \{L \sum_{k=0}^{M-1} \frac{\rho_k(1-\rho_k^L)}{1-\rho_k} \prod_{j=1}^{k-1} (\rho_j)^L + L\frac{\rho_M}{1-\rho_M} \prod_{j=1}^{M-1} (\rho_j)^L\}^{-1} \tag{5.17}$$

Similarly, transitions of service rate in Hysteresis Policy also have an equilibrium property.

Lemma 5.2. *Using the Hysteresis Policy, the transition rates between adjacent service rates is in balance.*

Proof. as per Lemma 5.1. □

The average rate of transitions from $\mu_k \rightarrow \mu_{k-1}$ is given by

$$\begin{aligned} r_{k-1,k} &= \mu_k p^{\mu_k}_{(k-1)L+1} \\ &= \frac{\lambda \prod_{j=1}^{k-1} (\rho_j)^L}{L\left(\sum_{i=0}^{M-1} \frac{\rho_i(1-(\rho_i)^L)}{1-\rho_i} \prod_{j=1}^{i-1} (\rho_j)^L + \frac{\rho_M}{1-\rho_M} \prod_{j=1}^{M-1} (\rho_j)^L\right)} \end{aligned} \tag{5.18}$$

The average rate of transitions can be obtained by applying Eq. (5.11). In addition to above, the average service rate can be calculated as

$$\bar{\mu} = \sum_{k=0}^{M} \mu_k p_S^{\mu_k}$$

5.2.4 Algorithmic Implementation

The algorithms to implement these policies are listed as algorithm 5.1 and 5.2. The feasibility and usability of hardware implementation are an advantage from other control policies. Algorithm 5.2, for example, can be implemented by a combinational logic circuit including 3 comparators, 1 adder and several logical gates as presented in Fig. 5.3.

5.3 Performance Evaluation of Control Policies

5.3.1 Numerical Studies of Transition rate

The characteristics of the two proposed control policies have been investigated by detailed numerical analysis using Maple. Eqs (5.10) and (5.18) were processed by Maple to conduct this analysis. Fig. 5.4 shows the multi-factorial relationship between the threshold step size (L), arrival rate (λ) and the transition rate (R) for both the Escalator and the Hysteresis control policies respectively. The numerical analysis is of a frequency scaled

Algorithm 5.1: Escalator Policy

Input: L: the length of threshold, $L > 0$
M: the number of thresholds, $M > 0$
μ: a set of multiple service rates $\{\mu_0, \ldots, \mu_M\}$
$l(t)$: the current queue length
Output: $\mu(t)$: the current service rate

```
1  Function generate_current_service_rate        ▷ executed when a packet arrives
2      if l(t) ≥ ML then
3          μ(t) ← μ_M;
4          return;
5      end
6      forall index i in (0 : M − 1) do
7          if iL ≤ l(t) < (i+1)L then
8              μ(t) ← μ_i;
9              return;
10         end
11     end
12 end
```

system implementing five service rates, chosen to match those generated in the NetFPGA testbed, which have been listed in Table 4.2.

It is clear from Fig 5.4 that the transition rate is relatively insensitive to packet arrival rate. This is particular to the Hysteresis policy, showing that it is successful in reducing the frequency of transitions. As expected, the transition rate is much higher from smaller step stages, as shown in Fig 5.5. The effect of this on energy efficiency will be investigated in the next subsection.

As shown in Fig. 5.6, for both control polices. For the same threshold step size, the Hysteresis Policy is more resilient to variations of arrival rate.

Although the threshold mechanism would most accurately track system load if the buffer occupancies were recorded by bytes, this would cause implementation difficulties due to the size of registers and arithmetic logic required. Instead, the unit of queuing length used is the packet. This metric tracks the amount of header processing required, which is justified by the proportion of overall energy consumption it causes.

Algorithm 5.2: Hysteresis Policy

Input: L: the length of threshold, $L > 0$
M: the number of thresholds, $M > 0$
μ: a set of multiple service rates $\{\mu_0, \ldots, \mu_M\}$
$l(t)$: the current queue length
$\mu(t-1)$: the last service rate
Output: $\mu(t)$: the current service rate

```
1  Function generate_current_service_rate ▷ executed when a packet arrives/departs
2      forall index i in (0 : M) do
3          if l(t) = iL then
4              μ(t) ← μ_i;
5              return;
6          end
7      end
8      μ(t) ← μ(t − 1);
9      return;
10 end
```

5.3.2 Queuing Performance

Although the numerical models presented in the previous subsection allow various insights to the gained about the operation of the frequency scaled switch, they are limited in that only a simple arrival process can be modelled and the model is limited to an assumed infinite buffer capacity so extending the numerical model is not feasible thus further evaluation of the system requires computer simulation.

The simulation developed supports both the Escalator and the Hysteresis policies. It is based on Java Modelling Tools (JMT)[1] [108] which consists of a suite of GPL licensed applications for performance analysis and workload characterization studies. We integrate our proposed control policy algorithms with the JMT framework and add a new dedicated measurement functionality for investigating the impacts of controlling transition rate.

The key parameter values used are enumerated in Table 5.3 and the simulation outputs are listed in Table 5.4.

[1] Version 1.0.1-Beta 2, released 2017-May-28.

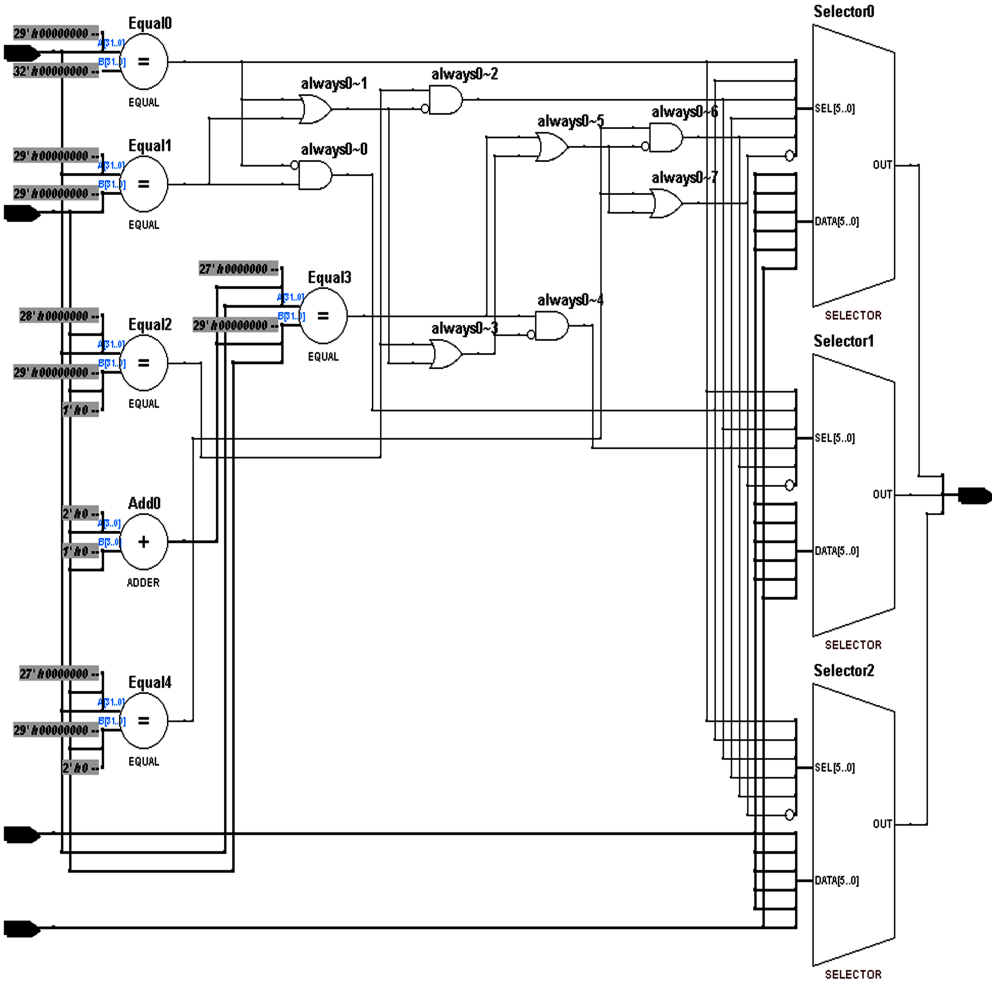
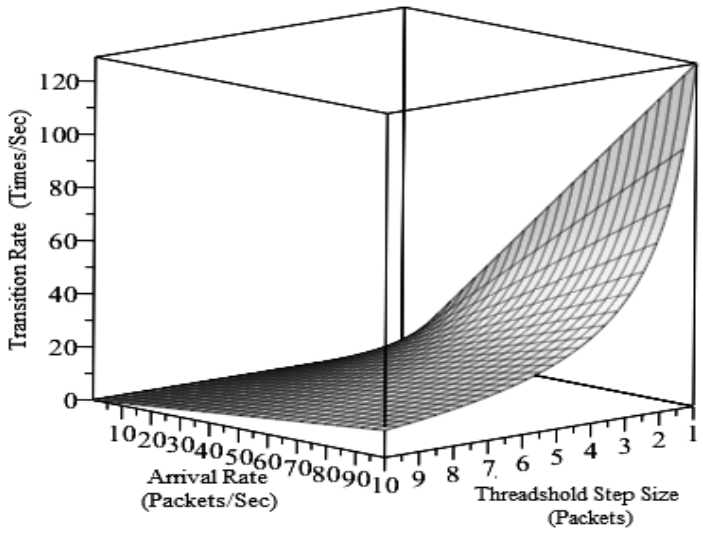


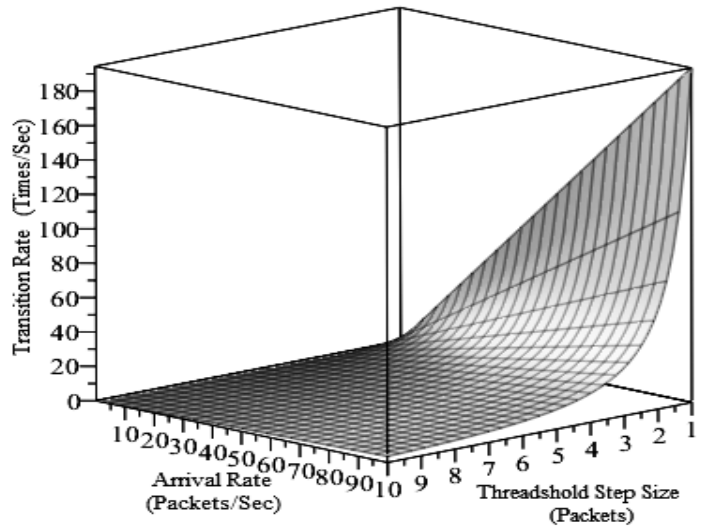
Fig. 5.3 Implementation of Hysteresis Policy in Hardware

The simulation outputs are processed using transient detection control and confidence interval estimation. Non-stationary data which is collected during the initial transient simulation period is detected by using R5 heuristic control [109] and discarded before calculating statistics. The confidence intervals of the results are computed by using the spectral analysis method presented in [110]. The process flowchart is shown in Fig. 5.7.

The simulation results match those obtained by the numerical studies closely. Fig. 5.8 compares the transition rate and average queue length obtained by these two approaches using the Hysteresis Policy values. Fig. 5.8a depicts the variations of the transition rate vs. different arrival rates varying from 0.5 to 10, and 5 traffic intensities are evenly dispersed



(a) Escalator Policy



(b) Hysteresis Policy

Fig. 5.4 Transition rates for both Policies

along the range of 0 to 1. Fig. 5.8b demonstrates the average queue length with the same conditions.

Fig. 5.9 illustrates the queue length distributions of obtained by simulation for an arrival rate of 10 packets/second. It can be seen that the larger threshold step size (10 rather than 5) in general results in higher queue length. Note that in the case of the Hysteresis Policy, the median queue length is equal to the step size. Thus the Hysteresis Policy would appear to

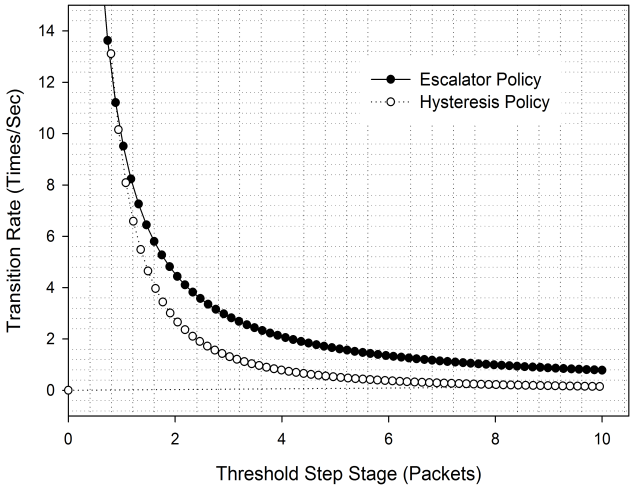


Fig. 5.5 Transition Rate vs. Threshold Step Size

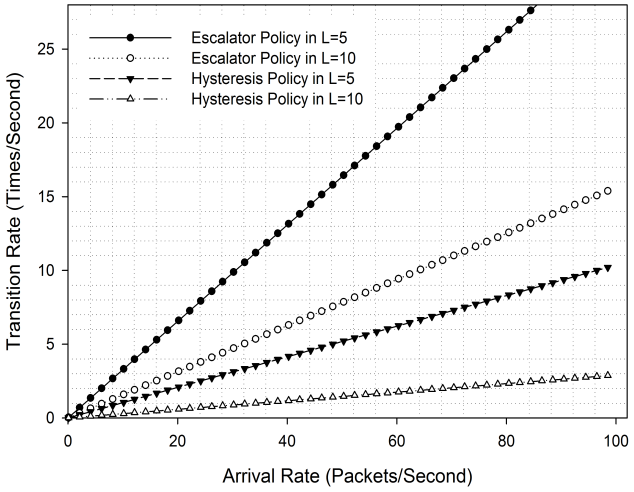


Fig. 5.6 Transition Rate vs. Threshold Step Size

offer worse delay properties. In fact, this is not the case, as will be discussed shortly. Also, its energy consumption is less, as shown by the empirical results collected form the testbed.

The Hysteresis Policy has worse queuing performance (i.e. delay, average queue) but better energy efficiency than the Escalator Policy. For both policies, increasing the threshold step size improves energy efficiency at the expense of QoS. The question then arises, which policy to use, and how to select *L*. Looking more deeply at the system QoS can inform this decision.

Table 5.3 Key Simulation Parameters

Parameters	Value
Max number of samples	1000000
Max simulation time	infinite
Confidence Level	0.99
Max Relative Error	0.03
Source process	exponential inter-arrival time distribution with mean varying in$[0.5, 10]$
Queue size	infinite
Service process	exponential service rate distribution with mean varying in $[0, 30]$
Number of threshold steps (M)	5
Traffic intensities	{0.18,0.36,0.54,0.72,0.9}
Threshold step size	varying in $[1, 10]$

Table 5.4 Simulation Outputs

Simulation Outputs
Average Transition Rate
Average Queue Length
Average Delay
Average Throughput
Average Drop Rate

Fig. 5.10 presents the relationship between the threshold step size (L), arrival rate and power consumption when using the Escalator and the Hysteresis control policies by using the data acquired from our frequency scaled NetFPGA-1G platform.

To investigate the QoS achieved using our proposed frequency scaling control policies, the impact of a finite queue size was considered. These simulations were configured with various arrival rates ranging between 1 to 20 packets/second. We assume that the system provides 5 different service rates ($M = 5$) which are 1, 2, 4, 8 and 16 packets/second. The threshold step size (L) is defined as $\{BufferCapacity/5\}$. For both the Escalator Policy and the Hysteresis Policy, we have recorded queue occupancy, delay, throughput and drop rate and compared the results with those for a legacy system which has a constant service rate

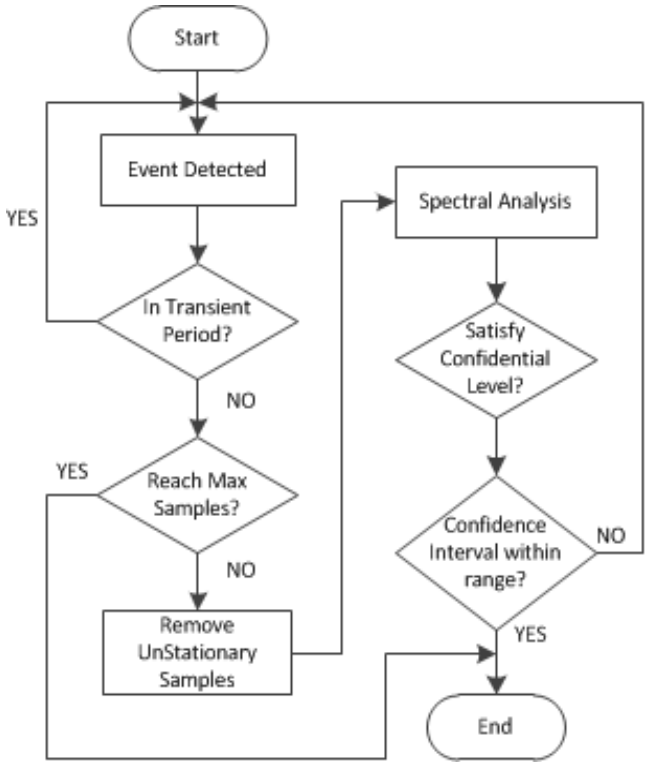
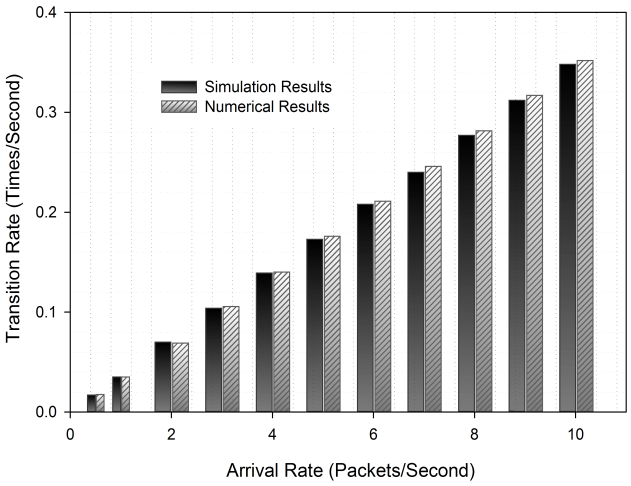


Fig. 5.7 Simulation Process Flowchart

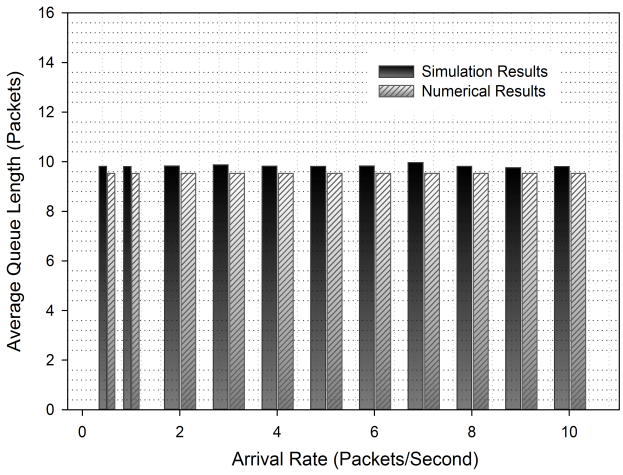
equivalent to the frequency scaled switch running at its highest speed. The results are shown in Figs. 5.11 - 5.14.

From the combination of numerical, simulation, and empirical data collected in this chapter, the following conclusions may be drawn:

- As expected from traditional queuing systems, average queue occupancy increases as load increases.
- Counter-intuitively, delay falls as load increases. This is because the frequency scaling increases the system throughput under high load.
- The average delay of the Hysteresis Policy is lower than that of Escalator Policy. This is because the Hysteresis Policy is more likely to use higher service rate.



(a) Average Transition Rate



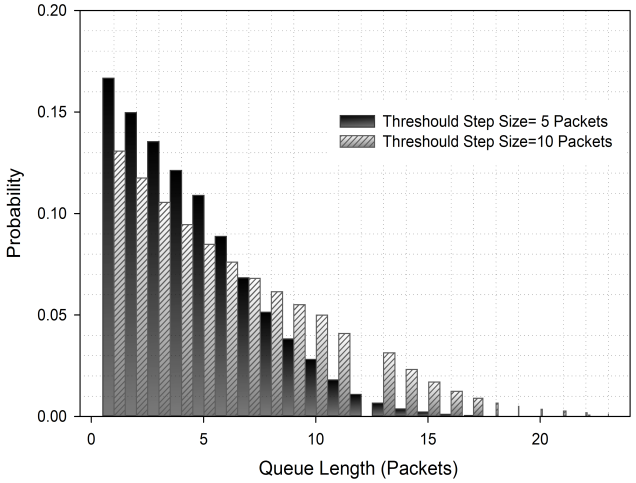
(b) Average Queue Length

Fig. 5.8 Comparison of Simulation and Numerical Results

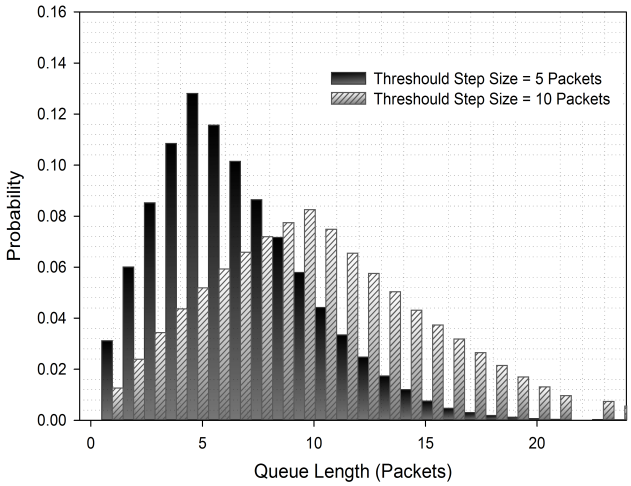
- The drop rate is higher when frequency scaling is used than when the system operates at a constant clock rate. This is a major performance drawback of using frequency scaling.

5.4 Summary

Energy efficient techniques such as frequency scaling, voltage gating etc. have matured and their effectiveness has been widely recognized. Policies for adapting these green tech-



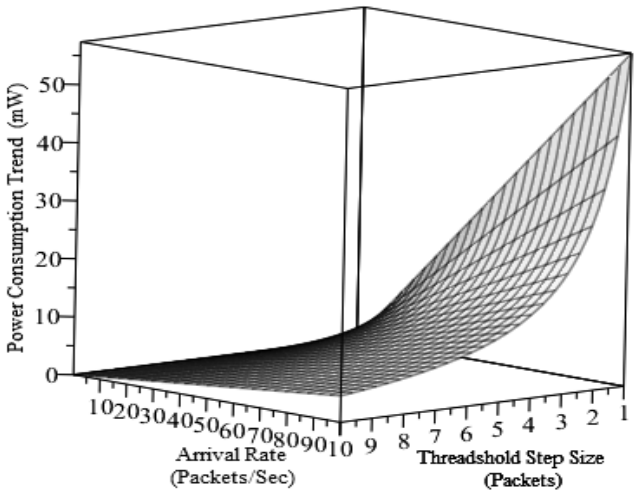
(a) Escalator Policy



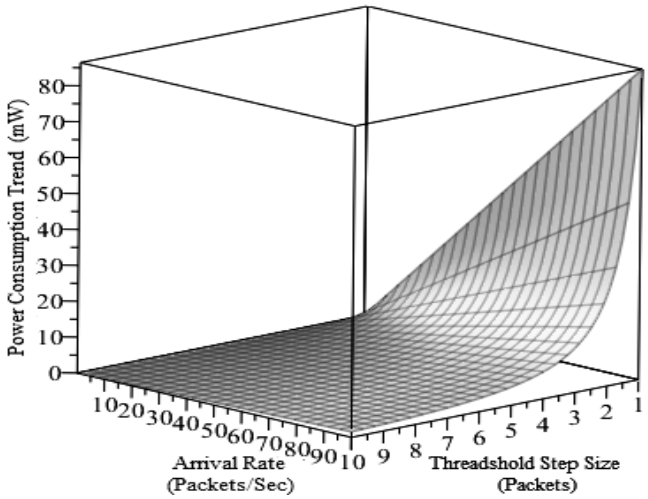
(b) Hysteresis Policy

Fig. 5.9 Queue Length Distribution for Escalator and Hysteresis Control Policies

niques in real time have received less attention. This chapter contains a thorough analysis of two novel control policies for handling packet queue in frequency scaled systems and the algorithms for implementing these policies have been presented. The performance impacts of using frequency scaling have been investigated using numerical analysis and simulation. The results show that, for example, mean delay paradoxically falls with increasing load in systems using frequency scaling, as the increase in load is compensated for by an increase in system capacity.

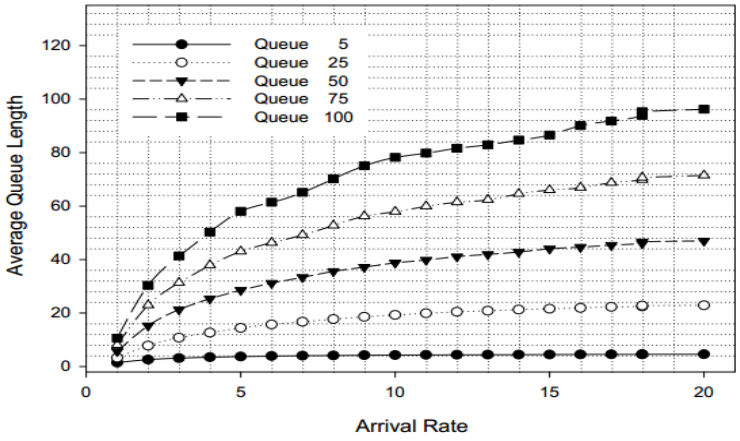


(a) Escalator Policy

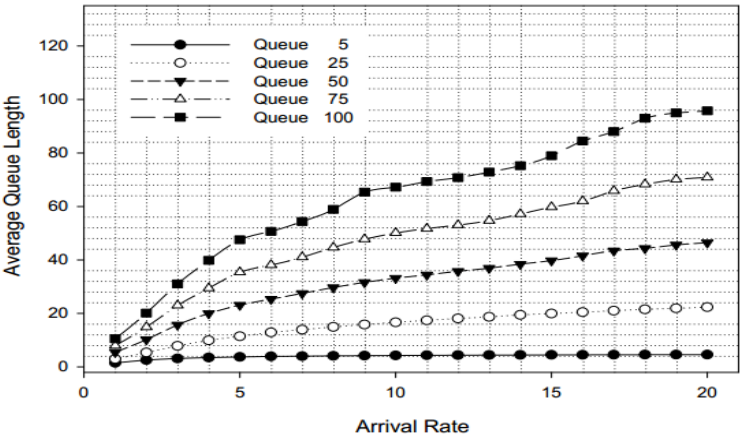


(b) Hysteresis Policy

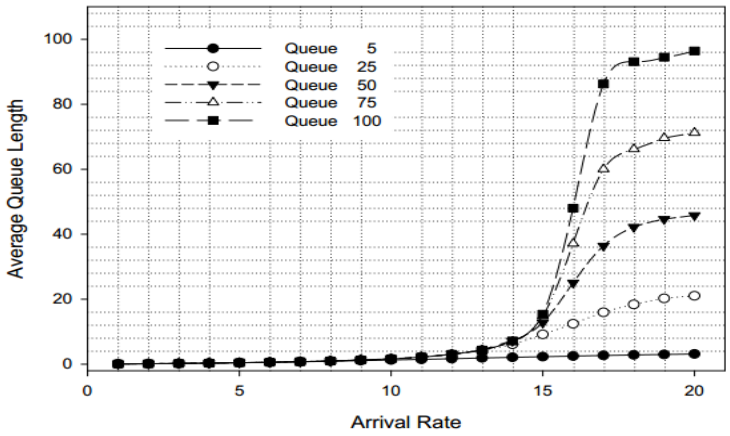
Fig. 5.10 Power Consumption Characteristics using Control Policies



(a) Escalator Policy

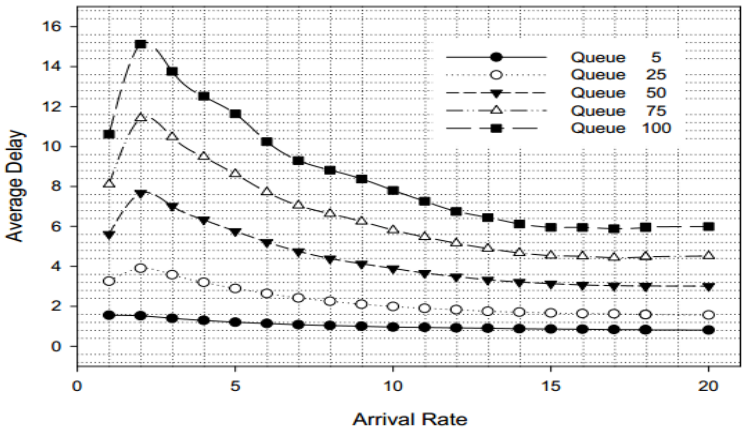


(b) Hysteresis Policy

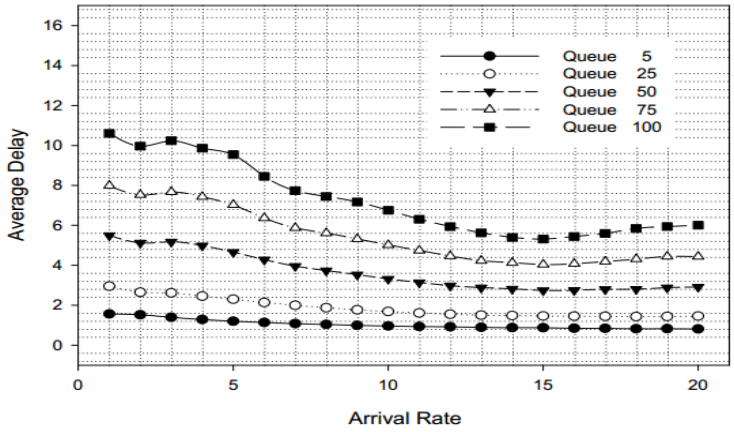


(c) Single Service Policy

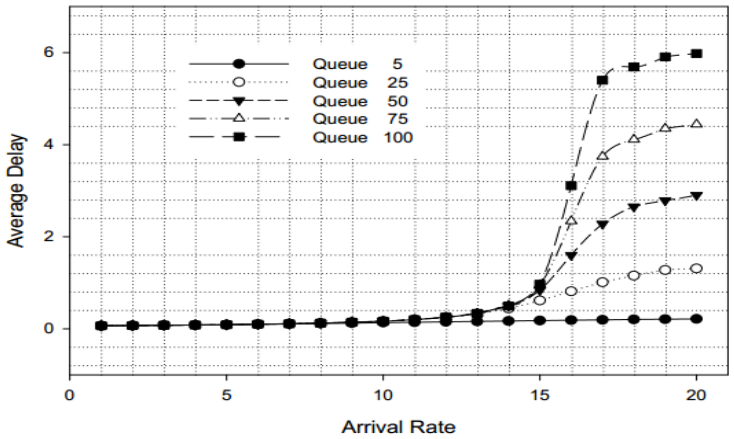
Fig. 5.11 Evaluations: Average Queue Length of the Systems



(a) Escalator Policy

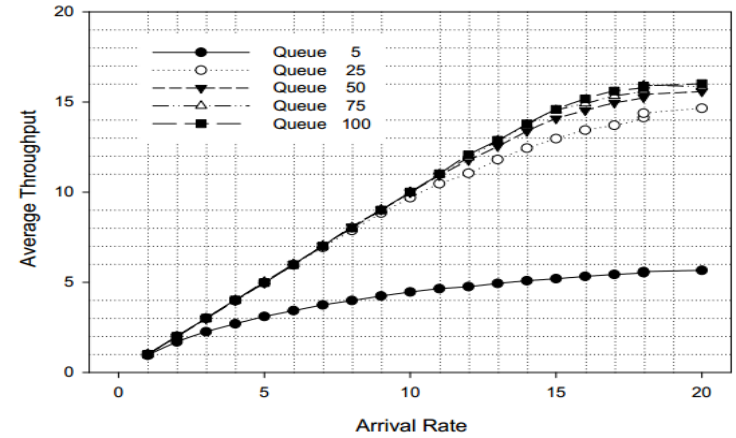


(b) Hysteresis Policy

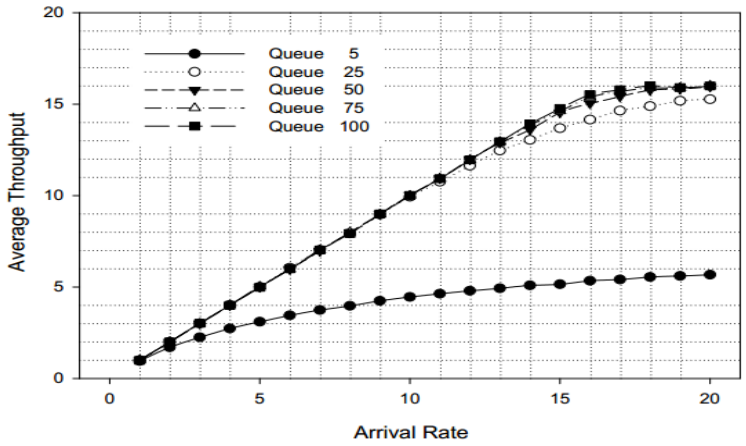


(c) Single Service Policy

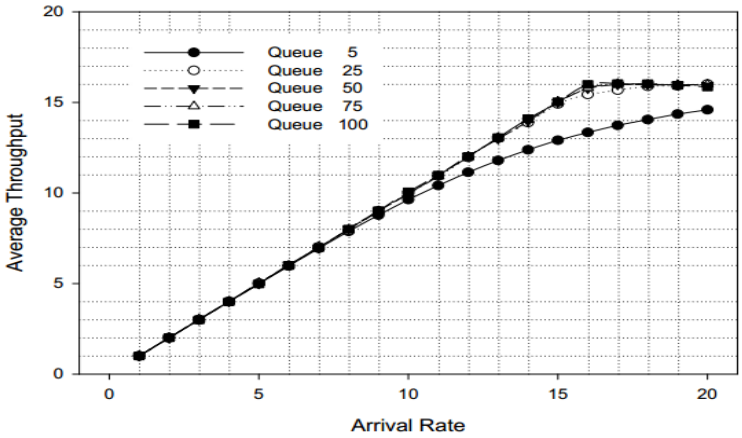
Fig. 5.12 Evaluations: Average Delay of the Systems



(a) Escalator Policy

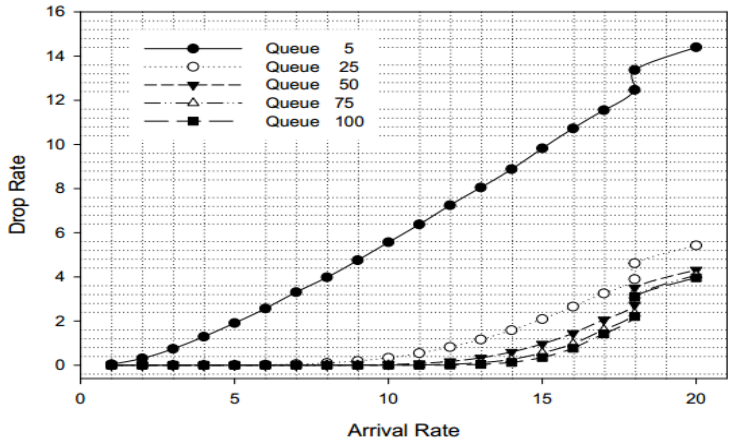


(b) Hysteresis Policy

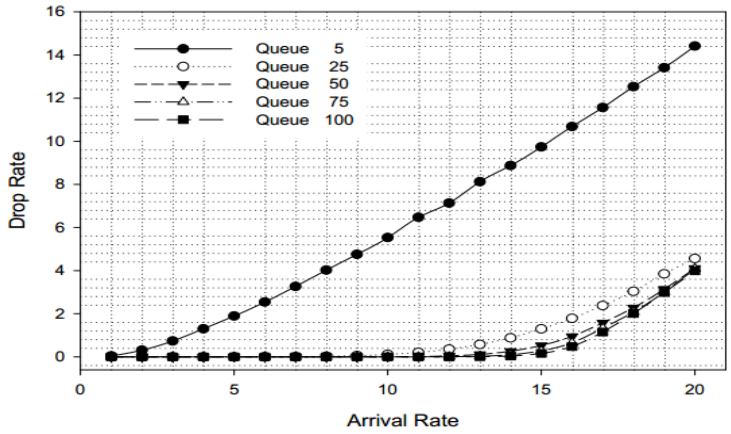


(c) Single Service Policy

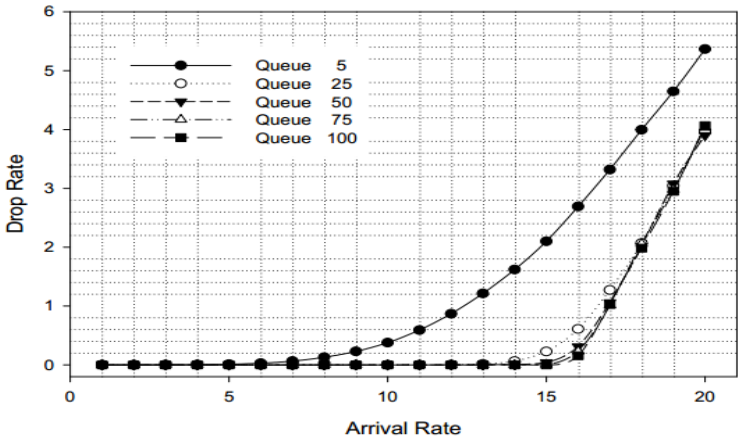
Fig. 5.13 Evaluations: Average Throughput of the Systems



(a) Escalator Policy



(b) Hysteresis Policy



(c) Single Service Policy

Fig. 5.14 Evaluations: Drop Rate of the Systems

Chapter 6

Energy-aware Resource Allocation at Network Edge

This chapter contains a proposal for energy-efficent resource allocation at edge networks. In this work, the programmable data plane NDs dynamically collaborate with the control plane to meet performance, flexibility and energy related challenges. The proposed Online Energy-efficient Resource Allocation (OERA) is based on an online approach using Integer Programming [26] with the objective of minimizing the edge network energy consumption.

6.1 Resource Allocation Model

The power-aware resource allocation model in this chapter is based on the edge network embodied with nodes, including networking (switches/routers) and processing (computing/storage) devices, which support frequency scaling.

1. Substrate Edge Network: The substrate edge network is modelled as a undirected connected graph G with specific weights, denoted by $G^S = (N^S, E^S, A_N^S, A_E^S)$, where:

- $N^S = (S^S, C^S)$ represents a set of substrate nodes that includes a network node set (S^S) and a processing (computing/storage) node set (C^S)[1];
- E^S denotes a set of substrate links;
- A_N^S and A_E^S describe the attribute sets of the substrate nodes and links (for example; clock frequency, CPU capacity and location are attributes of a node);
- $P^S(s,d)$ denotes the set of all loop-free paths in the substrate network from a source node s to a destination node d, $\forall s,d \in N^S$ and $s \neq d$.

In this study, we use the NDPM (as validated in Section 4.5) to represent each such node.

2. Virtual Network Request: The VNR is an undirected connected graph denoted as $G^V = (N^V, E^V, C_N^V, C_E^V)$, where:
   - N^V is the set of requested nodes;
   - E^V is the set of requested links;
   - C_N^V and C_E^V describe the constraints on the requested nodes and requested links;
   - P^v denotes the set of all loop-free paths in the virtual network request graph.
3. Virtual Network Request Mapping: The mechanism of mapping virtual network requests to a subset of the substrate network, is modelled as $\mathscr{M} : G^V \mapsto (N', P', R_N, R_E)$, where:
   - $N' \in N^s$;
   - $P' \in P^s$;
   - R_N and R_E represent the resources allocated to virtual nodes and virtual links respectively.

[1] $N^S = (S^S, 0)$ denotes a network switching node only.

In general, the virtual network mapping can be divided into two stages: node mapping and link mapping.

(a) Node Mapping: A virtual node mapping to a substrate node with constraints is defined as $\mathscr{M}^N : (N^V, C_N^V) \mapsto (N', R_N)$.

(b) Link Mapping: A virtual edge mapping to a substrate link/path with constraints is defined as $\mathscr{M}^E : (E^V, C_E^V) \mapsto (P', R_E)$.

The significant distinction between the new algorithm and others is that it can control substrate nodes running in their low power state, thus achieving the goal of minimizing overall power consumption of the entire edge network.

6.2 Online Energy-aware Resource Allocation Solution

6.2.1 Augmented Substrate Graph Construction

To maximise the benefit of edge networking, the processing units, including storage mediums, as much closer as where the user demands it to be, we associate each requested virtual node ($n^V \in N^V$) with a constrained radius (Rad^V). We denote a cluster of requested virtual nodes by $\Omega(n^V)$, which consists of the set of substrate nodes satisfying the geometrical positioning requirements and the conditions for each request:

$$\Omega(n^V) = \{n^s \in N^S | \ \|n^v, n^s\| \leq Rad^V\} \tag{6.1}$$

where $\|n^v, n^s\|$ is the distance between a request virtual node n^v and a substrate note n^s.
At a higher level of abstraction of the network, the set of nodes comprising $\Omega(n^V)$ will be regarded as a single *meta-node*. This meta-node is denoted as $\mu(n^V)$. Then, we connect $\mu(n^V)$ with all the substrate nodes in $\Omega(n^V)$ using infinite bandwidth *meta-edges* to complete the final augmented substrate graph which comprises:

- all the nodes and edges of the original substrate graph N^S;
- a set of meta-nodes, are for each virtual node in the request;
- a set of meta-edges, connecting each meta-node to the substrate nodes which are sufficiently chose to its ideal location, as denoted by Eq. 6.1.

Hence, the augmented substrate graph is denoted as $G^{s'} = (N^{S'}, E^{S'})$, where

$$N^{S'} = \{\mu(n^V) | n^V \in N^V\} \cup N^S$$

$$E^{S'} = \{(\mu(n^V), n^S) | n^V \in N^V, n^S \in \Omega(n^V)\} \cup E^S$$

6.2.2 Integer Programming for Energy-aware Resource Allocation

Once the augmented graph is established, the resource allocation problem can be modelled as a formal Multi-Commodity Flow Problem (MFP) [111]. Each flow starts from one meta-node and terminates at another meta-node. The notation used in the OERA is described in Table 6.1. The system model can be expressed using Integer Programming as follows:

Model Output Variables:

- $freq_n$: The clock frequency of a substrate node n.
- f_{uv}^i: The total amount of flow from u to v for the $i'th$ virtual edge.
- x_{mn}: A binary variable, assigned the value "1" when a virtual node m is allocated to the substrate node n.

Table 6.1 OERA Symbols and Their Descriptions

Symbols	Interpretation
N^S	Set of substrate nodes
N^V	Set of requested nodes
S^S	Network node
E^V	Set of requested edges
$(C^S)^2$	Processing node with computing and storage
A_N^S	Attribute set of substrate nodes
A_E^S	Attribute set of edges in substrate network
$P^S(s,d)$	Set of loop-free paths from source s to destination d
P^V	Set of virtual network loop-free paths
M^N	Node Mapping of VNR to a subset of substrate node
M^E	Edge Mapping of VNR to a subset of substrate link/path
$\Omega(n^V)$	Cluster of requested virtual node
$\mu(n^V)$	n^V representation as meta-node
$\eta(n)$	Weight factor of network node n
$\nu(n)$	Weight factor of computing node n
$R_{freq}(n)$	Residual frequency resource of node n
$R_c(n)$	Residual processing resource of node n
$\Lambda(u,v)$	Throughput from node u to v
$\Xi(n)$	Allocated CPU capacity of node n
$c(n)$	Computation capacity of node n
Bus_u	Bus width of node u
$Port_u$	No. of ports in node u
$freq_u$	Clock frequency of node u
f_{uv}^i	Flows from u to v for $i'th$ virtual edge
$\{s_i, d_i\}$	Source and Sink nodes of link i

Problem Formulation: We formulate the OERA problem for resource allocation:

$$Problem : \min \quad Power \tag{6.2}$$

subject to:

Frequency constraints(6.6)-(6.7)

Capacity constraint (6.8)-(6.9)

Flow constraints (6.10)-(6.14)

Meta and Binary constraints (6.15)-(6.17)

Domain constraints (6.18)-(6.20) where,

$$Power = \sum_{n \in N^S} \eta(n) P_{network}(n) + \sum_{n \in N^S} \nu(n) P_{comp}(n)$$

and the weight factors η and ν are set as

$$\eta(n) = \frac{\alpha_n}{R_{freq}(n) + \delta}, \quad \nu(n) = \frac{\beta_n}{R_c(n) + \delta}$$

where $R_{freq}(n)$ is the residual frequency resource of the network node and $R_c(n)$ is the residual processing resource of the computation node n. α_n and β_n are the variables to control load balancing and are limited to $1 \leq \alpha_n \leq R_{freq}(n)$ and $1 \leq \beta_n \leq R_c(n)$ respectively. The residual frequency resource is the difference between the resourceś maximum clock frequency and the currently assigned operation frequency. While the residual processing resource is the proportion of the processing capacity of a processing node which is currently unallocated, weighted by its peak processing capacity. We investigate the selection of these parameters later in this Chapter. δ is a small positive constant to avoid the case where the denominator is equal to zero. $P_{network}(n)$ describes the power consumption on the substrate node $n \in N^S(S^S, 0)$ and is obtained from the models presented in Chapter 4, with parameters obtained using the methods of Section 4.5. The term $P_{comp}(n)$ is used to model the relationship between workload and power consumption. For simplicity, here we use the linear model presented in [112] as follows,

$$P(u) = P_{idle} + (P_{busy} - P_{idle}) \cdot u \tag{6.3}$$

where u is the current proportion of the processing resource in use, i.e.,

$$u = 1 - \frac{R_c(n)}{R_c^{max}(n)} \tag{6.4}$$

Hence, the objective equation is rewritten in Eq. (6.5)

Minimize:

$$\begin{aligned}\sum_{v\in N^S}\frac{\alpha_v}{R_{freq}(v)+\delta}P^{(v)}_{network}(freq_v,\sum_{u\in N^{S'}}(\Lambda(u,v)+\sum_i f^i_{uv})\\ +\sum_{n\in N^S}\frac{\beta_n}{R_c(n)+\delta}P^{(n)}_{comp}(\frac{\Xi(n)+\sum_{m\in(N^{S'}\setminus N^S)}x_{mn}c(m)}{c(n)})\end{aligned} \tag{6.5}$$

subject to:

Frequency Constraints:

$$b(e^S_{uv})=min\{\frac{Bus_u\cdot freq_u}{Port_u},\frac{Bus_v\cdot freq_v}{Port_v}\},\quad \forall u,v\in N^S \tag{6.6}$$

$$\sum_{v\in N^S}(\Lambda(u,v)+\sum_i f^i_{uv})\le Bus_u\cdot freq_u,\quad \forall u\in N^S \tag{6.7}$$

Capacity Constraints:

$$\sum_i(f^i_{uv}+f^i_{vu})\le R_E(u,v)\quad \forall u,v\in N^{S'} \tag{6.8}$$

$$R_c(n)\ge x_{mn}c(m),\quad \forall m\in(N^{S'}\setminus N^S),\forall n\in N^S \tag{6.9}$$

Flow Constraints:

$$\sum_{w\in N^{S'}}f^i_{uw}=\sum_{w\in N^{S'}}f^i_{wu},\quad \forall i,\forall u\in(N^{S'}\setminus\{s_i,d_i\}) \tag{6.10}$$

$$\sum_{w\in N^S}f^i_{s_i,w}-\sum_{w\in N^S}f^i_{w,s_i}=b(e^V_i),\quad \forall i \tag{6.11}$$

$$\sum_{w\in N^S}f^i_{d_i,w}-\sum_{w\in N^S}f^i_{w,d_i}=-b(e^V_i),\quad \forall i \tag{6.12}$$

$$f^i_{s_i,w}=b(e^V_i)\cdot x(s_i,w),\quad \forall i,\forall w\in N^S \tag{6.13}$$

$$f^i_{w,d_i}=b(e^V_i)\cdot x(d_i,w),\quad \forall i,\forall w\in N^S \tag{6.14}$$

Meta and Binary Constraints:

$$\sum_{w\in\Omega(m)}x_{mw}=1,\quad \forall m\in(N^{S'}\setminus N^S) \tag{6.15}$$

$$\sum_{m \in (N^{S'} \setminus N^{S})} x_{mw} \leq 1, \quad \forall w \in N^{S} \tag{6.16}$$

$$f_{uv}^{i} = 0, \quad \forall i, \forall u, v \in (N^{S'} \setminus N^{S}) \tag{6.17}$$

Domain Constraints:

$$freq_{n} \geq 0, \quad \forall n \in N^{S'} \tag{6.18}$$

$$f_{uv}^{i} \geq 0, \quad \forall u, v \in N^{S'} \tag{6.19}$$

$$x_{u,v} \in \{1, 0\}, \quad \forall u, v \in N^{S'} \tag{6.20}$$

Remarks:

- The frequency constraints in (6.6)-(6.7) define the bandwidth provision of a network node. All NDs forward packets use a round-robin strategy to offer a fair bandwidth utilization of each port as shown in Eq. (6.6).
- The set of capacity constraint in (6.8)-(6.9) defines the computation capacity that can be offered by the nodes. Eq. (6.9) uses a binary variable x_{mn} to ensure that the mapped server has enough capacity to fulfil the resource requests.
- The set of flow constraints in (6.10)-(6.14) defines the traffic flow characteristics. Eq. (6.10)-(6.12) ensures flow conservation except at the source and sink nodes $\{s_i, d_i\}$, which have the constraints defined in Eq. (6.13)-(6.14)
- The meta and binary constraints in (6.15)-(6.17) define the behavior and efficiency strategy for OERA. Eq. (6.15)-(6.16) ensure that the node mapping $\mathscr{M}$ will be a one-to-one mapping, so that only one substrate node will be selected for each requested virtual node. Eq. (6.17) ensures that no flow is allocated between virtual nodes [113].
- The domain constraints in (6.18)-(6.20) denote the variable constraints.

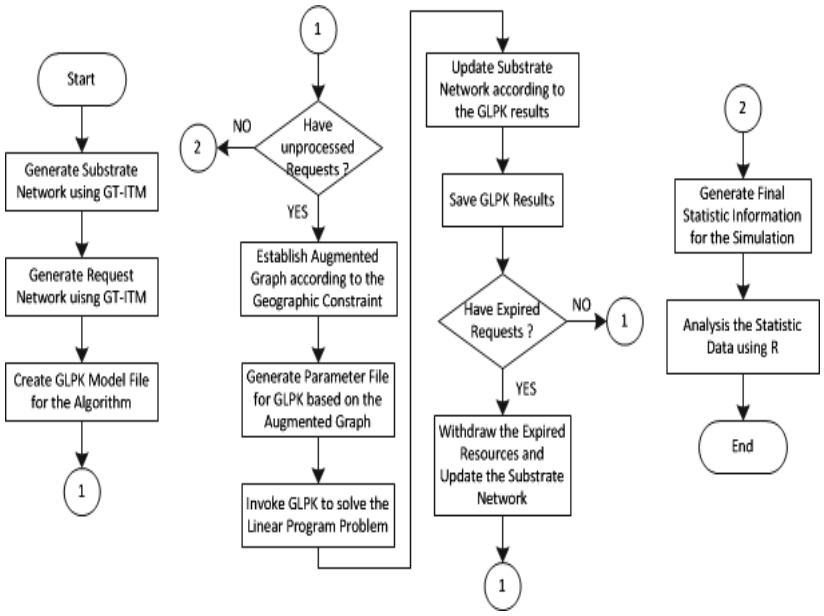


Fig. 6.1 The procedure of the operating simulation

6.2.3 OERA Simulation Environment

The OERA is implemented using C++ (Algorithm 6.1). The GNU Linear Programming Kit (GLPK) is used in the implementation. GT-ITM [114] is used to randomly generate a topology of edge network resources and virtual network requests for the simulation study. Fig. 6.1 describes the algorithm implementation and its simulation. The simulation parameters used are listed in Table 6.2.

6.3 OERA Performance Evaluation

For benchmarking, we compare OERA with two existing algorithms[2] namely *R-ViNE* and *D-ViNE* [27]. Both R-Vine and D-Vine are two-stage deterministic algorithms and minimize resource costs efficiently. However, OERA uses the NDPM and implements two levels of energy minimization (Device and Network). We compare OERA to [27] using three performance metrics,

[2]Based on the source code on http://www.mosharaf.com/ViNE-Yard.tar.gz

Algorithm 6.1 Online Energy-aware Resource Allocation Algorithm

Input: Substrate Graph $G^S = (N^s, E^s, A_N^S, A_E^S)$
Virtual Network Request $G^V = (N^v, E^V, C_N^V, C_E^V)$
Output: Frequency Set of Each ND $freq[0..n]$
Computation Hosts Mapping x_{mn}, Link Mapping f_{uv}

Procedure OERA ()
1. **for each** requested node n^V with location constrain ($dist(n^V)$) $\in G^V$, **do**
2. find the potential mapping cluster $\Omega(n^V)$
3. **if** ($\Omega(n^V) == \varnothing$) **then** mapping fail, reject G^V, **return**;
4. **else** create augmented substrate graph $G^{S'}$
5.
6. $freq[0..n]$, x_{mn}, $f_{uv} \leftarrow$ solve OREA Model using Eq. (6.5) – Eq. (6.20)
7. **if** Objective of OREA (Eq. (6.5)) is attained **then**
8. Update residual capacities of G^s;
9. **return** $freq[0..n]$, x_{mn}, f_{uv};
10. **else**
11. mapping fail, reject G^V, **return**;
12. **end procedure**

Table 6.2 Key Simulation Parameters

Parameters	Values
Weight Factors	
α_n	$R_{freq}(n)$
β_n	$R_c(n)$
Power Scaling Network Devices	
Coefficient of the throughput	uniformly distrb. in (0.05, 0.1)
Coefficient of the clock frequency	uniformly distrb. in (0.5, 1)
Base constant	uniformly distrb. in (10, 30)
Number of ports	node degree
Bus width	64
Substrate Network	
Number of substrate nodes	100
Computation and bandwidth capacity	uniformly distrb. in (50, 100)
Virtual Network Request	
Number of virtual nodes	uniformly distrb. in (2, 20)
Requested computation	uniformly distrb. in (0, 20)
Requested bandwidth	uniformly distrb. in (0, 50)
Inter-VNR time	exponentially distrb. with mean 25
Request valid time	exponentially distrb. with mean 250

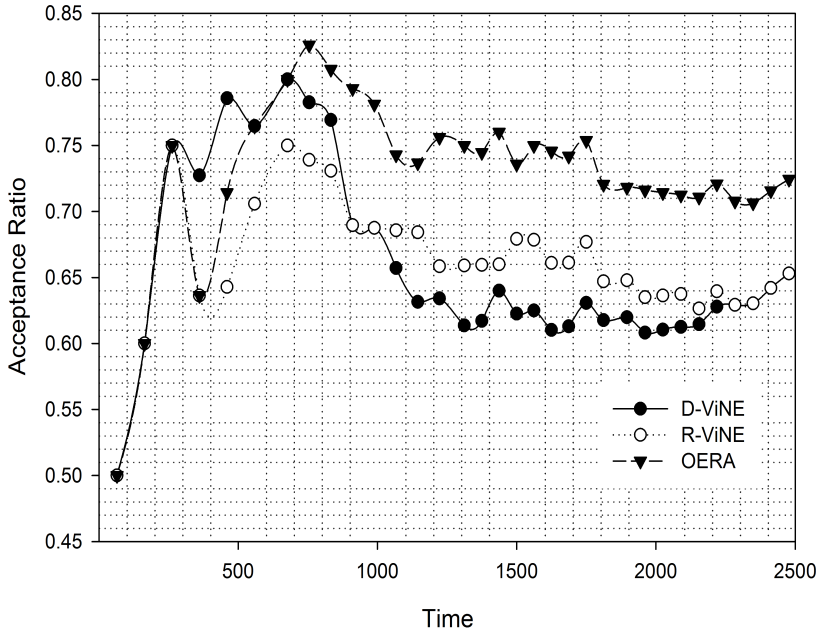


Fig. 6.2 Request Acceptance Ratio over Time

- The acceptance ratio is defined as the ratio of the number of successful requests to the total number of requests from the virtual network to the substrate network.

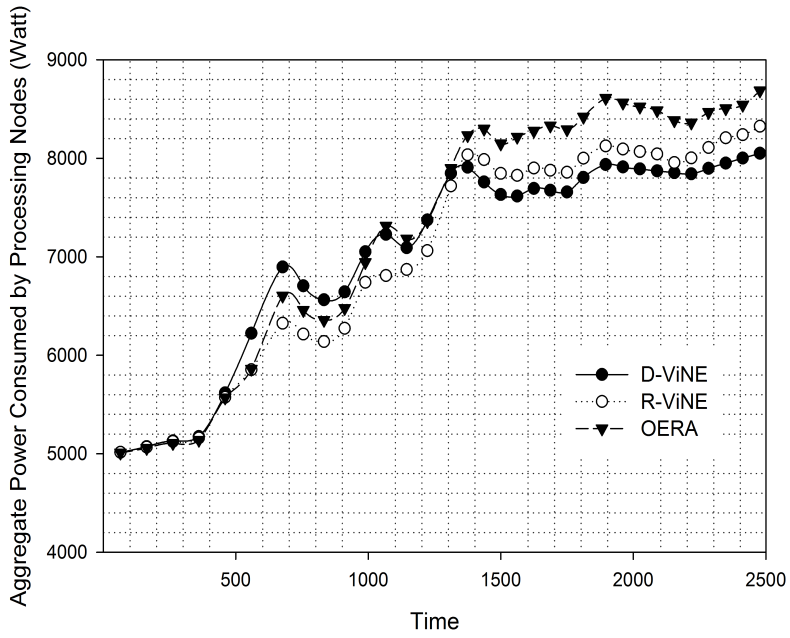
$$Acceptance\ Ratio = \frac{Number\ of\ accepted\ requests}{Number\ of\ total\ requests} \tag{6.21}$$

- The overall power dissipation of the edge network is calculated as

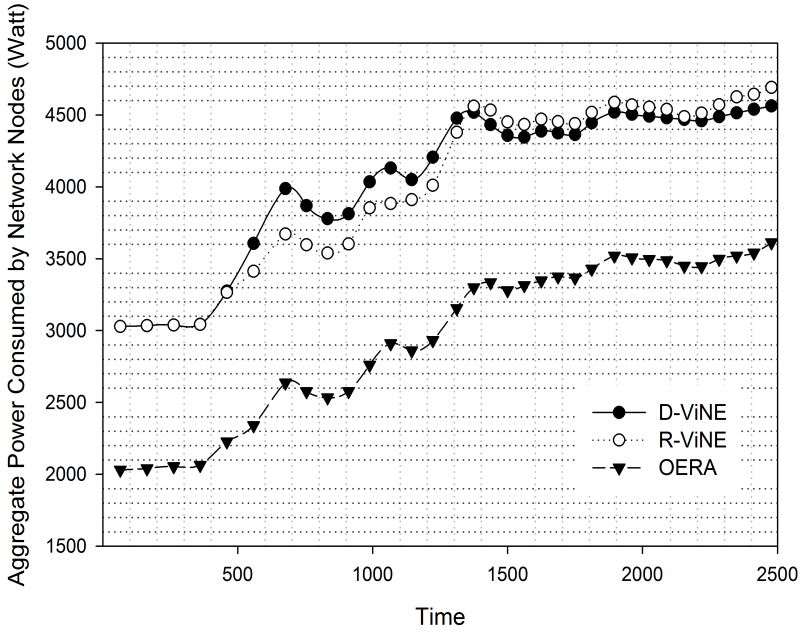
$$Power = P^n_{network} + P^n_{comp} \tag{6.22}$$

- The node and link utilization are defined as the ratio of the number of nodes (resp. links) used over the total number of nodes (resp. links).

$$\begin{aligned} Node\ Utilization &= \frac{Number\ of\ used\ nodes}{Number\ of\ total\ nodes} \\ Link\ Utilization &= \frac{Number\ of\ used\ links}{Number\ of\ total\ links} \end{aligned} \tag{6.23}$$

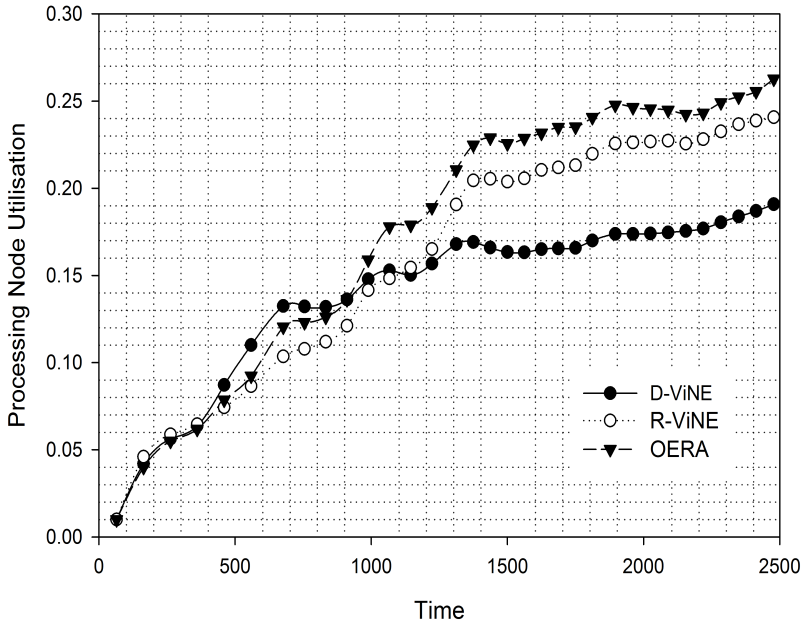


(a) Computational Hosts

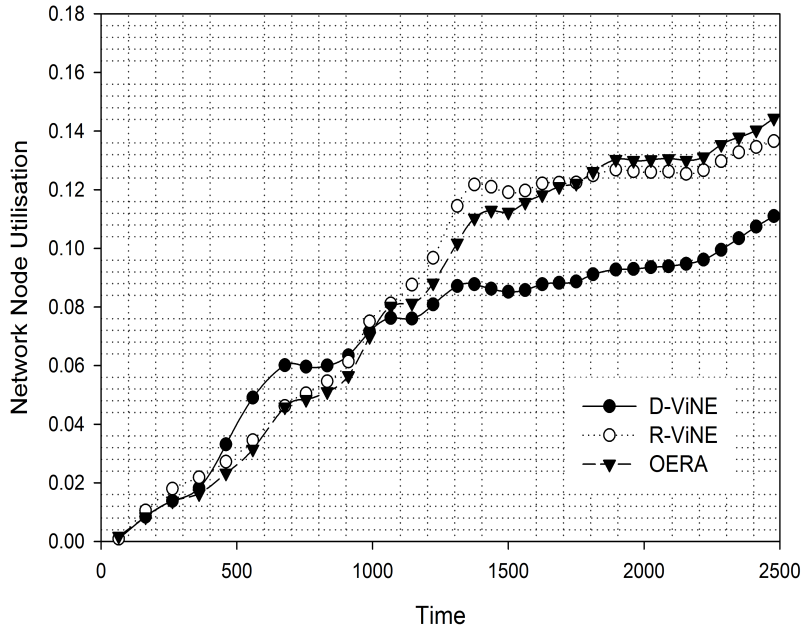


(b) Network Devices

Fig. 6.3 Power Consumption Evaluation



(a) Computing Nodes



(b) Network Nodes

Fig. 6.4 The Resource Utilization over Time

Fig. 6.2 shows that the OERA improves the acceptance ratio by approximately 11% and 17% in comparison with D-ViNE and R-ViNE. Fig. 6.3a and Fig. 6.3b show the power consumption of the computational hosts and NDs. The computational hosts are modelled using Eq. (6.3) for power consumption evaluation in comparing all three techniques. Once the acceptance ratio stabilizes, the average power consumption in computational hosts using OERA is higher than that of R-ViNE and D-ViNE; this is merely because OERA, as shown in Fig. 6.2, supports an 11% higher acceptance rate which results in improved resource allocation. In comparison, OERA exhibits approximately 9% lower power dissipation in the ND resources as shown in Fig. 6.3b. In particular, this power saving result is due to the use of proposed frequency scaling ability in NDPM. Fig. 6.4 illustrates the resource utilization (nodes and links) comparison. Here, OERA achieves better resource utilization than *D-ViNE*. OERA and *D-ViNE* both perform better than *R-ViNE*.

6.4 Parameter Selection in OERA

The objective function in Eq. (6.2) depends on the weight factors η (weight of network node) and ν (weight of computing node). These weights allow the network service providers to control cost of the network operations. For example, in a densely populated area with high resource demands, the weights of both network nodes and computing nodes can be configured a little higher than the sparely populated areas in order to decrease the probability of being selected for further requests.

In this section, we investigate the weight factors for establishing their relationships with the residual capacity of the network resources ($R_{freq}(n)$) and computing resources ($R_c(n)$). If α_n is set to $R_{freq}(n)$ and β_n is set to $R_c(n)$, the weight values will be approximately equal to 1 for every network node. In comparison, if both α_n and β_n are set to 1, the objective function Eq.(6.2) will rely on $R_{freq}(n)$ and $R_c(n)$. We present three use-cases to evaluate the

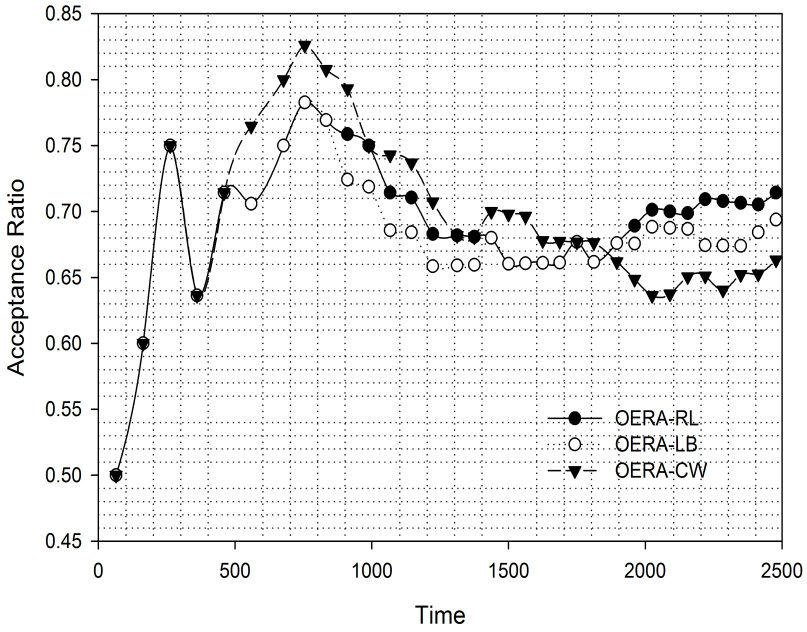


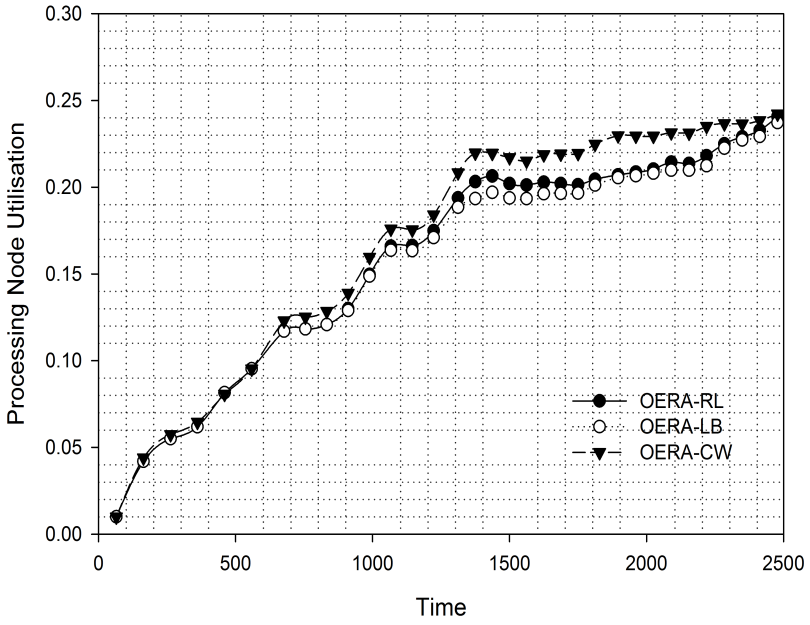
Fig. 6.5 Acceptance Ratio with Weight Factor Variations

impact of weight variations called OERA-CW(Constant Weight), OERA-LB(Load Balanced) and OERA-RL(Random Load) respectively as shown in Table 6.3.

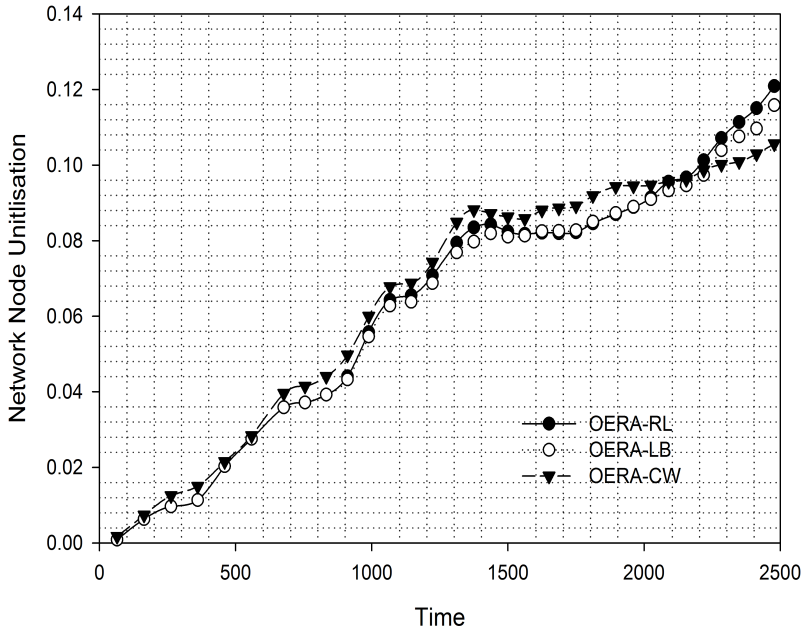
Table 6.3 OERA Weight Assignments Use-Cases

Notation	Description
OERA-CW	α_n is set to $R_{freq}(n)$, β_n is set to $R_c(n)$.
OERA-LB	α_n and β_n are set to 1.
OERA-RL	α_n is uniformly distr. in $[1, R_{freq}(n)]$, β_n is uniformly distr. in$[1, R_c(n)]$.

We observe from Fig. 6.5 that both OERA-LB and OERA-RL have moderately better acceptance ratio than OERA-CW. In OERA-CW, the configurations of α_n and β_n ignore the resource allocation status of different network nodes. Whereas, in OERA-LB and OERA-RL, the weight parameters have direct impact over the residual capacity of the network resources. This reflects that the lower residual capacity of a particular network node leads to the higher value of Eq. (6.2), which decreases the resource utilisation and keep the critical nodes available to increase the acceptance ratio. Fig. 6.6 shows that both OERA-LB and OERA-RL have merely higher network node utilisation than OERA-CW. This is due to the



(a) Computing Nodes



(b) Network Nodes

Fig. 6.6 Resource utilisation with Weight Factor Variations

Table 6.4 OERA Use-cases Performance Results

	Acceptance Ratio	Comp. Nodes Util.	Network Nodes Util.
OERA-CW	0.663	0.242	0.106
OERA-LB	0.693	0.237	0.116
OERA-RL	0.714	0.241	0.121

fact of avoiding lower residual capacity nodes. The OERA-LB and OERA-RL select the nodes with higher residual capacities to improve the performance.

6.5 Summary

This chapter proposed an Online Energy efficient Resource Allocation (OERA) scheme that using the NDPM model (has been discussed in previous chapter) to reduce power consumption in NDs supporting frequency scaling. The model's results show 11-17% more virtual networks can be supported (i.e., a higher acceptance ratio) as well as a 9% reduction in power consumption compared to existing solutions. The goal of the energy aware problem (OREA) introduced in this chapter is to allocate the set of virtual network requests in a reduced group of physical network equipment and propose a mixed integer program (MIP) to optimally solve it. The implementation of OREA model is not scalable for very large networks as MIP is NP-hard. However, the OREA model can provide an optimal bound for future energy-aware resource allocation heuristics.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

Rising energy costs and the requirements to reduce global Greenhouse Gas emissions have made energy efficient networks a vital research topic. One effective approach to controlling the power dissipation of CMOS network devices is frequency scaling. This thesis describes the implementation and evaluation of the frequency scaling technique in a CMOS network device. Two multiple frequency clocks approaches to support frequency scaling, namely Dynamic Clock Generation (DCG) and Dynamic Clock Selection (DCS) are developed. Two control policies, namely the Escalator and the Hysteresis control policies are introduced in the thesis to select the operating frequency. The performance of these policies has been compared to that of a conventional fixed-clock system using both numerical and simulation studies. A model of CMOS device energy consumption and its dependence on system clock rate has been developed. Empirical measurements of power consumption from the frequency scaled switch testbed have been used both to validate the model and to obtain realistic values for its parameters.

The control policies on the bestbed react only to local conditions (specifically the queue length). The wider problem of how to minimise network energy consumption, as

opposed to merely network device energy consumption, also requires attention. This has been addressed in this thesis in the context of edge network. Edge networks are assumed to offer better performance than cloud networks because of their proximity to the end user. The thesis contains a description and evaluation of the online energy efficient resource allocation (OERA), a novel algorithm to reduce overall energy consumption in an edge network, when overlaying a virtual network on top of the physical substrate network. This uses the network device power model (NDPM) developed earlier in a linear programming algorithm to reduce overall power consumption. The simulation results show that OERA gives rise to better acceptance ratios and lower power consumption than two well-known existing schemes.

7.2 Future Work

The work described here can be extended further, in directions such as the following:

- Dynamic Voltage Scaling

  The power consumption analysis of CMOS devices presented in Section 3.1 shows a quadratic relationship between the dynamic power consumption and the supply voltage. Thus, more dramatic improvements in energy consumption can be achieved using voltage scaling than frequency scaling (where the dependence is linear rather than quadratic). This thesis focused on frequency scaling technique as this approach can be easily implemented by accessing hardware resources to control the operating clock. In contrast, dynamically scaling the supply voltage needs more complex hareware support, which is not widely provided by existing technology. Recently, hardware has begun to emerge with voltage scaling supported, such as the 7 Series FPGAs and Zynq-7000 Programmable SoCs from Xilinx [96], so that research on the voltage scaling will be feasible. Since higher voltages allow faster charge and discharge of CMOS capacitances, and thus faster operation, the minimisation of energy consumption by combining voltage and frequency scaling will present interesting challenges.

- Optimal Control Policy features "Just Enough" Work

  Two frequency scaling control policies have been discussed in Section 5.2. Future work will seek to identify an optimal control policy where the mean delay is ideally invariant with offered load, so that the system does "just enough" work to meet delay and other performance targets. Such an algorithm should exhibit optimal energy efficiency, and the work described in this thesis provides a soled platform for evaluating candidate algorithms. The ultimate goal will be to implement real-time scheduling, where the hardware state is modified continually in response to QoS demands.

- Improved Clock Generation

  Recent FPGA designs feature two built-in digital PLLs. Using these to implement two frequency-synthesised clocks would allow a hybrid clock generating scheme, where the system alternates between the two clocks using DCS and the clock signal is never chaotic. This would provide a greater range of clocks than DCS, for the same footprint, without the clock stability issues of DCG.

- Network Resource Reconfiguration

  The OERA algorithm presented in Section 6.1 maps virtual network requests to the underlying substrate network in an energy efficient manner. It will be of increasing importance in the future to support user and application mobility, and this is particularly challenging when using the edge networking concept. Extending OERA to efficiently handle the resulting migration of virtual network requests from one geographical location to another will be a useful enhancement to its scope of operation.

- Resource Allocation across Autonomous Domain

  OERA is a centralized resource allocation approach, which assumes that the entire substrate network is operated and maintained by a single infrastructure network provider (InP), and so the allocation algorithm has a global view of the entire network. However,

the physical network typically will be under the control of multiple InPs. This will result in decisions having to be made based on incomplete information, as each InP will not disclose full state information to its neighbours. Some research on this Multi-InP problem [39] [40] have been undertaken, and OERA can be extended to incorporate such capabilities.

References

[1] (2015) BOINC volunteer computing. [Online]. Available: https://boinc.berkeley.edu/trac/wiki/VolunteerComputing

[2] K.P.saharan and A. Kumar, "Fog in comparison to cloud: A survey," *International Journal of Computer Applications*, vol. 122, no. 3, pp. 10–12, July 2015.

[3] C. K. T. T. Truong-Huu and D. Niyato, "To offload or to wait: An opportunistic offloading algorithm for parallel tasks in a mobile cloud," in *IEEE 6th International Conference on Cloud Computing Technology and Science*, 2014, pp. 182–189.

[4] M. Chiang, S. Ha, C. L. I, F. Risso, and T. Zhang, "Clarifying fog computing and networking: 10 questions and answers," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 18–20, April 2017.

[5] Global e-sustainability initiative. [Online]. Available: http://gesi.org/

[6] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys Tutorials*, vol. 13, no. 2, pp. 223–244, 2011.

[7] W. Van Heddeghem, S. Lambert, B. Lannoo, D. Colle, M. Pickavet, and P. Demeester, "Trends in worldwide ict electricity consumption from 2007 to 2012," *Comput. Commun.*, vol. 50, pp. 64–76, Sep. 2014. [Online]. Available: http://dx.doi.org/10.1016/j.comcom.2014.02.008

[8] M. Andrews, A. F. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–9.

[9] M. Chowdhury, M. R. Rahman, and R. Boutaba, "Vineyard: Virtual network embedding algorithms with coordinated node and link mapping," *IEEE/ACM Transactions on Networking*, vol. 20, no. 1, pp. 206–219, Feb 2012.

[10] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, April 2005.

[11] J. S. Turner and D. E. Taylor, "Diversifying the internet," in *GLOBECOM '05. IEEE Global Telecommunications Conference, 2005.*, vol. 2, Dec 2005, pp. 6 pp.–760.

[12] A. Botta, W. de Donato, V. Persico, and A. Pescapé, "On the integration of cloud computing and internet of things," in *2014 International Conference on Future Internet of Things and Cloud*, Aug 2014, pp. 23–30.

[13] M. Jarschel, T. Zinner, T. Hossfeld, P. Tran-Gia, and W. Kellerer, "Interfaces, attributes, and use cases: A compass for sdn," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 210–217, June 2014.

[14] (2013) OpenFlow Switch Specification. [Online]. Available: https://www.opennetworking.org/

[15] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355746

[16] J. D. Touch, Y. S. Wang, V. Pingali, L. Eggert, R. Zhou, and G. G. Finn, "A global x-bone for network experiments," in *First International Conference on Testbeds and Research Infrastructures for the DEvelopment of NeTworks and COMmunities*, Feb 2005, pp. 194–203.

[17] D. Komosny, J. Pruzinsky, P. Ilko, J. Polasek, P. Masek, and O. Kocatepe, "On geographic coordinates of planetlab europe," in *2015 38th International Conference on Telecommunications and Signal Processing (TSP)*, July 2015, pp. 642–646.

[18] S. da Silva, Y. Yemini, and D. Florissi, "The netscript active network system," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 3, pp. 538–551, Mar 2001.

[19] E. Grasa, S. Figuerola, A. Forns, G. Junyent, and J. Mambretti, "Extending the uclp software with a dynamic optical multicast service to support high performance digital media," in *2008 International Conference on Optical Network Design and Modeling*, March 2008, pp. 1–6.

[20] J. E. van der Merwe and I. M. Leslie, *Switchlets and Dynamic Virtual ATM Networks*. Boston, MA: Springer US, 1997, pp. 355–368. [Online]. Available: https://doi.org/10.1007/978-0-387-35180-3_27

[21] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar, "An integrated experimental environment for distributed systems and networks," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 255–270, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844152

[22] N. Niebert, S. Baucke, I. El-Khayat, M. Johnsson, B. Ohlman, H. Abramowicz, K. Wuenstel, H. Woesner, J. Quittek, and L. M. Correia, "The way 4ward to the

creation of a future internet," in *2008 IEEE 19th International Symposium on Personal, Indoor and Mobile Radio Communications*, Sept 2008, pp. 1–5.

[23] N. M. M. K. Chowdhury and R. Boutaba, "Network virtualization: state of the art and research challenges," *IEEE Communications Magazine*, vol. 47, no. 7, pp. 20–26, July 2009.

[24] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 17–29, Mar. 2008. [Online]. Available: http://doi.acm.org/10.1145/1355734.1355737

[25] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proceedings IEEE INFOCOM 2006.*, April 2006, pp. 1–12.

[26] A. Schrijver, *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., 1986.

[27] N. M. M. K. Chowdhury, M. R. Rahman, and R. Boutaba, "Virtual network embedding with coordinated node and link mapping," in *INFOCOM 2009, IEEE*, April 2009, pp. 783–791.

[28] X. Cheng, S. Su, Z. Zhang, H. Wang, F. Yang, Y. Luo, and J. Wang, "Virtual network embedding through topology-aware node ranking," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 2, pp. 38–47, Apr. 2011.

[29] G. Sun, H. Yu, L. Li, V. Anand, Y. Cai, and H. Di, "Exploring online virtual networks mapping with stochastic bandwidth demand in multi-datacenter," *Photonic Network Communications*, vol. 23, no. 2, pp. 109–122, 2012.

[30] T. Guo, N. Wang, K. Moessner, and R. Tafazolli, "Shared backup network provision for virtual network embedding," in *2011 IEEE International Conference on Communications (ICC)*, June 2011, pp. 1–5.

[31] S. Zhang, Z. Qian, B. Tang, J. Wu, and S. Lu, "Opportunistic bandwidth sharing for virtual network mapping," in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*, Dec 2011, pp. 1–5.

[32] I. Houidi, W. Louati, W. B. Ameur, and D. Zeghlache, "Virtual network provisioning across multiple substrate networks," *Computer Networks*, vol. 55, no. 4, pp. 1011 – 1023, 2011.

[33] J. F. Botero, X. Hesselbach, M. Duelli, D. Schlosser, A. Fischer, and H. de Meer, "Energy efficient virtual network embedding," *IEEE Communications Letters*, vol. 16, no. 5, pp. 756–759, May 2012.

[34] J. Lischka and H. Karl, "A virtual network mapping algorithm based on subgraph isomorphism detection," in *Proceedings of the 1st ACM Workshop on Virtualized Infrastructure Systems and Architectures*, ser. VISA '09. New York, NY, USA: ACM, 2009, pp. 81–88.

[35] N. Farooq Butt, M. Chowdhury, and R. Boutaba, *Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 27–39.

[36] H. Yu, V. Anand, C. Qiao, H. Di, and X. Wei, "A cost efficient design of virtual infrastructures with joint node and link mapping," *Journal of Network and Systems Management*, vol. 20, no. 1, pp. 97–115, 2012.

[37] J. Liu, T. Huang, J.-y. Chen, and Y.-j. Liu, "A new algorithm based on the proximity principle for the virtual network embedding problem," *Journal of Zhejiang University SCIENCE C*, vol. 12, no. 11, pp. 910–918, 2011.

[38] S. Zhang, J. Wu, and S. Lu, "Virtual network embedding with substrate support for parallelization," in *Global Communications Conference (GLOBECOM), 2012 IEEE*, Dec 2012, pp. 2615–2620.

[39] J. Fan and M. H. Ammar, "Dynamic topology configuration in service overlay networks: A study of reconfiguration policies," in *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, April 2006, pp. 1–12.

[40] C. C. Marquezan, L. Z. Granville, G. Nunzi, and M. Brunner, "Distributed autonomic resource management for network virtualization," in *2010 IEEE Network Operations and Management Symposium - NOMS 2010*, April 2010, pp. 463–470.

[41] C. Wang, S. Shanbhag, and T. Wolf, "Virtual network mapping with traffic matrices," in *2012 IEEE International Conference on Communications (ICC)*, June 2012, pp. 2717–2722.

[42] H. Yu, C. Qiao, V. Anand, X. Liu, H. Di, and G. Sun, "Survivable virtual infrastructure mapping in a federated computing and networking system under single regional failures," in *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, Dec 2010, pp. 1–6.

[43] X. Gao, H. Yu, V. Anand, G. Sun, and H. Di, "A new algorithm with coordinated node and link mapping for virtual network embedding based on lp relaxation," in *Asia Communications and Photonics Conference and Exhibition*, Dec 2010, pp. 152–153.

[44] Y. Zhou, Y. Li, D. Jin, L. Su, and L. Zeng, "A virtual network embedding scheme with two-stage node mapping based on physical resource migration," in *Communication Systems (ICCS), 2010 IEEE International Conference on*, Nov 2010, pp. 761–766.

[45] D. Chen, X. Qiu, Z. Qu, S. Zhang, and W. Li, "Algorithm for virtual nodes reconfiguration on network virtualization," in *Advanced Intelligence and Awareness Internet (AIAI 2011), 2011 International Conference on*, Oct 2011, pp. 333–337.

[46] R. Chang and C. Wu, "Green virtual networks for cloud computing," in *2010 5th International ICST Conference on Communications and Networking in China*, Aug 2010, pp. 1–7.

[47] C. Yang, K. Wang, H. Cheng, C. Kuo, and W. C. C. Chu, "Green power management with dynamic resource allocation for cloud virtual machines," in *2011 IEEE International Conference on High Performance Computing and Communications*, Sept 2011, pp. 726–733.

[48] A. Mallik, B. Lin, G. Memik, P. Dinda, and R. P. Dick, "User-driven frequency scaling," *IEEE Computer Architecture Letters*, vol. 5, no. 2, pp. 16–16, Feb 2006.

[49] C. T. Chow, L. S. M. Tsui, P. H. W. Leong, W. Luk, and S. J. E. Wilton, "Dynamic voltage scaling for commercial fpgas," in *Proceedings. 2005 IEEE International Conference on Field-Programmable Technology, 2005.*, Dec 2005, pp. 173–180.

[50] F. Li, Y. Lin, L. He, and J. Cong, "Low-power fpga using pre-defined dual-vdd/dual-vt fabrics," in *Proceedings of the 2004 ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays*, ser. FPGA '04. New York, NY, USA: ACM, 2004, pp. 42–50. [Online]. Available: http://doi.acm.org/10.1145/968280.968288

[51] (1996) The ACPI Specification. [Online]. Available: http://www.acpi.info

[52] A. Cianfrani, V. Eramo, M. Listanti, M. Marazza, and E. Vittorini, "An energy saving routing algorithm for a green ospf protocol," in *2010 INFOCOM IEEE Conference on Computer Communications Workshops*, March 2010, pp. 1–5.

[53] C. Gunaratne, K. Christensen, and B. Nordman, "Managing energy consumption costs in desktop pcs and lan switches with proxying, split tcp connections, and scaling of link speed," *International Journal of Network Management*, vol. 15, no. 5, pp. 297–310, 2005. [Online]. Available: http://dx.doi.org/10.1002/nem.565

[54] S. Nedevschi, L. Popa, G. Iannaccone, S. Ratnasamy, and D. Wetherall, "Reducing network energy consumption via sleeping and rate-adaptation," in *Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 323–336. [Online]. Available: http://dl.acm.org/citation.cfm?id=1387589.1387612

[55] C. Gunaratne, K. Christensen, B. Nordman, and S. Suen, "Reducing the energy consumption of ethernet with adaptive link rate (alr)," *IEEE Transactions on Computers*, vol. 57, no. 4, pp. 448–461, April 2008.

[56] A. Francini and D. Stiliadis, "Rate adaptation for energy efficiency in packet networks," *Bell Labs Technical Journal*, vol. 15, no. 2, pp. 131–146, Sept 2010.

[57] M. Andrews, S. Antonakopoulos, and L. Zhang, "Energy-aware scheduling algorithms for network stability," in *2011 Proceedings IEEE INFOCOM*, April 2011, pp. 1359–1367.

[58] W. Meng, Y. Wang, C. Hu, K. He, J. Li, and B. Liu, "Greening the internet using multi-frequency scaling scheme," in *2012 IEEE 26th International Conference on Advanced Information Networking and Applications*, March 2012, pp. 928–935.

[59] T. Song, Z. Jiang, Y. Wei, X. Shi, X. Ma, O. Ormond, M. Collier, and X. Wang, "Traffic aware energy efficient router: Architecture, prototype and algorithms," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3814–3827, Dec 2016.

[60] R. Bolla, R. Bruschi, F. Davoli, L. D. Gregorio, P. Donadio, L. Fialho, M. Collier, A. Lombardo, D. R. Recupero, and T. Szemethy, "The green abstraction layer: A standard power-management interface for next-generation network devices," *IEEE Internet Computing*, vol. 17, no. 2, pp. 82–86, March 2013.

[61] *Green Abstraction Layer (GAL):Power management capabilities of the future energy telecommunication fixed network nodes*, ETSI Std. ES 203 237, 2014.

[62] R. Bolla, R. Bruschi, M. Chiappero, L. D'Agostino, P. Lago, C. Lombardo, S. Mangialardi, and F. Podda, "EE-DROP: An energy-aware router prototype," in *Digital Communications - Green ICT (TIWDC), 2013 24th Tyrrhenian International Workshop on*, Sept 2013, pp. 1–6.

[63] R. Bolla, C. Lombardo, R. Bruschi, and S. Mangialardi, "DROPv2: energy efficiency through network function virtualization," *Network, IEEE*, vol. 28, no. 2, March 2014.

[64] (2013) Mellanox Technologies. [Online]. Available: http://www.mellanox.com/

[65] R. H. Dennard, F. H. Gaensslen, V. L. Rideout, E. Bassous, and A. R. LeBlanc, "Design of ion-implanted mosfet's with very small physical dimensions," *IEEE Journal of Solid-State Circuits*, vol. 9, no. 5, pp. 256–268, Oct 1974.

[66] B. N. Anantha P. Chandrakasan and J. M. Rabaey, *Digital Integrated Circuits: A Design Perspective*. Pearson, 2003.

[67] G. K.Yeap, *Practical Low Power Digital VLSI Design*. Massachusetts USA: Kluwer Academic Publishers, 2002.

[68] A. P. B. Chandrakasan and R. W., *Low power digital CMOS design*. Boston; London: Kluwer Academic Publishers, c1995, 1995.

[69] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand, "Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits," *Proceedings of the IEEE*, vol. 91, no. 2, pp. 305–327, Feb 2003.

[70] S. T. Tim Tuan, "The power of fpga architectures." [Online]. Available: www.edn.com/Pdf/ViewPdf?contentItemId=4137952

[71] R. Bolla, F. Davoli, R. Bruschi, K. Christensen, F. Cucchietti, and S. Singh, "The potential impact of green technologies in next-generation wireline networks: Is there room for energy saving optimization?" *IEEE Communications Magazine*, vol. 49, no. 8, pp. 80–86, August 2011.

[72] M. Jimeno, K. Christensen, and B. Nordman, "A network connection proxy to enable hosts to sleep and save energy," in *2008 IEEE International Performance, Computing and Communications Conference*, Dec 2008, pp. 101–110.

[73] C. K. J. and G. F. 'Bo', "Enabling power management for network-attached computers," *International Journal of Network Management*, vol. 8, no. 2, pp. 120–130. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/%28SICI%291099-1190%28199803/04%298%3A2%3C120%3A%3AAID-NEM273%3E3.0.CO%3B2-R

[74] Y.-H. Lu, L. Benini, and G. D. Micheli, "Dynamic frequency scaling with buffer insertion for mixed workloads," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 21, no. 11, pp. 1284–1305, Nov 2002.

[75] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *IEEE/ACM International Conference on Computer Aided Design, 2002. ICCAD 2002.*, Nov 2002, pp. 195–202.

[76] F. Li, Y. Lin, and L. He, "Fpga power reduction using configurable dual-vdd," in *Proceedings. 41st Design Automation Conference, 2004.*, July 2004, pp. 735–740.

[77] ——, "Vdd programmability to reduce fpga interconnect power," in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.*, Nov 2004, pp. 760–765.

[78] ——, "Field programmability of supply voltages for fpga power reduction," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 26, no. 4, pp. 752–764, April 2007.

[79] A. Shen, A. Ghosh, S. Devadas, and K. Keutzer, "On average power dissipation and random pattern testability of cmos combinational logic networks," in *1992 IEEE/ACM International Conference on Computer-Aided Design*, Nov 1992, pp. 402–407.

[80] T. S. Czajkowski and S. D. Brown, "Using negative edge triggered ffs to reduce glitching power in fpga circuits," in *2007 44th ACM/IEEE Design Automation Conference*, June 2007, pp. 324–329.

[81] J. Lamoureux, G. G. F. Lemieux, and S. J. E. Wilton, "Glitchless: Dynamic power minimization in fpgas through edge alignment and glitch filtering," *IEEE Transactions*

on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 11, pp. 1521–1534, Nov 2008.

[82] S. Huda, M. Mallick, and J. H. Anderson, "Clock gating architectures for fpga power reduction," in *2009 International Conference on Field Programmable Logic and Applications*, Aug 2009, pp. 112–118.

[83] Q. Wang, S. Gupta, and J. H. Anderson, "Clock power reduction for virtex-5 fpgas," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '09. New York, NY, USA: ACM, 2009, pp. 13–22. [Online]. Available: http://doi.acm.org/10.1145/1508128.1508132

[84] K. Agarwal, K. Nowka, H. Deogun, and D. Sylvester, "Power gating with multiple sleep modes," in *Proceedings of the 7th International Symposium on Quality Electronic Design*, ser. ISQED '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 633–637. [Online]. Available: http://dx.doi.org/10.1109/ISQED.2006.102

[85] S. Donthi and R. L. Haggard, "A survey of dynamically reconfigurable fpga devices," in *Proceedings of the 35th Southeastern Symposium on System Theory, 2003.*, March 2003, pp. 422–426.

[86] S. Liu, R. N. Pittman, A. Form, and J. L. Gaudiot, "On energy efficiency of reconfigurable systems with run-time partial reconfiguration," in *ASAP 2010 - 21st IEEE International Conference on Application-specific Systems, Architectures and Processors*, July 2010, pp. 265–272.

[87] D. B. Thomas, L. Howes, and W. Luk, "A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '09. New York, NY, USA: ACM, 2009, pp. 63–72. [Online]. Available: http://doi.acm.org/10.1145/1508128.1508139

[88] W. Wang, M. Bolic, and J. Parri, "pvfpga: Accessing an fpga-based hardware accelerator in a paravirtualized environment," in *2013 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Sept 2013, pp. 1–9.

[89] C. Pascoe, A. Lawande, H. Lam, A. George, and W. G. Farmerie, "Reconfigurable supercomputing with scalable systolic arrays and instream control for wavefront genomics processing," 2010.

[90] K. F. Lysakov and M. Y. Shadrin, "Fpga-based hardware accelerator for high-performance data-stream processing," *Pattern Recognition and Image Analysis*, vol. 23, no. 1, pp. 26–34, Mar 2013. [Online]. Available: https://doi.org/10.1134/S1054661812030054

[91] J. Fowers, G. Brown, P. Cooke, and G. Stitt, "A performance and energy comparison of fpgas, gpus, and multicores for sliding-window applications," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '12. New York, NY, USA: ACM, 2012, pp. 47–56. [Online]. Available: http://doi.acm.org/10.1145/2145694.2145704

[92] J. Sun, G. D. Peterson, and O. O. Storaasli, "High-performance mixed-precision linear solver for fpgas," *IEEE Transactions on Computers*, vol. 57, no. 12, pp. 1614–1623, Dec 2008.

[93] http:/NetFPGA.org/, "Netfpga," *NetFPGA*, 2010. [Online]. Available: http:/NetFPGA.org/

[94] http://NetFPGA.org/1G_specs.html/, "Netfpga1g," *NetFPGA1G*, 2010. [Online]. Available: http://NetFPGA.org10G/1G_specs.html/

[95] NetFPGA 10G Platform. [Online]. Available: http://NetFPGA.org/10G_specs.html/

[96] XILINX Website. [Online]. Available: http://www.xilinx.com/itp/xilinx10/isehelp/xpa_c_togglerates.htm

[97] M. Ghasemzadeh, S. Mahdavi, A. Zokaei, and K. Hadidi, "A new adaptive pll to reduce the lock time in 0.18 um technology," in *2016 MIXDES - 23rd International Conference Mixed Design of Integrated Circuits and Systems*, June 2016, pp. 140–142.

[98] R. Bolla, R. Bruschi, F. Davoli, and F. Cucchietti, "Energy efficiency in the future internet: A survey of existing approaches and trends in energy-aware fixed network infrastructures," *IEEE Communications Surveys Tutorials*, vol. 13, no. 2, pp. 223–244, Second 2011.

[99] L. L. Analyzer, "Logicport logic analyzer," *LogicPort Logic Analyzer*.

[100] "A new power profiling method and power scaling mechanism for energy-aware NetFPGA gigabit router," *Computer Networks*, vol. 78, pp. 4 – 25, 2015.

[101] http://ultraviewcorp.com, "PCIEXT-64U-Live Insertion PCI Bus Extender," *PCIEXT-64U-Live Insertion PCI Bus Extender*, 2012. [Online]. Available: http://ultraviewcorp.com

[102] National Instruments, "NI 6251 Device Specifications," *NI 6251 Device Specifications*, 2012. [Online]. Available: http://www.ni.com/pdf/manuals/375213c.pdf

[103] Open source network tester. [Online]. Available: http://osnt.org/

[104] http://NetFPGA.org/10G_specs.html/, "Xilinx:quiescentpower," *Xilinx*, 2009. [Online]. Available: https://www.xilinx.com/support/documentation/sw_manuals/xilinx11/xpa_c_quiescent_power.htm

[105] Y. D. Lin, E. T. H. Chu, Y. C. Lai, and T. J. Huang, "Time-and-energy-aware computation offloading in handheld devices to coprocessors and clouds," *IEEE Systems Journal*, vol. 9, no. 2, pp. 393–405, June 2015.

[106] J. Wolberg, *Data Analysis Using the Method of Least Squares*. USA: Springer, 2006.

[107] B. Razavi, *Design of Analog CMOS Integrated Circuits*. USA: McGraw-Hill Education, 2000.

[108] M. Bertoli, G. Casale, and G. Serazzi, "Jmt: performance engineering tools for system modeling," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 10–15, 2009.

[109] G. S. Fishman, "Statistical analysis for queueing simulations," Jan 1973. [Online]. Available: https://ideas.repec.org/a/inm/ormnsc/v20y1973i3p363-369.html

[110] P. Heidelberger and P. D. Welch, "A spectral method for confidence interval generation and run length control in simulations," *Commun. ACM*, vol. 24, no. 4, pp. 233–245, Apr. 1981. [Online]. Available: http://doi.acm.org/10.1145/358598.358630

[111] F. D. R. Ford, Lester Randolph, *Flows in networks*. Princeton USA: Princeton N.J: Princeton University Press, 1962.

[112] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 13–23, Jun. 2007. [Online]. Available: http://doi.acm.org/10.1145/1273440.1250665

[113] B. Lu, J.-y. Chen, H.-y. Cui, T. Huang, and Y.-j. Liu, "A virtual network mapping algorithm based on integer programming," *Journal of Zhejiang University SCIENCE C*, vol. 14, no. 12, pp. 899–908, 2013. [Online]. Available: http://dx.doi.org/10.1631/jzus.C1300120

[114] (2000) Gtitm. [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

Appendix A

Related Resource Allocation Algorithms

In this appendix, several typical VNE algorithms will be discussed elaborately.

A.1 Stress Based Algorithm

The paper [25] describes this algorithm in detail. Node stress and link stress has been discussed in Section 2.5.2, in terms of mathematical evaluations. The prime objective of this algorithm is to maintain a balanced node and link stress with a low value in the substrate network. To achieve this goal, [25] has proposed a heuristic approach which is transcribed in algorithm. The key idea lies with the algorithm, firstly, select a cluster of substrate nodes that are combination of both lightly stress nodes and links, secondly, mapping the virtual nodes to those substrate nodes, and finally, mapping links between the nodes processed in the previous stage.

A.2 Path Splitting Algorithms

This algorithm is firstly proposed in [24], as shown in algorithm A.1. The target of this algorithm is to maximize the long average revenue. It is an online algorithm which is

designed to efficiently allocate resources to VNRs according to their revenue. Node mapping is performed by using a greedy algorithm based on the residual resources of the substrate nodes. Path splitting algorithm assumes that the virtual path can be mapped into multiple physical paths. To achieve the expected mapping the algorithm constructs linear constraints based on MFP. Node remapping (line 4) will be recursively used to find alternative nodes, because there is a probability of unsuccessful solution to MFP, therefor mapping the links by another MFP have to be performed to maximizing the likelihood of successful mapping.

Algorithm A.1: Splittable Path Mapping Algorithm based on [24]

1 **Step 1:** For all requests with splittability, construct linear constraint on the commodities for each substrate link.

2 **Step 2:** Solve Multicommodity Flow Problem (MFP).

3 **Step 3:** If feasible, stop.

Node Remapping :

4 **Step 4:** Randomly choose one virtual link that is originally mapped at the *bottleneck* link, pick one end of the virtual link and map it to another substrate node with maximum remaining resource H. Then GOTO **Step 2** with new linear constraints.

Step 5: If remapping of virtual nodes for T_try times does not produce a feasible solution, eliminate one of the VN requests having the "largest" impact on infeasibility. Then, construct the linear constrains only with the remaining requests, and GOTO **Step 2**

▷ $H(n^s) = CPU(n^s) \sum_{l^s \in L(n^s)} bw(l^s)$

A.3 Coordinated Node and Link Algorithms

Chowdhury *et al.* [9] proposed an online algorithm to resolve the VNE problem by presenting a new node mapping stage (introducing Metanodes and Metaedges). Comparing with path splitting algorithms [24], this algorithm encounters a requirement for geographical location of the virtual nodes. This algorithm uses Mixed integer linear programming (MILP) to solve the optimized objective, i.e. to find the minimum value of weighted sum of the allocated bandwidth and CPU. Before the node mapping stage, this algorithm creates an augmented graph over the substrate network by considering the location constraints of the virtual nodes. The linear programing equations defining this algorithm take both node and link constrains into account and for avoiding *NP-hard* issue of MILP [26], the algorithm relaxes the integer constraints leading to the solutions which then can be solved in polynomial time. Then the algorithm approximates the values of the binary variables for the initial MILP by two policies: deterministic and randomized rounding. Based on this, the proposed algorithms are named as D-ViNE (Deterministic VN Embedding) and R-ViNE (Randomized VN Embedding) respectively.

A.4 Topology-Aware VNE Algorithms

In [35], Butt *et al.* propose a key parameter called *Critical Index* of the substrate nodes and links. *CI* is the value representing a node or a link in a connected topology, and it measures the unavailability likelihood of the residual substrate network becoming an unconnected graph if a certain node or a link is removed.

A.5 Node Ranking Based VNE Algorithms

Several papers propose a ranking system for the virtual and substrate nodes along with applying greedy algorithm into the mapping stage. [24] gives a hybrid formula, which considers both computing and link capacity into it, to measure the value of *NodeRank* (NR) for each node, as follows.

$$H(u) = CPU(u) \sum_{l \in L(u)} BW(l) \tag{A.1}$$

Where, $CPU(u)$ donates the computing capacity of node u, and $\sum_{l \in L(u)} BW(l)$ gives the sum of link capacities of all directly connected links to node u. NR of node u, expressed as a product of its computing capacity and total link capacities.

[28] takes a step further from above NR definition. The novelty of the NR calculation in [28] is inspired by the Google's PageRank search algorithm, which is then again based on the classical model of Markov random walk. The initial NR of node u defined as:

$$NR^{(0)}(u) = \frac{H(u)}{\sum_{v \in V} H(v)} \tag{A.2}$$

and defined the transmission probability, which are then divided into Jump probability and Forward probability, from node u to node v as:

$$p^{J}_{uv} = \frac{H(v)}{\sum_{w \in V} H(w)} \tag{A.3}$$

$$p^{F}_{uv} = \frac{H(v)}{\sum_{w \in nbr(u)} H(w)} \tag{A.4}$$

In which, the Jump probability implies the multi-hop between u and v. The NR of node $v \in V$ at time $(t+1)$ is defined as:

$$NR^{(t+1)}(v) = \sum_{u \in V} p^J_{uv} \cdot p^J_u \cdot NR^{(t)}(u) \quad + \sum_{u \in nbr(u)} p^F_{uv} \cdot p^F_u \cdot NR^{(t)}(u) \tag{A.5}$$

Where the p^J_u and p^F_u are bias factors for weighing Jump probability and Forward probability. In [28], they are set as 0.15 and 0.85 respectively. The above equation can be simplified as follows:

$$NR^{(t+1)} = \mathbf{T} \cdot NR^{(t)}, \tag{A.6}$$

and

$$\mathbf{T} = \begin{pmatrix} p^J_{11} & p^J_{12} & \cdots & p^J_{1n} \\ p^J_{21} & p^J_{22} & \cdots & p^J_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p^J_{n1} & p^J_{n2} & \cdots & p^J_{nn} \end{pmatrix} \cdot \begin{pmatrix} p^J_1 & 0 & \cdots & 0 \\ 0 & p^J_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p^J_n \end{pmatrix} \quad + \quad \begin{pmatrix} 0 & p^F_{12} & \cdots & p^F_{1n} \\ p^F_{21} & 0 & \cdots & p^F_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ p^F_{n1} & p^F_{n2} & \cdots & 0 \end{pmatrix} \cdot \begin{pmatrix} p^F_1 & 0 & \cdots & 0 \\ 0 & p^F_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & p^F_n \end{pmatrix} \tag{A.7}$$

T is stable due to a fact that a Markov matrix guarantees that one is an eigenvalue of it (See the MIT opencourseware 18.06 Lecture 24). The NR value for each node can be calculated by algorithm A.2

Algorithm A.2: NodeRank Computation based on [28]

Given a positive value ε, $i \leftarrow 0$

while $\delta \geq \varepsilon$ **do**

    $NR^{i+1} \leftarrow \mathbf{T} \cdot NR^{(}i)$;

    $\delta \leftarrow ||NR^{(i+1)} - NR^{(i)}||$;

    $i++$;

end

Appendix B

Publications arising from this research

Pengcheng Liu, A. Ghiasian, X. Wang and M. Collier, "A programmable energy efficient 40 Gb/s switch using frequency scaling and OpenFlow", 2017 IEEE International Conference on Communications Workshops (ICC Workshops), Paris, 2017, pp. 405-410.

Pengcheng Liu, Xiaojun Wang, S R Chaudhry, Khalid Javeed and Martin Collier, "Secure Video Streaming with Lightweight Cipher PRESENT in an SDN Testbed", Computers, Materials & Continua. [Accept]

Pengcheng Liu, S R Chaudhry, Tao Huang, Xiaojun Wang, and Martin Collier, "Multi-factorial Energy Aware Resource Management in Edge Networks", IEEE Transactions on Green Communications and Networking. [Major Review]

Pengcheng Liu, S R Chaudhry, Xiaojun Wang, and Martin Collier, "Service Rate Control Modeling for Frequency Scaled CMOS-Based Green Router", ACM ACM Trans. Model. Perform. Eval. Comput. Syst. [Submitted]

Daniel Irwin, **Pengcheng Liu**, Saqib Chaudhry, Xiaojun Wang and Martin Collier, "A Performance Comparison of the PRESENT Lightweight Cryptography Algorithm on Different Hardware Platforms", 2018 Irish Signals and Systems Conference, Belfast 2018.