Experimentation on Dynamic Congestion Control in Software Defined Networking (SDN) and Network Function Virtualisation (NFV)

A thesis submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy (PhD)

by

AMALINA FARHAN KAMARUDDIN

Department of Electronic and Computer Engineering
College of Engineering, Design and Physical Sciences
Brunel University London

December 2017

Abstract

In this thesis, a novel framework for dynamic congestion control has been proposed. The study is about the congestion control in broadband communication networks. Congestion results when demand temporarily exceeds capacity and leads to severe degradation of Quality of Service (QoS) and possibly loss of traffic. Since traffic is stochastic in nature, high demand may arise anywhere in a network and possibly causing congestion. There are different ways to mitigate the effects of congestion, by rerouting, by aggregation to take advantage of statistical multiplexing, and by discarding too demanding traffic, which is known as admission control. This thesis will try to accommodate as much traffic as possible, and study the effect of routing and aggregation on a rather general mix of traffic types.

Software Defined Networking (SDN) and Network Function Virtualization (NFV) are concepts that allow for dynamic configuration of network resources by decoupling control from payload data and allocation of network functions to the most suitable physical node. This allows implementation of a centralised control that takes the state of the entire network into account and configures nodes dynamically to avoid congestion. Assumes that node controls can be expressed in commands supported by OpenFlow v1.3. Due to state dependencies in space and time, the network dynamics are very complex, and resort to a simulation approach. The load in the network depends on many factors, such as traffic characteristics and the traffic matrix, topology and node capacities. To be able to study the impact of control functions, some parts of the environment is fixed, such as the topology and the node capacities, and statistically average the traffic distribution in the network by randomly generated traffic matrices. The traffic consists of approximately equal intensity of smooth, bursty and long memory traffic.

By designing an algorithm that route traffic and configure queue resources so that delay is minimised, this thesis chooses the delay to be the optimisation parameter because it is additive and real-time applications are delay sensitive. The optimisation being studied both with respect to total end-to-end delay and maximum end-to-end delay. The delay is used as link weights and paths are determined by Dijkstra’s algorithm. Furthermore, nodes are configured to serve the traffic optimally which in turn depends on the routing. The proposed algorithm is a fixed-point system of equations that iteratively evaluates routing – aggregation – delay until an equilibrium point is found.

Three strategies are compared: static node configuration where each queue is allocated 1/3 of the node resources and no aggregation, aggregation of real-time (taken as smooth and bursty) traffic onto the same queue, and dynamic aggregation based on the entropy of the traffic streams and their aggregates. The results of the simulation study show good results, with gains of 10-40% in the QoS parameters. By simulation, the positive effects of the proposed routing and aggregation strategy and the usefulness of the algorithm. The proposed algorithm constitutes the central control logic, and the resulting control actions are realisable through the SDN/NFV architecture.

Dedication

To all my beloved family,

This thesis is dedicated for you.

Declaration of Authorship

I certify that the effort in this thesis has not previously been submitted for a degree nor has it been submitted as part of requirements for a degree. I also certify that the work in this thesis has been written by me. Any help that I have received in my research work and the preparation of the thesis itself has been duly acknowledged and referenced.

Signature of the Student,

[illegible]

Amalina Farhan Kamaruddin
December 2017, London.

Acknowledgements

Praise be to Allah SWT, His uncountable blessings; and best prayers and peace be upon His best Messenger, Prophet SAW, his pure descendants, his family and his noble companions.

First, I would like to express my deep and sincere gratitude to my Supervisors, Prof Hamed Al-Raweshidy and Prof Maozhen Li for their help and guidance during my studies. The continuous advice and support, also guidance during this research are very much appreciated. I appreciate the patience and encouragement of these professors who were always taking care of my research to keep progressing and finishing on time. This research would not have been done without their help and resourceful input.

Next, I would like to thank my treasured family, especially to my Dad, Mr Kamaruddin b Sa'ari, (who already returned to his Creator during my first year of study), Mom, Mrs Rasidah bt Hussain and my siblings for endless support and inspirations.

Last but not least, deeply love expression to my beloved husband, Mr Dilshod Muhammadiyev, who patiently accepted my busy works yet still understands me and give his fully supports and encouragement towards this PhD research.

Thank you all from the bottom of my heart.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ABR	Available Bit Rate
ACO	Ant Colony Optimisation
ATM	Asynchronous Transfer Mode
CAPEX	Capital Expenditure
CBR	Constant Bit rate
CGF	Cumulant Generating Function
CPU	Central Processing Unit
DSCP	Differentiated Services Code Point
DWDM	Dense Wavelength Division Multiplexing
DWT	Discrete Wavelet Transform
ECN	Explicit Congestion Notification
FBM	Fractional Brownian Motion
FCFS	First-Come First-Served
GI	General Independent
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
LRD	Long-Range Dependent
MAP	Markovian Additive Process
MGF	Moment Generation Function
MIPS	Millions of Instructions Per Second
MODWT	Maximum Overlap Discrete Wavelet Transform
MPLS	Multiprotocol Label Switching
NFV	Network Function Virtualisation
NP	Non-Deterministic Polynomial-Time
NRT-VBR	Non-Real-Time Variable Bit rate
OPEX	Operating Expenditure
Po	Poisson Process
QoS	Quality of Services

RAM	Random Access Memory
RED	Random Early Detection
RT-VBR	Real-Time Variable Bit Rate
SDH	Synchronous Digital Hierarchy
SDN	Software-Defined Networking
SLA	Service Level Agreement
TCP	Transmission Control Protocol
ToS	Type of Service
TSP	Traveling Salesman Problem
UDP	User Datagram Protocol
VM	Virtual Machine
VNF	Virtual Network Function
VR	Virtual Reality
WDM	Wavelength Division Multiplexing

Chapter 1

INTRODUCTION

1.1 Background and Context

5G networks are often characterised by a set of strict performance criteria. The reference to 5G networks is an example where architectures built on SDN/NFV and optical fibre transmission are expected. To cater for vastly diverse traffic and quality of service to various Use Cases, a network is required to have;

- Massive system capacity
- Very high data rates
- Low latency
- Extremely high reliability and availability
- Energy efficient and secure operation

A fairly new paradigm in communication networking is network *elasticity,* which can be interpreted as an optimal utilisation of resources and – as far as possible – a demand-driven allocation of capacity. The justifications for a flexible resource allocation are at least twofold. From a capacity point of view, resource pooling is always more efficient than distributed resources in term of utilisation. Secondly, dynamic functional allocations through Network Function Virtualisation (NFV) improves the resilience approaching that of a distributed logic. To achieve these goals cost effectively, mechanism to achieve high resource utilisation and resilience are imperative. Of fundamental important is a novel approach to congestion control, which must be able to cater for a wide range of traffic sources from applications such as massive sensor networks, telemedicine and virtual reality (VR). Furthermore, it is widely known that aggregation of many types of traffic [13][32][38][54] leads to self-similar behaviour, often leading to starvation of some traffic types by others. Another incentive is that user mobility has been tackled by system over-provisioning in traditional networks, which becomes inviable as traffic demands and mobility tends to increase further.

The present study addresses congestion control in communication networks, where the improvements in network performance and functional resilience of a virtual hierarchical congestion controller has been exploited. In this work, the meaning of congestion control is a set of functions having the purpose of ensuring service quality for diverse traffic types and demands with high robustness and flexibility. In principle, congestion control can be viewed from two different angles – distributed or centralised control. A distributed control, on the other hand, is very resilient allows fast decisions for packet processing and routing, but the logic is for the same reason based on simple policies and limited network status information. In the centralised view, assuming that the Software-Defined Networking (SDN) Controller possesses full network information and can therefore determine network optimal policies. A centralised function also ensures consistency and minimises control message overhead. The disadvantages of a single controlling function are its vulnerability to failure and communication latency caused by processing load or transmission delay.

On a node level, congestion control amounts to traffic aggregation and fast routing. Such controls have no or limited possibility to adapt to changing network conditions. It can be viewed as a heuristic, node centric approach that may result in network states far from the global optimum. On the network level, on the other hand, traffic routes can be assigned so that the *network-wide end-to-end performance* is optimised. A novel framework for congestion control is being proposed, taking advantage of both the centralised and the distributed views through functionality separation by means of SDN and NFV. The functionality is separated into computationally intensive determination of optimal traffic aggregation strategies and routing policies, traffic measurement, and reporting, and local resource scaling, traffic aggregation and routing. By adopting SDN and NFV, a hierarchical implementation on three levels to ensure both high resilience and resource utilisation is assumed.

The network optimal congestion control is implemented in three hierarchies (Figure 1.1) consisting of;

(1) **SDN Controller**, responsible for network monitoring, assignment of orchestrator virtual machines, and acts as direct interface to network managements systems;
(2) **SDN Orchestration**, responsible for global optimisation and policy creation, traffic collection, statistical analysis and network state reporting to the controller, as well as updating policies and coordinating routing tables on the router level;
(3) **Aggregation and routing**, responsible for real-time traffic aggregation and route selection in accordance with policies.

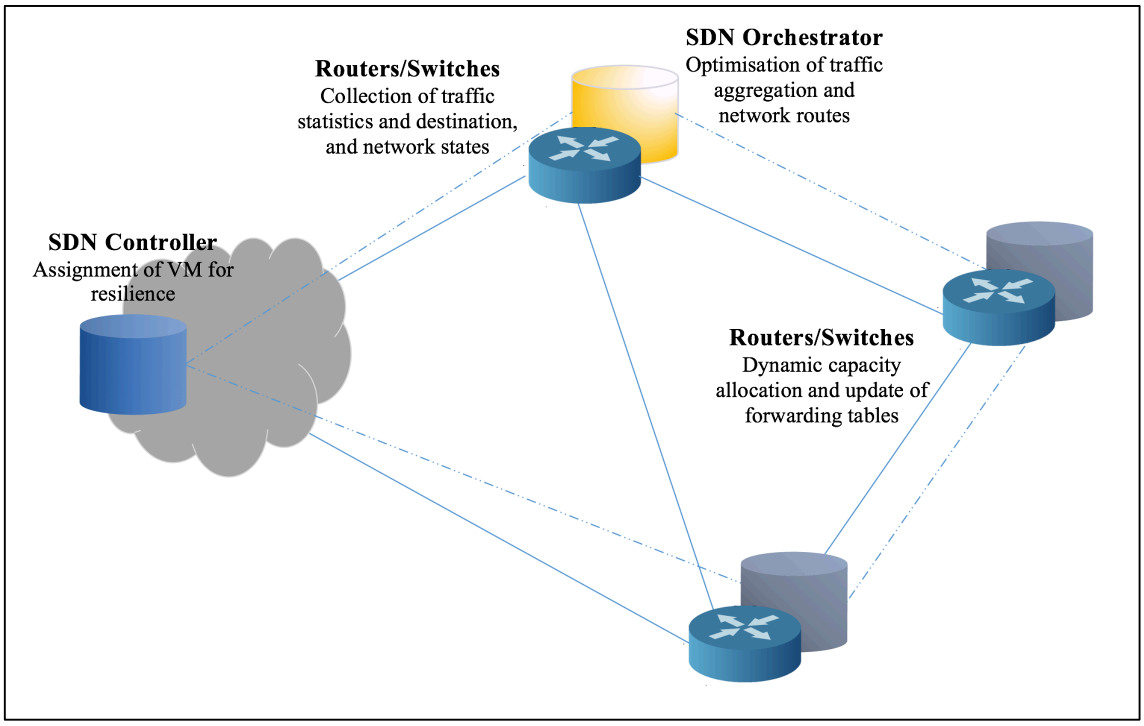

SDN Orchestrator
Optimisation of traffic aggregation and network routes
Routers/Switches
Collection of traffic statistics and destination, and network states
SDN Controller
Assignment of VM for resilience
Routers/Switches
Dynamic capacity allocation and update of forwarding tables

Figure 1.1: Schematic Architecture of Proposed Framework Model based on SDN/NFV Environment

This function separation allows reduction of control plane communication overhead, resilience on concentrator level and efficient use of resources. In effect, this framework aims at improving all five main requirements on 5G networks. Congestion control implemented using available resources more flexibly leads to a substantially improved quality of service. By showing in the simulation, dynamic traffic aggregation improves resource utilisation on individual router level, whereas optimal routing takes advantage of the router load levels to distribute the traffic throughput in the network, improving quality of service. The experimental set-up includes a simulator generating three and resource scaling on node level, and a network route optimiser. An optimal strategy of control function allocation is also presented.

1.2 Motivation

Motivated by the empirical success of the previous studies, literately discussed in Chapter 2, this thesis provides few motivations that have driven this research.

(1) Regarding the most common debate nowadays, centralised and separation network, SDN which promising the ability to handle real network programmable and more agile, to provide fast changing to the virtualisation of the resources in data centre and as a balancing as well to the cloud computing [51].
(2) There are some motivations concerning in developing or increasing the Quality of Service (QoS) issues in congestion control traffic aggregation.
(3) Another concern regarding the traffic aggregation issues in centralised control is static aggregation (in preliminary results) proves to have no better results than dynamic aggregation.
(4) Lastly, to proves of having higher energy consumption when assigning the Virtual Machine (VM) to another host with larger free capacity by keeping the allocation of network function in virtualisation.

1.3 Aim and Objectives

The aim of this thesis is to improve the resource utilisation which is throughput and QoS that represent delay and packet loss by the mean of network programmable SDN in congestion control. The research aim is adopted through the following objectives.

(1) The state of the arts of the previous research studies related to the problem of congestion control of heterogeneous traffic in a network has been studied. Traffic of equal intensities of Poisson process (Po), Markovian Additive Process (MAP) and fractional Brownian motion (fBm) are generated means processes with the same expected arrival rate. The variances and scaling properties under aggregation, however, are very different. Both the processing power and the buffer size can be scaled independently by the central control logic.

(2) SDN and NFV related works are being investigated so that it can be applied to the current research studies. The studied logic needs SDN/NFV to work, which means having the ability to configure nodes (switches) dynamically and in real time. This has been assumed as a condition.

(3) Designing and building a novel framework for congestion control, both in node level and network level through functionality separation by means SDN/NFV. The function separation allows reduction of control plane communication overhead, elasticity on concentrator level and effective use of resources.

(4) Simulating the SDN controlled traffic aggregation and optimal routing, both in terms of resource utilisation (throughput) and QoS represented by delay and packet loss so that the traffic can be controlled to take routes through the network to minimised the end-to-end delay.

(5) Implementing the allocation of network function based on Ant Colony Optimisation (ACO) using CloudSim simulation is to measure a cloud operator's cost by reducing the energy function.

1.4 Research Contributions

There are three research contributions that have been assumed as a hierarchical performance on three levels in this thesis which are summarised in the following:

(1) Proposing a novel framework for congestion control regarding the traffic aggregation in the node level. The development of the proposed framework is based on the logic that can be implemented on SDN where the intelligence and logic are embedded into the control function. Conceptually, this research is using entropy as a single traffic aggregation decision parameter as it is convenient due to some parsimonious characterisation of the traffic which is also suitable in a Big Data context, since it can be estimated from streaming data, which then would allow for truly real-time congestion control.

(2) Implementing an optimal routing for congestion control taking advantage of SDN technology in the network level and comparing the difference aggregation and routing strategies and showed (by simulation) that aggregation in combination

with min-max optimal routing gives substantial gain in all performance metrics, and was better than any other combination of studied actions.

(3) Discovering a strategy for on-line assignment policy of VM on hosts that minimises the energy (to reduce the cost effectively) in allocation of network functions using ACO. For the purpose of congestion control, the most important is to have a mechanism that ensures the resilience of the allocation of the controller, which is enabled through NFV.

1.5 Thesis Structure

This thesis contains 7 chapters which is related to SDN and NFV which briefly explained below:

⇒ Chapter 1: Introduction
This chapter introduces the research study briefly, the aim and objectives of the studies, motivations under this studies, some additional knowledge and the thesis outlines.

⇒ Chapter 2: Literature Review
In this chapter, the current state of arts being discussed including the technology of SDN and NFV, the previous research which related to traffic aggregation, traffic types like Po, MAP, and also fBm. This chapter also elaborates about ACO and the CloudSim simulation that had been used in this research. As for the current state of the literature, it is clear that there is scope for the development of an allocation of network function system on CloudSim that limits the amount of energy consumed while endeavouring to sphere the performance of applications.

⇒ Chapter 3: Traffic Aggregation for Congestion Control
Chapter 3 describes work on traffic aggregation which including three traffic types; Po, MAP, and fBm. In this chapter also explains the mathematical tools; wavelet analysis that analysing time series or images by decomposing them with respect to different scales.

- ⇒ Chapter 4: Optimal Routing for Congestion Control

  This chapter elaborates the workload management process, where traffic routes can be assigned so that the network-wide end-to-end performance is optimised.

- ⇒ Chapter 5: Simulations and Results

  This chapter is explaining the simulation for traffic aggregation and optimal routing and showing the results for both simulations.

- ⇒ Chapter 6: Allocation of Network Function using Ant Colony Optimisation

  This chapter evaluates the allocation of network function by comparing three policies for assigning VMs to hosts: round-robin, a customer greedy heuristic, and an optimised allocation policy derived from an ACO algorithm to measure a cost for cloud operator by using an energy function. The main objective for this chapter is to find an assignment policy that minimises this energy.

- ⇒ Chapter 7: Conclusion and Future Work

  The final chapter is discussing about the conclusion of this research and the future works that could expand for the next research to improve the current research.

Chapter 2

LITERATURE REVIEW

2.1 Introduction

Congestion in the network generally is described when the load on the network is greater than the capacity of the network can handle. Congestion control meanwhile, refers to mechanisms and techniques used to control congestion and keep the traffic below the capacity of the network. The investigation focuses on the effects of congestion control in a network subject to self-similar traffic implemented using SDN and NFV frameworks.

2.2 Distributed and Centralised Control

In principle, network control functions can be either distributed or centralised. At present time, network control is typically decentralised and residing in the routers due to its fast reaction time and resilience against failure. The control actions, however, are only based on network state information in a small neighbourhood of the node itself. Distributed control functions are also based on rather simple logic, which may not lead to flow control which is optimal for the entire network. Meanwhile, the advantage of a centralised control is the ability to apply a control policy taking the state of the entire network into account. The disadvantage is delay in receiving information updates and applying control actions, and the risk of overload or failure of a single centralised control function. With the introduction of SDN, control functions can be separated into fast flow control and routing functions, which are decentralised and implemented locally in routers and switches, and longer term control strategies residing in a network entity referred to as a SDN Orchestrator. Figure 2.1 illustrates the difference between distributed and centralised network.

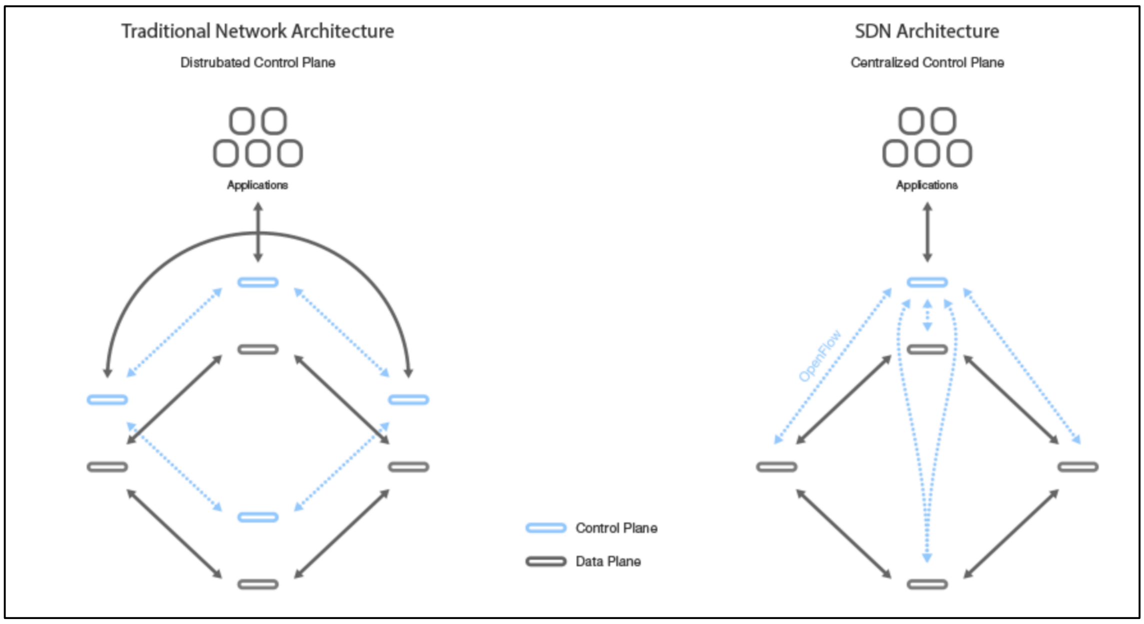

Traditional Network Architecture
Distrubated Control Plane
Applications
SDN Architecture
Centralized Control Plane
Applications
OpenFlow
Control Plane
Data Plane

Figure 2.1: Traditional Network vs SDN Architecture [52]

2.3 Software Defined Networking

The concept of SDN, which originated in the mid 1990s, is a framework to manage network behaviour dynamically via programmable applications using open interfaces to the network. This allows for a flexible utilisation of network resources in contrast to a static allocation of resources in traditional networks. In such networks, the control of a packet flow is realised by a node forwarding packets to the best destination according to the packet header information and a set of more or less static rules. In effect, routers try to accommodate the quality of service on a per flow basis, mostly agnostic of the states in other parts of the network. In this case, the control of the packet flow and the flow itself are fundamentally linked together. As a result, some parts of the network may be overloaded whereas others may be under-utilised.

SDN allows decoupling of the control from the flow, referred to as the control plane and the data plane, respectively. Figure 2.2 shows the SDN architecture. A consequence of this decoupling is the possibility to create resource and flow control logic that is based on more information that is contained in individual flows. The logic may thus depend on the state of different parts of the networks, the characteristics of various flows, or external factors (for example, dynamic reservation of resources). As a matter of fact, the separation between the control plane and the data plane was implemented for

Synchronous Digital Hierarchy (SDH) (connection-oriented) networks already in the mid 1990s. The implementation was facilitated by the fact that the routes in such networks already were semi-static, and the relative few nodes constituting the networks [15].

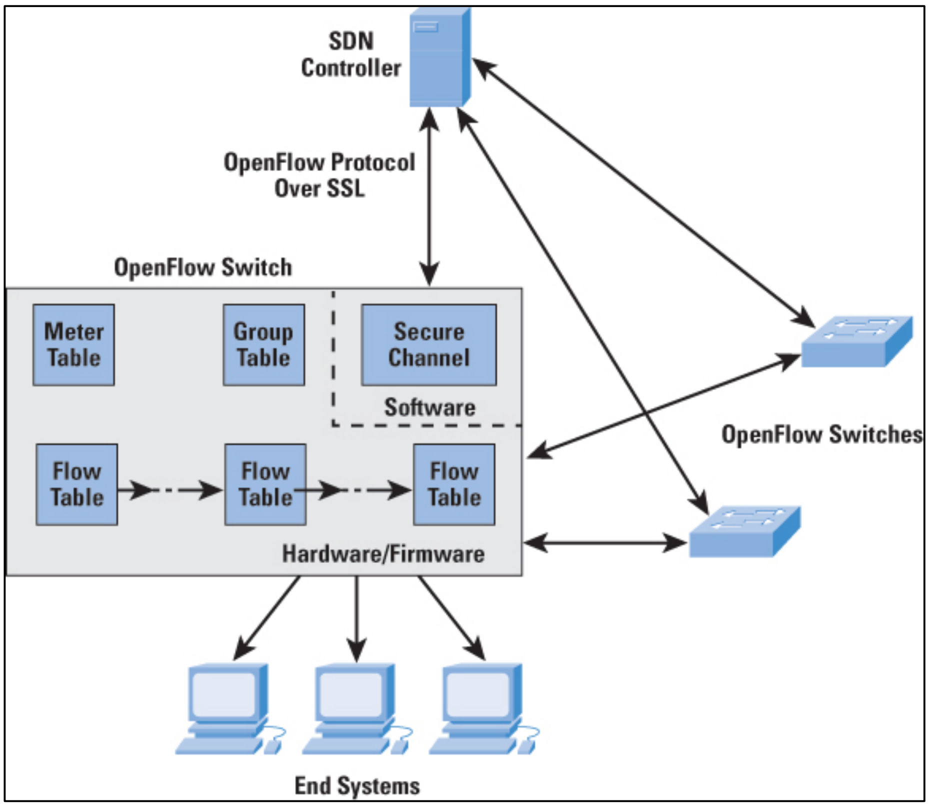


Figure 2.2: Software-Defined Network Architecture [49]

The architectural and structural properties of SDN are described in, for example, [3][27]. D’souza et al.[11] note that QoS is a relatively new feature in SDN. Using three QoS parameters: delay, packet loss and throughput, the delay is given by the average queue length in the system, while the packet loss is the proportion of lost packets when a queue is full to the total number of arriving packets and the throughput is the number of delivered packets to the number transmitted packets. The main difference between the delay and the throughput QoS is that delay is determined per queue, and the values are added to give a total end-to-end delay, whereas throughput is given by the de facto delivered packets and therefore is defined for an origin-destination pair or generalised for the whole network. There are several incentives for SDN. Firstly, traffic characteristics change much more rapidly today than the past through the fast development of new services. Secondly, traffic flows depend on an increasingly cloud-based service distribution. Thirdly, with rapidly increasingly traffic volumes and big data, resources need to be managed cost efficiently. The investigation is based on the capabilities of OpenFlow as part of the SDN/NFV framework, which ensures necessary resilience and scalability. Mills et al. [34] evaluate different heuristic strategies of allocation of virtual machines in clouds. Considered as a strategy based on initial placement – known as consolidation – where new requests are allocated subject to available resources and a new request may modify existing allocation to achieve lower cost, or a trade-off between Service Level Agreements (SLA) and cost. Several papers also consider optimisation with respect to geographically distributed clusters [22][34], and include network bandwidth as a system parameter. Another view is to divide requests into tasks, and schedule the tasks optimally [46]. Both consolidation and scheduling requires the possibility of migration or resizing of tasks. Lee et al. [30] propose using a vector of different resources to describe the state of the system state economically. The system variables can be stored in a central database for easy retrieval of cloud state data by the resource allocation function.

2.3.1 OpenFlow-enable Switches

In this research, network consists of OpenFlow-enabled switches and routers. The congestion control logic triggers queue settings and forwarding tables using OpenFlow. It should also be possible to collect traffic traces and destination addresses from the nodes. The queues should have configurable rates, which could be specified as a minimum/maximum rate pair [39]. Version 1.3 of OpenFlow supports slicing, that is, multiple configurable queues per port, which could be used in the implementation of this research congestion control. A physical switch or router typically has several queues which are served by a common processing unit with one or more processors. There is also a memory bank that can be used by the queues. The resources, processors and memory, can either be shared or partitioned between the queues. For memoryless traffic, sharing is the most efficient since it is fair. For traffic mixtures with long memory, however, resource sharing is not optimal, and one traffic type can exacerbate the performance of other traffic types. When resources are partitioned, the queues are isolated from each other and each queue protects its workload from the influence from the load in other queues. In this investigation, partitioning which is dynamically controlled, and the resources allocated depend on the actual load per traffic type is used. Similarly, the buffer sizes assigned to the queues should be configurable. In principle, it is sufficient to be able to specify a maximum buffer size per queue. This is particularly important to limit the impact of self-similar traffic on the rest of the network [38]. Figure 2.3 illustrates the OpenFlow-enabled switch architecture [50].

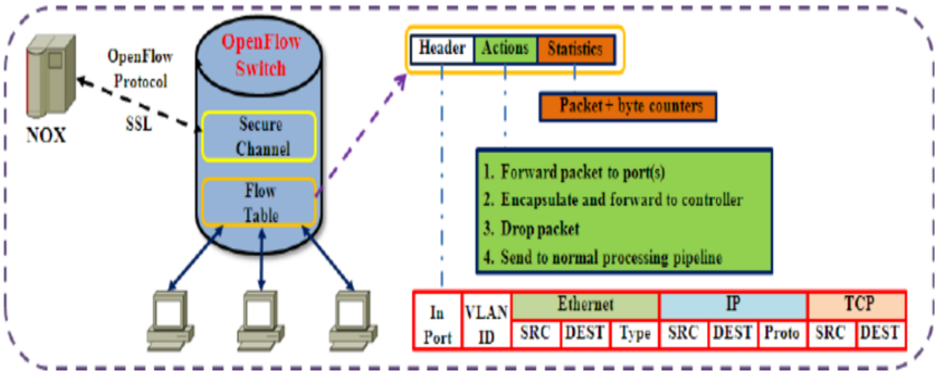


Figure 2.3: OpenFlow-enabled Switch

The update of packet forwarding tables in switches and routers is a standard operation, where the routes are determined by the central logic by assuming that traffic traces are available, either via OpenFlow or some other means. Full traces rather than statistics are required, since the entropy is determined based on traces and then used to control the aggregation. OpenFlow also implements the Orchestrator-Forwarder interface for manipulating flow tables, and requesting traffic statistics and network state information from the nodes to model the network topology and for load monitoring. For large networks, multiple orchestrators, and the OpenFlow is used to share information between these entities by classifying traffic efficiently, and ideally use lower-order protocol header fields to extract this information. Examples are traffic class header field in Multiprotocol Label Switching (MPLS), the type of service field in Internet Protocol version 4 (IPv4), traffic class in IPv6, and possibly the source IP address, source and/or destination ports.

2.4 Network Function Virtualisation

NFV is essentially decoupling of software from hardware. It provides a layer between the hardware with its operating system and software applications, which are known as hypervisors. Thus, NFV represents the physical implementation of functions as virtual machines or virtual servers on given hardware platforms. In the past, when traditional network functions (Network Address Translation, Load Balancer, Firewall) is mentioned, referred to functions by the physical devices. As this functions need to be programmed by hand to those devices, this procedure will create some issues when any novelty or updates in the system deployment is needed. Whereas, Virtualised Network Function (VNF) is a software-based and decouple from hardware. In VM cases, the software image of network function can be deployed. By being highly flexible and dynamic, these networks can decrease Capital Expenditure (CAPEX) and Operating Expenditure (OPEX) significantly [52]. The differences are shows in Figure 2.4 [53].

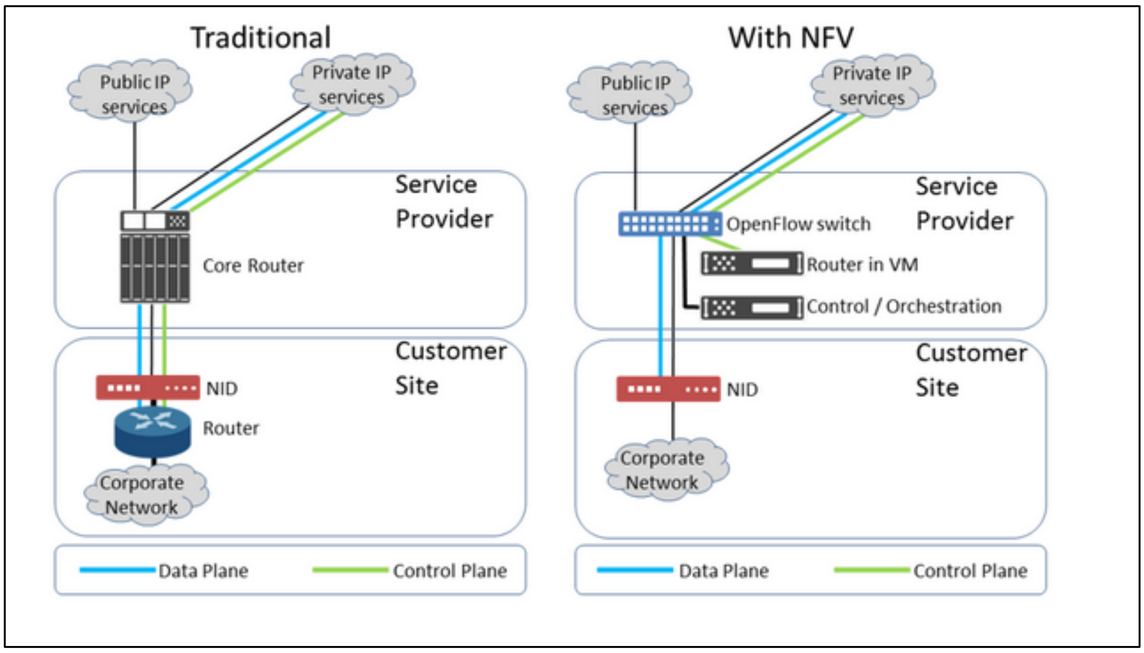
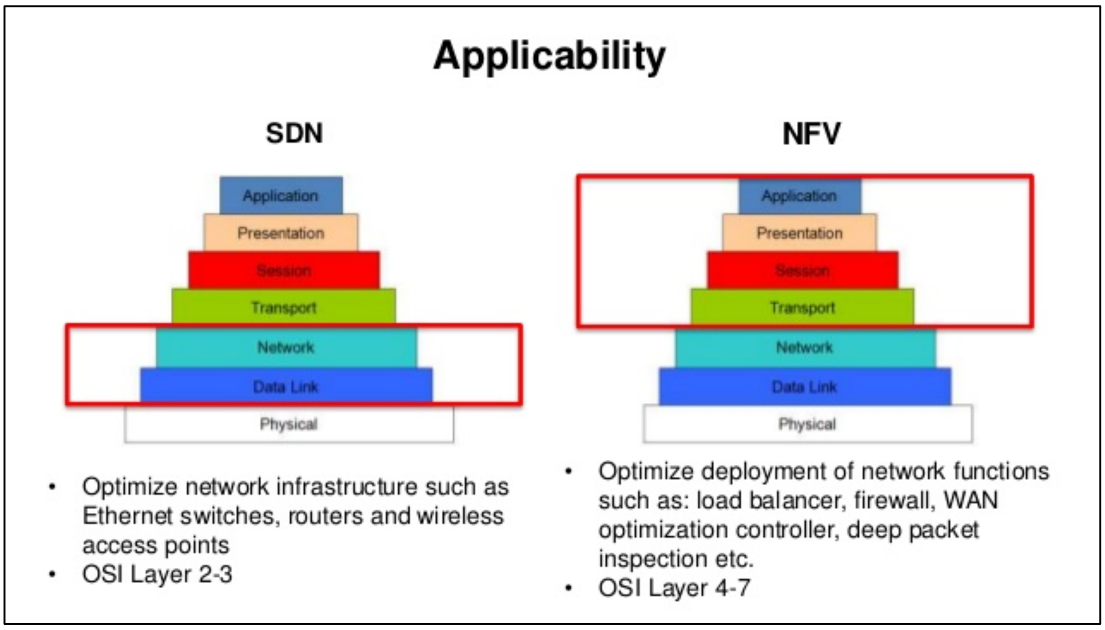


Figure 2.4: Differences between Traditional Network with NFV and without NFV

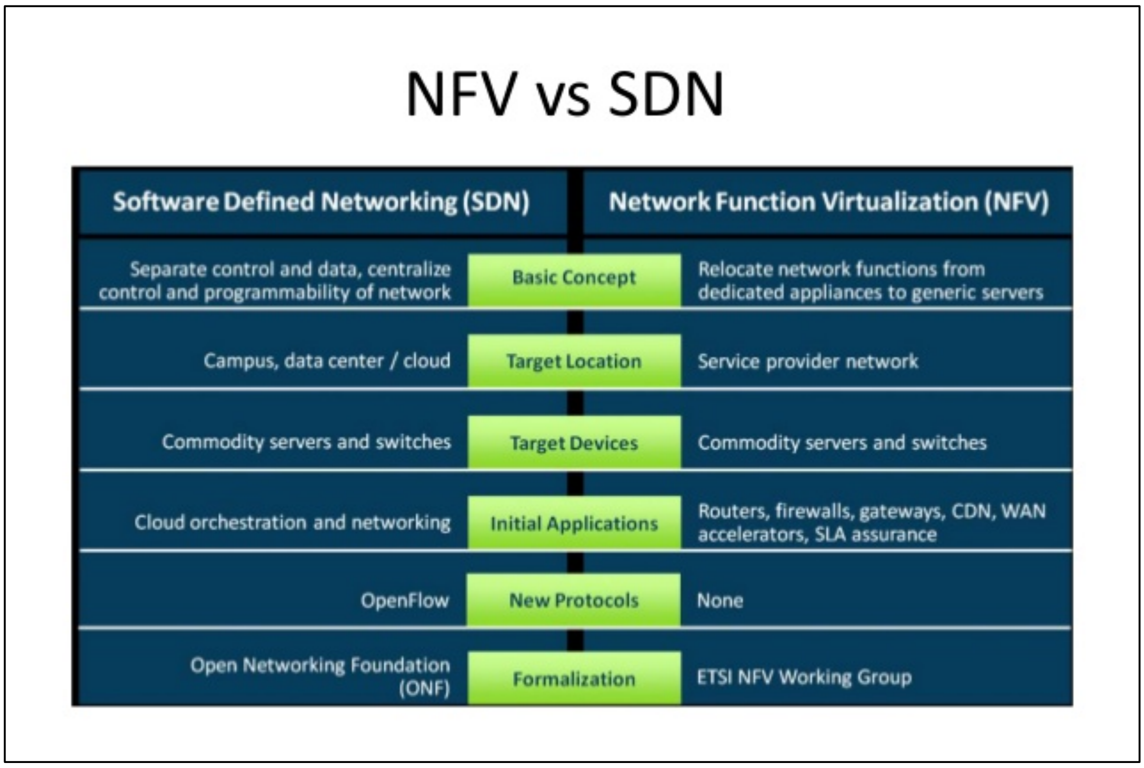
To distinguish between SDN and NFV, (see Figure 2.5 and Figure 2.6), note that SDN refers to the decoupling of network logic from data transport, whereas NFV is the rapid provisioning and control through virtualisation [27]. The resilience concerns of a centralised logic is addressed by NFV, that can be seen as resource sharing on platform level (Platform-as-a-Service). This approach allows for efficient scaling and utilisation of network resources, and a robust and resilient implementation of a centralised control logic. Furthermore, the SDN/NFV concept supports open interfaces, such as OpenFlow, reducing the implementational complexity of a centralised intelligence in complex networks.



- Optimize network infrastructure such as Ethernet switches, routers and wireless access points
- OSI Layer 2-3

- Optimize deployment of network functions such as: load balancer, firewall, WAN optimization controller, deep packet inspection etc.
- OSI Layer 4-7

Figure 2.5: SDN vs NFV in OSI Layer



NFV vs SDN

Software Defined Networking (SDN)		Network Function Virtualization (NFV)
Separate control and data, centralize control and programmability of network	Basic Concept	Relocate network functions from dedicated appliances to generic servers
Campus, data center / cloud	Target Location	Service provider network
Commodity servers and switches	Target Devices	Commodity servers and switches
Cloud orchestration and networking	Initial Applications	Routers, firewalls, gateways, CDN, WAN accelerators, SLA assurance
OpenFlow	New Protocols	None
Open Networking Foundation (ONF)	Formalization	ETSI NFV Working Group

Figure 2.6: Differences between SDN and NFV

In this research set-up, NFV is mainly used to ensure resilience of the control logic. Assumed that there are a number of nodes that can host the SDN orchestrator, without specifying exactly where or how, since the architecture is not the focus of this study. The VM allocation algorithm described in this paper is adapted to the initial placement problem for the reservation type of cost and allocation in a single cluster. Consolidation and scheduling are not considered. The algorithm as such, however, is sufficiently general to be extended to more general resource allocation problems. The optimisation uses the ACO, described in [10][17], where it is used on the travelling salesman problem (TSP) and bin packing problems. This research proposed strategy is to reformulated as a dynamic program for on-line optimisation.

2.5 Congestion Control

A number of congestion control mechanisms are investigated by Gerla and Kleinrock [19] and Jacobson and Karels [26]. The definition of congestion used as an over-demand of resources, leading to a performance decrease when compared to states with lower demand [23]. It is therefore imperative to assume that network resources are sufficient to cater for the offered traffic most of the time. Resource planning aspects are discussed in [48] for MPLS and [18] in access networks. In the latter, macroscopic traffic characteristics are generated by superposition of a number of simple Markovian on-off sources. The drawback with this method of traffic generation is that it is an asymptotic result, and the number of simple sources therefore need to be very large. Congestion control under self-similar traffic is also studied by Tuan and Park [45], Abbasi et al. [2] and others [37].

Most congestion control algorithms in the literature are flow-based in general and the Transmission Control Protocol (TCP) in particular. TCP is a feedback congestion control on flow-level where the transmission rate is based on a sliding window. Packets are acknowledged by the receiving side, and when congestion is detected by the sender not receiving an acknowledgement before a set time elapse, or receiving duplicate acknowledgement, it retransmits the unacknowledged packets [28]. In [45], the authors investigate the predictability of self-similar and how this can be used in congestion control to improve the throughput of TCP by proposing a feedback congestion control, using the throughput as a control variable, and adjusting the bandwidth from the client to maximise the throughput. In [11], the authors propose using Type of Service (ToS) and Differentiated Service Code Point (DSCP) to distinguish between traffic types, and can therefore be characterised as a priority-based control. Some authors highlight the need for more efficient detection of network congestion. Huang et al. [25] note that TCP is suboptimal for high-speed networks and suggest using free router capacity, ingress aggregate traffic and queue length as decision variables to make TCP converge faster and achieve fairness, and Haas and Winters [23] suggest probing for congestion with test packets. An alternative algorithm to Random Early Detection (RED) using Explicit Congestion Notification (ECN) is presented in [16], where packet loss and link utilisation are used, rather than queue length, to detect congestion. The result is a faster detection of congestion and more adequate rate adjustment to mitigate the congestion.

Using the method classification in [19], this research proposes a congestion control scheme which works on hop level and measures flow quality of service, disregarding admission control and transport protocol functions. The effect of end-to-end performance is studied – delay, packet loss and throughput – by dynamic traffic aggregation at the nodes and optimal routing with respect to delay. A theoretical investigation of feedback congestion control strategies can be found in [44]. It discusses the differences between feedback from aggregate traffic and individual flows. It is also shown, that the transmission rate resulting from a feedback congestion control can be expressed as a fixed-point equation, a technique used in this approach to determine optimal routes. Fixed-point equations have also been studied in simulation contexts for TCP controlled networks [21]. It should be noted, however, that there may be more than one stable solution, and care must be taken to avoid oscillatory network behaviour. This type of network instability effects has been studied in loss networks as well [20].

2.6 Traffic Aggregation

The potential advantages of centralised routing as compared to distributed routing in terms of QoS is studied in [27][35]. The presence of long-range dependence and self-similarity in data networks has been thoroughly established and studied [31][7][8]. Self-similarity and long-range dependence may be caused by heavy-tailed file size distributions, aggregation of a large number of short-memory traffic sources [38], certain protocols TCP or human interaction [7][8]. Modelling of bursty and self-similarity traffic types have also been studied extensively [38][4][29][2][1]. Smooth traffic is modelled by a Po, which is the traditional model for plain telephony. Many models have been suggested for video traffic [40][42][36]. We have chosen a short-memory process - typically modelled by a Markov modulated process - as proposed in [5][12]. The chosen model is referred to as a MAP and self-similar traffic is modelled by a fBm [32][31][4][2]. This process is modified to reflect packet generation by assuming positively correlated arrivals and selecting sample paths (reflected at $X(t) = 0$) so that $X(t) \geq 0$ for all t, since traffic load must be positive. Wavelets have shown to be very useful in analysis of self-similar traffic [42][41]. We use wavelet multi-resolution analysis to visualise composite traffic.

In [9], Dolzer et al. investigate various traffic aggregation strategies, and conclude that aggregation of audio and video sources reduce the need for resources, indicating that aggregation into two traffic classes – denoted real-time and non real-time traffic – gives the best result. In a SDN context, Nguyen et al. [35] optimise the assignment of forwarding rules to nodes with limited memory in the network, with the objective to maximise delivered traffic. This approach, however, does not cater for dynamic traffic aggregation at the nodes or optimised routes with respect to a network wide QoS measure. Wallner and Cannistra [47] use the ToS or DSCP fields in the IP header to map traffic to a queue port with properly set queueing parameters, and in [43], it is shown that a dynamic allocation of buffer size lead to an increase in throughput.

Egilmez et al. [14] analyse dynamic routing of multimedia flows. By avoiding using resource reservation or prioritisation due to the adverse effects such QoS provisioning methods may have on flows not subjected to QoS, such as packet lost and latency. The authors propose traffic classification based on the traffic header field in MPLS, the ToS field in IPv4, and possibly source IP address, source and/or destination ports. In contrast to the approach in [14], this research proposes a much simpler optimisation method, somewhat trading speed for accuracy, since forecasting future load based on past load by necessity is associated with prediction errors.

As for algorithm, entropy is being used as a traffic descriptor. Classification of traffic using entropy to identify traffic patterns is discussed in [6][13][24][45]. In [24], the authors identify traffic clusters for aggregation using a priority parameter. The clustering is based on a parameter vector, including source and destination IP addresses. Two of the objectives for traffic clustering is to identify abnormal traffic and identifying the largest flows. Thus, the algorithm is on flow level, whereas in this research, the study is focused on the effect of aggregation with respect to the overall offered traffic intensity. In [13], the authors suggest that the entropy can be used to estimate asymptotic properties of network queues, such as packet loss and delay. In this simulation approach, the determination of these properties are directly from the queues, as differences between queues are important in order to determine optimal routes, and these figures therefore need to be more exact. It can be shown that entropy is closely related to the scaled Cumulant Generation Function (CGF) in large deviations theory [6][33], a technique used for queueing analysis in Asynchronous Transfer Mode (ATM) networks.

2.7 Chapter Summary

Network traffic is expected to be heterogeneous, showing dissimilarity on various time scales. It has been extensively acknowledged that many traffic types are self-similar (long-range dependent), which may lead to starvation of traffic with shorter memory, just as prioritising of short-memory traffic may normalise long-range dependent traffic. Even if the control is implicit to be much slower than the traffic processing speed, the analysis of existing traffic and network load should be miserly and fast enough to capture dissimilarities on, say, time scales of seconds or fractions of seconds.

Chapter 3

TRAFFIC AGGREGATION FOR CONGESTION CONTROL

3.1 Introduction

The requirements on the service and network capabilities are dependent on traffic type. Traditionally, traffic has been classified in real-time and non real-time types of services, where real-time services tend to be more delay sensitive, whereas non real-time services are more sensitive to packet loss. A typical distinction in IP are services using User Datagram Protocol (UDP) and TCP, respectively. In ATM, this roughly correspond to Constant bit rate (CBR) or Real-Time Variable Bit Rate (RT-VBR) and Non-Real-Time Variable Bit Rate (NRT-VBR) or Available bit rate (ABR) services.

To describe the vast range of communication services available today, many having different characteristics, many models have been suggested in the literature. In the simulations, three traffic types are used – smooth, bursty, and long-range dependent traffic. Rather than trying to model traffic directly, this thesis tries to approximate traffic aggregates by mixing stochastic processes with different characteristics. The aim is to approximate aggregates by different proportions of the constituent stochastic processes. These roughly correspond to the services telephony (audio), streaming video, and Internet traffic. These three traffic types can for be described on a high level by three quantities – intensity, burstiness, and autocorrelation. Traditional plain voice traffic is often modelled as arriving according to a Po with exponential duration. Such traffic is also known as Markovian, or memoryless. It is well known that this model is inappropriate to model bursty video traffic or data traffic exhibiting a non-negligible memory. Therefore, the use of MAP is to model bursty traffic, and a fBm is to represent aggregate Internet traffic. These models are described below.

The main characteristic of traffic is its *intensity*, which is defined as

$$\rho = \frac{\lambda}{\mu} \ , \tag{3.1}$$

where λ is the number of packet arrivals per time unit and μ^{-1} is the mean processing time of a packet, so that μ is the average number of processed jobs per time unit. Traffic intensity is a measure of the average occupancy of a server or resource during a specified period of time. This is defined by ITU-D [54]. The traffic intensity represents the mean number of requests in progress per time unit, or equivalently, the work load on in a queue. The intensity is used to describe traffic at any point in the network, that is, offered traffic, carried traffic and blocked traffic (as a percentage of offered traffic). It should be pointed out, however, that for traffic types with long memory, this quantity is difficult to determine.

Bursty traffic can be characterised by its *peakedness factor*, defined as

$$\mathrm{Z} = \frac{\mathrm{Var(X)}}{\mathbf{E}(\mathrm{X})} \, . \tag{3.2}$$

For a Poisson process that $\mathbf{E}(\mathrm{X}) = \lambda = \mathrm{Var(X)}$, so its peakedness factor is $\mathrm{Z} = 1.$ In contrast, bursty traffic has a peakedness $\mathrm{Z} > 1.$ It is also useful to define the autocorrelation of a process X as

$$\rho(i,j) = \gamma(i,j)/\sigma^2, \tag{3.3}$$

where

$$\gamma(i,j) = \boldsymbol{E}\left((X_i - \lambda)(X_j - \lambda)\right), \tag{3.4}$$

where X_i and X_j are observations (number of packets) at times i and j, respectively, on some time scale. The quantity λ is the average arrival rate of the process.

For Poisson (memoryless) traffic, $\gamma(i,j)$ is negligible for $i \neq j$. In fact, this property defines lack of memory. Short memory processes have non-zero correlation on lags $i \neq j$ which tends to decrease exponentially – at least asymptotically. For long-memory processes, however, the correlation decreases slower than exponentially and can be pronounced on very long time lags. This has the effect of relatively long periods of low or high arrival rates, which has a large impact on network performance.

3.2 Entropy

Traffic models have different statistical properties and are described by different sets of parameters. This make them hard to compare with traditional statistical methods. For example, the mean and variance of long-range dependent traffic is difficult to determine on short time scales. A different approach is to use the entropy of the traffic as a single parameter characteristic. The entropy measures the "randomness" of an arrival process, so it can be interpreted as a measure of the memory and the behaviour of a given traffic trace. The entropy $H(X)$ of a stochastic process X is defined as

$$H(X) = -\sum_{i=1}^{n} p_i \, log \, p_i, \tag{3.5}$$

where n is the number of states and p_i is the probability of the process attaining state i. The states are defined by the number of packets arriving during a time interval corresponding to two consecutive sampling times.

The higher the entropy, the more random the process is. Therefore, this thesis expects the entropy of the Po to be the highest, which also turns out to be the case. On average, the MAP traffic is close to the Poisson traffic, but it has higher variance. The fBm traffic has significantly lower entropy on average, but much larger variance. The variability in entropy allows fast characterisation of traffic and decision of traffic aggregation so that the entropy of the aggregated traffic has lower variance than the fBm traffic itself. Along with appropriate resource scaling of the router queues, we show that the QoS and/or throughput is improved. Since entropy is not additive, direct scaling is not recommended. Both an estimation of mean and of high frequency variability are needed to estimate the scaling parameters. During investigation, this thesis uses the resource scaling factor $1.75H_a$ (where H_a is the entropy of the traffic aggregate) to the entropy H_t of the remaining traffic. The value is chosen based on empirical investigation. In principle, say that the aggregate consists of two traffic streams, one with higher entropy than the other. Then the total entropy is < 2 times the highest entropy - since the traffic with highest entropy dominates the entropy of the aggregate. 1.75 is chosen as an approximation to 2 times the mean entropy of the two streams.

3.3 Traffic Aggregation

Network traffic is assumed to be heterogeneous, exhibiting variation on various time scales. It has been widely accepted that many traffic types are self-similar (long-range dependent), which may lead to starvation of traffic with shorter memory, just as prioritisation of short-memory traffic may regulate long-range dependent traffic. Since long-range dependent traffic may cause congestion where resources are allocated statically, it might be possible to improve the network performance by allocating resources dynamically based on the characteristics of the traffic. In fact, the long range dependence of traffic means that the load it induces can be predicted using its autocorrelation structure. The predicted load may serve as a control variable for dynamic traffic aggregation, which must be performed on much longer time scales than, for example, routing table lookups.

Even if the control is assumed to be much slower than the traffic processing speed, the analysis of offered traffic and network load should be parsimonious and fast enough to capture variations on, say, time scales of seconds or fractions of seconds by suggesting using entropy of the traffic traces and the resulting aggregation strategies as a single measure of their randomness. Given the three traffic types and their different characteristics, the focus is in the most efficient way to aggregate the traffic and map it onto available resources and use as efficiency measure the statistical performance in throughput, packet loss and delay.

Firstly, it is instructive to look at the scaling properties of these processes and define the time aggregated traffic as the average of a time block of size m, so that

$$X_t^{(m)} = \frac{1}{m}\left(X_{tm-m+1} + \cdots + X_{tm}\right). \tag{3.6}$$

When looking at the sample mean $\overline{X}$ of a traffic process X (where $\overline{X}$ is approximating λ in Equation (3.1) above), a standard result in statistics is that the variance of $\overline{X}$ decreases linearly with sample size. That is, if X_1, X_2,..., X_n represent instantaneous traffic with $\lambda = \mathbf{E}(X_i)$ and variance $\sigma^2 = \mathrm{Var}(X_i) = \mathbf{E}((X_i - \lambda)^2)$, then the variance of $\overline{X} = n^{-1}\sum_{i=1}^{n} X_i$ equals

$$\mathrm{Var}(\overline{X}) = \sigma^2 n^{-1}. \tag{3.7}$$

For the sample mean $\overline{X}$, having for large samples that

$$\lambda \in \left[\overline{X} \pm z_{\alpha/2} s . n^{-1/2}\right], \tag{3.8}$$

where $z_{\alpha/2}$ is the upper $(1-\alpha/2)$ quantile of the standard normal distribution which appears only when constructing confidence intervals for the parameter and $s^2 = (n-1)^{-1}\sum_{i=1}^{n}\left(X_i - \overline{X}\right)^2$ is the sample variance estimating σ^2. The condition under which Equation (3.7) and Equation (3.8) hold are that;

(1) The process mean $\lambda = \boldsymbol{E}(X_i)$ exists and is finite,

(2) The process variance $\sigma^2 = Var(X_i)$ exists and is finite,

(3) The observations $X_1, X_2, \ldots, X_n$ are uncorrelated, that is,

$$\rho(i,j) = 0, \quad \text{for} \quad i \neq j. \tag{3.9}$$

This thesis assumes that conditions (1) and (2) always hold, but not necessary condition (3).

3.3.1 Smooth Traffic

Smooth traffic is traditionally modelled by a Po. The Poisson process, also called Markovian or *memoryless*, aggregates in time so that the variance to the mean decreases fast. The average magnitude tends to a well-defined limit (that is, the aggregate variance tends to zero) as m increases. This produces a smoothing effect in time, which is very advantageous from a performance point of view. Thus, only one parameter – the intensity – is required to characterise a Poisson process, and the same parameter determines the server capacities necessary for a given performance.

In contrast, self-similar traffic does not exhibit such behaviour [38][4]. Time aggregates does not smooth such processes rapidly, and the large variations can be experienced even on large time scales. This phenomenon motivates an adaptive traffic aggregation that allocates resources in proportion to some measure of the load on a particular time scale.

3.3.1.1 The Poisson Process (Po)

The traditional traffic arrival model is the Po, which can be derived in a straightforward manner. Note that the Po is a discrete process (for example, the number of packets) in continuous time. Fixing a time t and looking ahead a short time $t + h$, a packet may or may not arrive in the interval $(t, t + h]$. If h is small enough and the packet arrivals are independent, then the probability of a packet arrival in this interval can be assumed to be approximately proportional to length of the interval, h [54]. The probability of two or more arrivals in this interval can be considered negligible by defining a Po as follows.

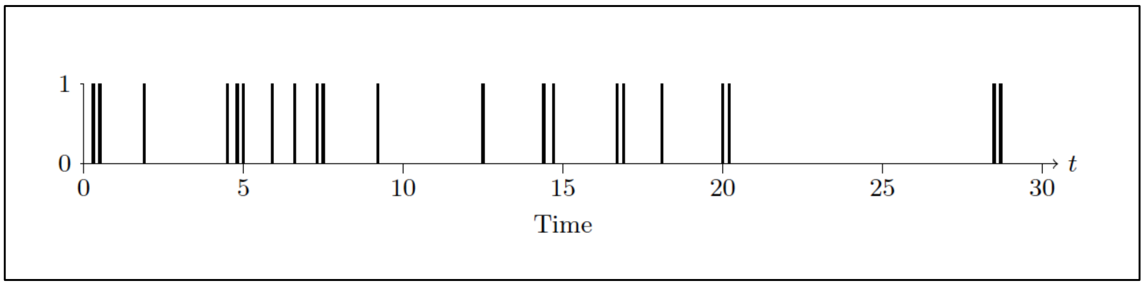


Figure 3.1: A Poisson call arrival process [29]

Definition 3.3.1.1. A Poisson process with *intensity* λ *is a process* $X = \{N(t) : t \geq 0\}$ *taking values in* $S = \{0,1,2,\ldots\}$ *such that*

(1)

$$N(0) = 0, and\ if\ s < t\ then\ N(s) \leq N(t),$$

(2)

$$\boldsymbol{P}(N(t+h) = n + m | N(t) = n = \begin{cases} \lambda h + o(h) & if\ m = 1 \\ o(h) & if\ m > 1 \\ 1 - \lambda h + o(h) & if\ m = 0 \end{cases}$$

(3) *if* $s < t$ *then the number* $N(t) - N(s)$ *of arrivals in the interval* $(s, t]$ *is independent of the times of arrivals during* $[0, s]$.

An example of packets arriving according to a Poisson process are illustrated in Figure 3.1, showing arrivals in continuous time. In discrete-time, this thesis is concerned with the arrivals in a time interval $(t, t+h]$, which can be described by the counting process related to the Poisson process, where $N(t)$ represents the number of arrivals of a process $X(t)$ up to time t.

Theorem 3.3.1.1. *The counting process $N(t)$ has the Poisson distribution with parameter λt that is,*

$$\boldsymbol{P}(N(t) = j) = \frac{(\lambda t)^j}{j!} e^{\lambda t}, \qquad j = 0, 1, \dots. \tag{3.10}$$

By substituting t for h, the length of a discrete-time interval, $N(h)$ is the number of arrivals in each interval. This is the way Poisson traffic is generated in the simulations. Note that the Poisson process is one of the simplest continuous-time Markov process, which means that it is memory-less, which follows from condition (3) in the definition. Figure 3.2 shows the scaling of a Poisson process, where the time interval becomes longer as in Equation (3.6) with a scaling factor $m = 8$ in two consecutive steps.

3.3.2 Bursty Traffic

It has been shown that video sources exhibit various degree of burstiness and autocorrelation [40][40][42]. The autocorrelation can be interpreted as *short-range memory.* This is partly explained by the nature of most movies and their coding. In its original form, a movie consists of *frames* that change in rapid succession. Within a movie, it is common that scenes do not change with high frequency. To save bandwidth, video coding therefore utilises this fact and encode *changes* to a reference frame. When a scene changes, more data is needed to set up a new scene, whereas within the same scene, only updates with lower bandwidth need to be transmitted. The size of these updates also depend on the type of scene.

Modelling of video traffic is complicated by a number of possible encodings, compression, and source type, such as movies or video conferences. Simply refer to bursty traffic as video traffic, without any claim that the chosen model MAP is suitable for all video sources. Also note that interactive multimedia, such as video conferencing, is more sensitive to delay, whereas streaming video is more sensitive to delay variation (jitter) [14]. By using delay as the optimisation target, the results may be considered more relevant for the former type of video source.

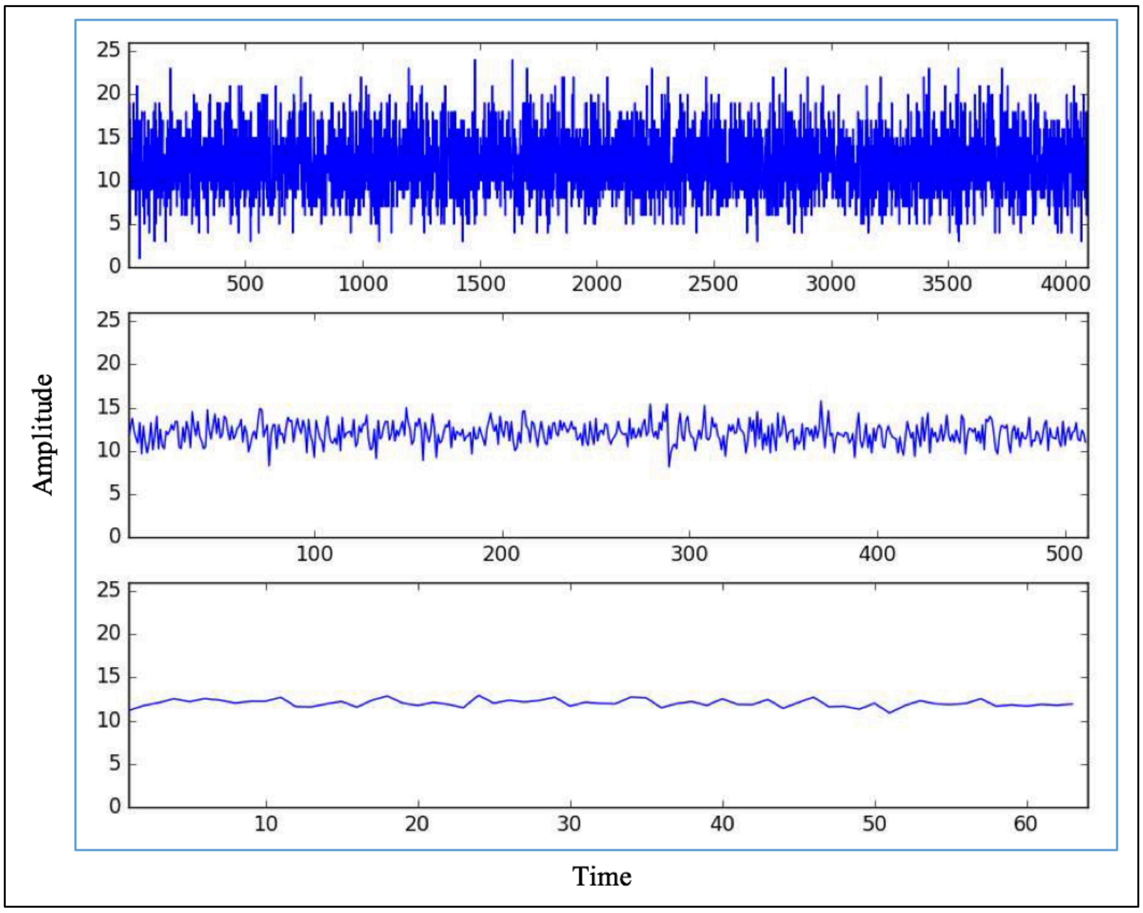


Figure 3.2: Aggregation of Poisson traffic

3.3.2.1 The Markovian Additive Process (MAP)

The MAP has been suggested as a model for traffic types exhibiting burstiness and autocorrelation [5][12][37]. The process is generated by creating a jump between two states (active and silent) that is controlled by a Markov chain. A Markov chain is represented by a matrix where each row sums to unity (probability matrix). The simulation of Markov chains is discussed in the cited reference [56]. The parameters are set so that the probability of remaining in either state is high if the process was in this state immediately before. This generates a bursty arrivals process with autocorrelation (memory), contrary to the arrivals following a Poisson process.

To define the MAP, assume that there are N independent traffic sources which are controlled by the same Markov chain. The chain jumps between the active and the silent states, which is represented by $S = \{0,1\}$, with zero representing the silent state and one the active state. The Markov chain is defined by the transition probabilities between the silent and the active states

$$a = \boldsymbol{P}(X_t = 1 | X_{t-1} = 0) \tag{3.11}$$

$$d = \boldsymbol{P}(X_t = 0 | X_{t-1} = 1). \tag{3.12}$$

The Markov chain can now be expressed as

$$M = \begin{pmatrix} 1-a & a \\ d & 1-d \end{pmatrix}, \tag{3.13}$$

where the steady state probabilities are $\pi_0 = \frac{d}{a+d}$ and $\pi_1 = \frac{a}{a+d}$, respectively. The smaller the parameters a and d are, the burstier the traffic generated by the model. As defined above, the MAP is a discrete time model, which makes simulation straightforward. Note that the lengths of a sources remaining in the state is geometrically distributed. Thus, the mean duration of an active period is $\sum_{n=1}^{\infty} dn(1-d)^{n-1} = 1/d$ time units, and the mean duration of a silence period is $1/a$ time units.

When fed to a queue with service capacity s, a stability criterion is required to have a limited stationary queue length, that is

$$\frac{a}{a+d} < \frac{s}{N}. \tag{3.14}$$

Figure 3.3 shows the scaling of a MAP as in Equation (3.6) with a scaling factor $m = 8$ in two steps. Although bursty, the MAP is still Markovian, which means that even if packet arrivals are correlated, the change of states of the source is memoryless. The aggregation of a MAP is shown in Figure 3.3. It scales similarly to the Poisson process (Figure 3.2), but converges slightly slower to its process mean.

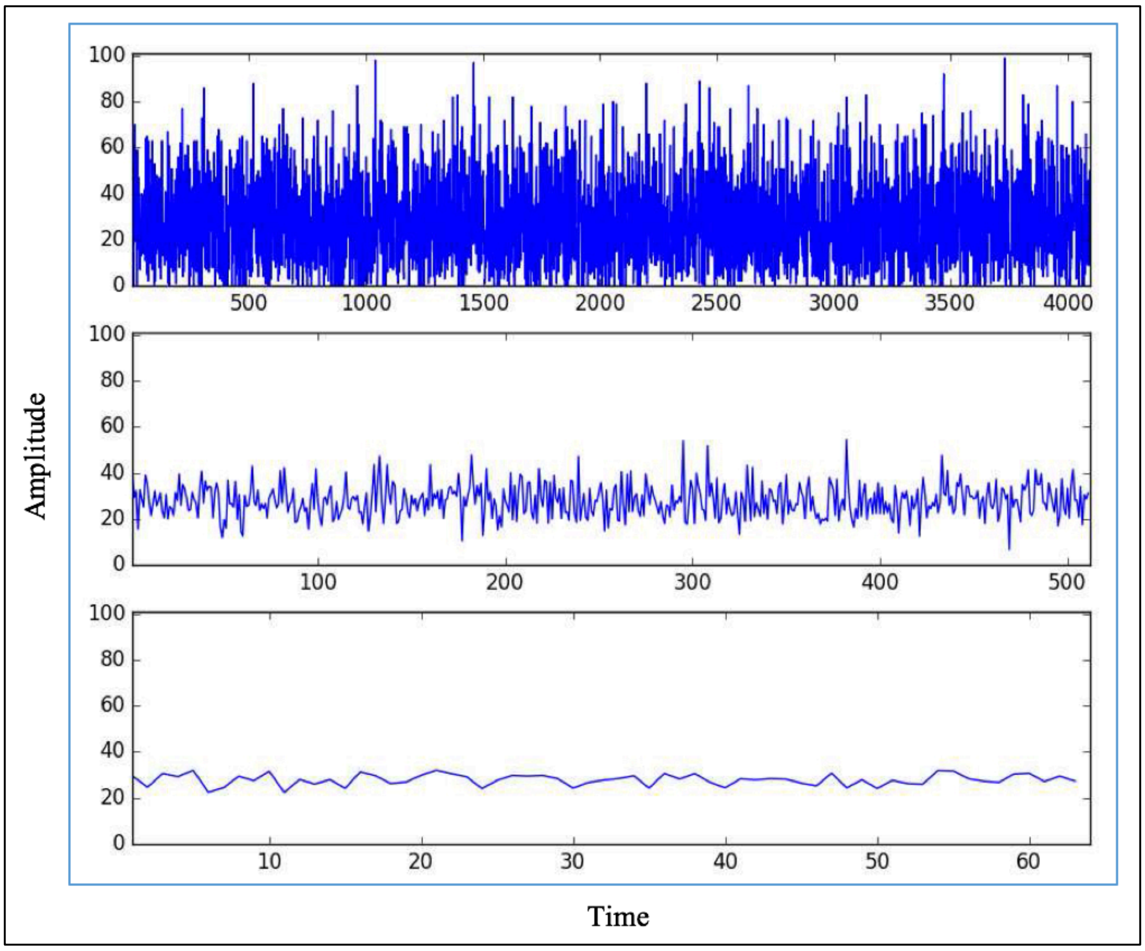


Figure 3.3: Aggregation of MAP traffic.

3.3.3 Long-Range Dependent Traffic

It may happen that the autocorrelation (3.3) decay very slowly, so that

$$\sum_{k=-\infty}^{\infty} \rho(k) = \infty, \tag{3.15}$$

This is the case when

$$\rho(k) \approx C_1 |k|^{-\alpha} \text{ as the time lag } k \to \infty, \tag{3.16}$$

where $\alpha \in (0, 1)$ and $C_1 > 0$ is a constant, that is, the autocorrelation decays according to a power law distribution. A process for which (3.16) holds is known as a long-range dependent process, or a long memory process. The aggregation of such processes is such that

$$Var\left(X^{(m)}\right) = \sigma^2 m^{-\alpha}, \tag{3.17}$$

with $0 < \alpha < 1$, that is, the aggregate converges to the sample mean slower than for a short-memory process.

In contrast, for Markovian and short-memory processes, the autocorrelation is bounded as

$$|\rho(k)| \leq b.a^k, \tag{3.18}$$

where $0 < b < \infty$ and $0 < a < 1$ are positive constants and k is the time lag. It follows that the sum of autocorrelation

$$\sum_{k=-\infty}^{\infty} \rho(k) = C_2 < \infty \tag{3.19}$$

is finite.

Long-range dependent traffic shows significant dependence in time on long time scales. This type of traffic is generated by superimposing a large number of short-memory processes or by certain flow control, such as the bandwidth of TCP-controlled traffic [31]. It should be noted that long-range dependence is difficult to ascertain, since it requires determination of how correlations converge to zero, and therefore needs investigation of traffic measured during long time intervals. Long-range dependent processes have the interesting property that can be predicted with better accuracy than short-memory processes. The stronger dependence of an observation X_t and past values $X_t - 1, X_t - 2, \ldots$, the more certain a future value $X_t + h$ close to past values is likely to be. This fact is used by Tuan and Park [48] for predictive congestion control.

Long-range dependence is closely related to the concept of self-similarity, where the latter describes the scaling properties of a process. The degree of dependence of the increments $X(t) = Y(t) - Y(t-1)$ of a process is specified by the *Hurst parameter H*, where

$$H \in (0, 1). \tag{3.20}$$

If $H = \frac{1}{2}$, the process is independent (or memoryless). When $H > \frac{1}{2}$, the process is positively correlated, and when $H < \frac{1}{2}$, it is negatively correlated. Assume that the process is positively correlated, so that $\frac{1}{2} < H < 1$. This follows from the nature of Internet protocols as well as, for example, human browsing behaviour.

A cumulative process in continuous time $Y(t)$ is self-similar with Hurst parameter H if

$$Y(at) = |a|^H Y(t), \tag{3.21}$$

for all $a > 0$ and $t \geq 0$. Long-range dependence and self-similarity are not equivalent in general. However, when $\frac{1}{2} < H < 1$, self-similarity implies long-range dependence and vice versa. In this case, the autocorrelation function expressed in the Hurst parameter H is

$$\rho(k) \approx H(2H-1)k^{2H-2}, \tag{3.22}$$

and $\rho(k)$ behaves asymptotically behaves as (3.16) with $\alpha = 2 - 2H$.

3.3.3.1 Fractional Brownian Motion (FBM)

A parsimonious model for self-similar, long-range dependent traffic is the *fBm,* $B_H(t)$, defined on an interval $(0, T)$. The fBm is a process such that

1. $B_H(t)$ is Gaussian on $t \in (0, T)$,
2. The process starts from zero, that is $B_H(0) = 0$ almost surely,
3. $B_H(t)$ has stationary increments,
4. The expectation is $\boldsymbol{E}(B(t) - B(s)) = 0$ for any $s, t \in (0, T)$,
5. The autocovariance of $B_H(t)$ is

$$\boldsymbol{E}\big(B_H(t)B_H(s)\big) = \tfrac{1}{2}\left(|t|^{2H} + |s|^{2H} - |t-s|^{2H}\right) \text{ for any } s, t \in (0, T). \tag{3.23}$$

The process is self-similar and long-range dependent for $H > \frac{1}{2}$.

Figure 3.4 shows the aggregation of an fBm subject to a scaling factor of $m = 8$. The aggregate does not converge to any well-defined mean value, but a noticeable variability persists for the aggregate traffic.

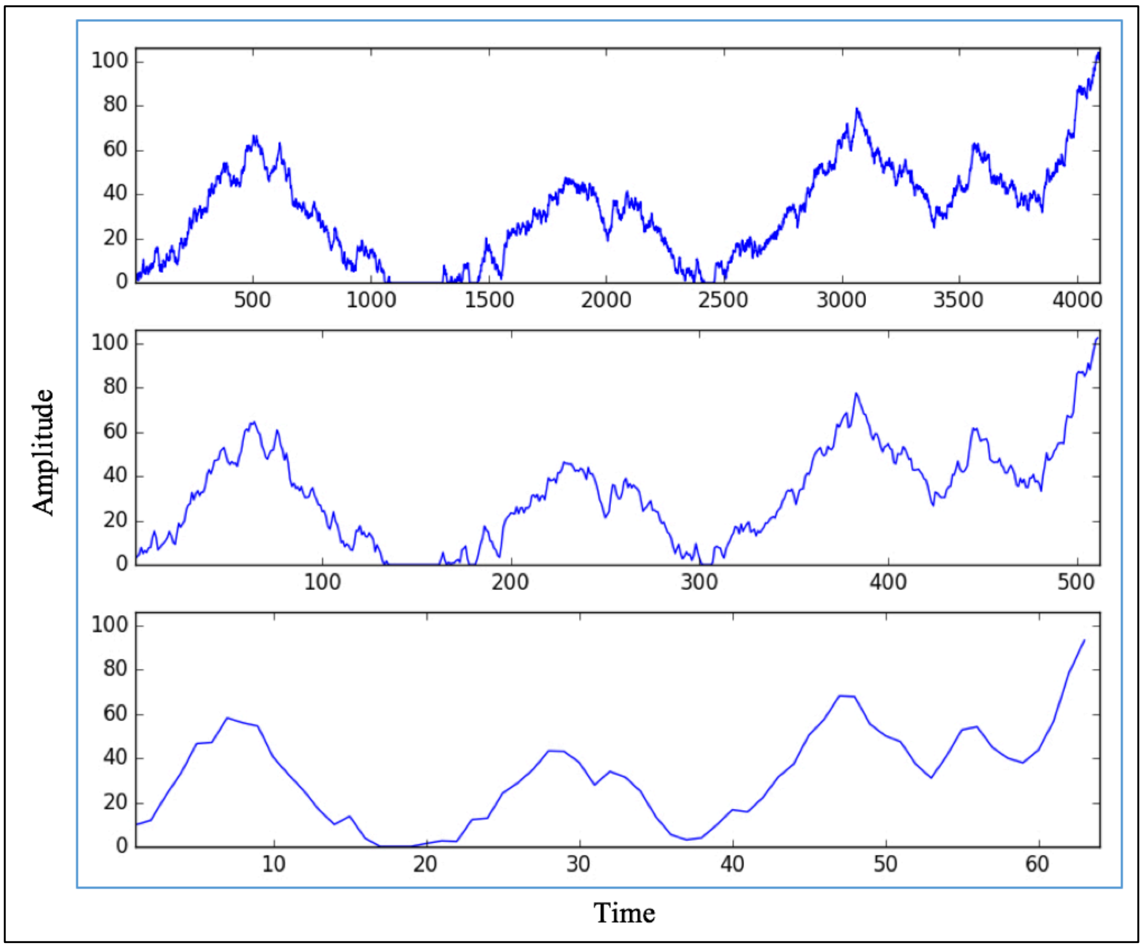


Figure 3.4: Aggregation of FBM traffic.

3.4 Congestion Control by Traffic Aggregation

Congestion control by aggregation uses the statistical multiplexing gain by superimposing different traffic types on a single queue. Due to long-range dependence inherent in many traffic types and present on long time scales, this type of control logic is well suited to be located in the SDN Orchestrator. Considering three traffic types – voice, video, and long-range dependent traffic, modelled by a Po, a MAP and fBm, respectively. A deterministic queue model *G/D/n* is considered for simplicity, where $X \sim G$ is any traffic aggregate following a general distribution *G, s* is the deterministic server capacity in packets per time unit related to *D,* and *n* is the number of queues. For comparison, delay, packet loss and throughput are studied in the absence of traffic aggregation, where each traffic type is fed to its own queue. The traffic streams have similar long-term intensity, and each queue can either be allocated capacity statically or dynamically. In the former case, all resources are simply divided into *n* equal blocks allocated to the queues. In dynamic capacity allocation, a queue is configured for a capacity determined by the entropy of the traffic it is fed with.

By using entropy, an aggregation logic can be formulated based on a single parameter. The entropy of a traffic trace is a measure of its information content or, equivalently, its "randomness". The more similar to a uniform distribution the traffic is, the higher the entropy. In this thesis scheme, the instantaneous traffic descriptor is assumed and available from every node at all times. It is sufficient to let the traffic be described by the number of packets arriving at each discrete time instant (of some convenient granularity) of each source type.

The statistical multiplexing gain by aggregation of traffic increases the utilisation of the queue resources and thereby likely improves the throughput and the QoS for all traffic types. Heuristically, the idea is that short-memory traffic should be allocated resources sufficient to provide services with low (but non-zero) overflow probability. At the same time, the variability of the self-similar traffic can be expected to be limited by the scaled buffer size. The QoS parameters used are the delay and packet loss for each queue during the simulation time frame, from which the total and maximum delay and packet loss of the node are determined. It is assumed that the SDN Orchestrator can measure the traffic intensity of each traffic type so that the control logic can decide which two traffic types to aggregate onto a single server, leaving the third traffic type on a separate server to avoid bandwidth starvation of any of the traffic types.

3.5 Wavelet Analysis

In analysis of communication networks, different characteristics appear on different *scales*. *Wavelets* are mathematical tools for analysing time series or images by decomposing them with respect to different scales. A Wavelet is a "small wave", which essentially grows and decays in a limited time period. A comprehensive exposition on wavelet analysis of time series can be found in [39]. See also [38] for wavelet analysis of long-range dependent processes. A wavelet transform is a representation of a signal – in this case packet arrivals – that allows decomposition into fluctuations on different scales. The discrete wavelet transform (DWT) is defined on discrete time instances that correspond to the sampling rate in the system. A multiresolution analysis is such a decomposition which is aligned in time. Due to its construction, it is a suitable method to analyse self-similar.

In a continuous time, a wavelet is a transform that satisfies

$$\int_{-\infty}^{\infty} \psi(u)du = 0 \tag{3.24}$$

$$\int_{-\infty}^{\infty} \psi(u)du = 1\,, \tag{3.25}$$

for some kernel (or filter) $\psi(u)$, together with other regularity conditions. The kernel can be chosen in many ways and this generates different types of wavelet transforms. The discrete wavelet transform can be regarded as an approximation to the continuous wavelet transform. Consider the real-valued wavelet filter $\mathcal{W}$ with filter coefficients $\{h_l : l = 0, \ldots, L-1\}$ where the width L of the filter is an even integer, so that $h_0 \neq 0, h_{L-1} \neq 0$. define $h_l = 0$ for $l < 0$ and $l \geq L$. A wavelet filter must satisfy

$$\sum_{l=0}^{L-1} h_l = 0\,, \tag{3.26}$$

$$\sum_{l=0}^{L-1} h_l^2 = 1\,, \tag{3.27}$$

$$\sum_{l=0}^{L-1} h_l h_{l+2n} = \sum_{l=-\infty}^{\infty} h_l h_{l+2n} = 0, \quad n \neq 0. \tag{3.28}$$

An interesting and important fact is that the wavelet coefficients are approximately uncorrelated. The DWT therefore uncorrelated even highly correlated series. Let $\boldsymbol{X}$ be a time series represented by a column vector and let $\boldsymbol{W} = \mathcal{W}\boldsymbol{X}$, where $\mathcal{W}$ is the discrete wavelet filter. Define the wavelet *details* as $\mathcal{D}_j = \mathcal{W}_j^T \boldsymbol{W}_j,\ j = 1, \ldots., J$ and $\mathcal{S}_J = \mathcal{V}_J^T \boldsymbol{V}_J$, where $\mathcal{V}$ is the scaling filter. Then

$$\boldsymbol{X} = \sum_{j=1}^{J} \mathcal{D}_j + \mathcal{S}_J \tag{3.29}$$

is called a *multiresolution analysis* of $\boldsymbol{X}$. The multiresolution forms an additive decomposition where each component can be associated with a particular scale $\lambda_j = 2^j$.

The *Daubechies D* (4) *wavelet filter* is based on the filter coefficients

$$h_0 = \frac{1-\sqrt{3}}{4\sqrt{2}} \ ; \tag{3.30}$$

$$h_1 = \frac{-3+\sqrt{3}}{4\sqrt{2}} \ ; \tag{3.31}$$

$$h_2 = \frac{3+\sqrt{3}}{4\sqrt{2}} \ ; \tag{3.32}$$

$$h_3 = \frac{-1-\sqrt{3}}{4\sqrt{2}} \ ; \tag{3.33}$$

Let $\mathcal{T}$ be the time shift operator defined by

$$\mathcal{T}\boldsymbol{X} = [X_{N-1}, X_0, X_1, \dots, X_{N-2}] \tag{3.34}$$

Let $\mathcal{W}_i$ denote the i^{th} row in the wavelet filter. Then the rows in the wavelet transformation matrix $\mathcal{W}$ are related by $\mathcal{W}_{i+1} = \mathcal{T}^2\mathcal{W}_i$ and

$$\mathcal{W} = \begin{pmatrix} h_1 & h_0 & 0 & 0 & \cdots & 0 & 0 & h_3 & h_2 \\ h_3 & h_2 & h_1 & h_0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & h_3 & h_2 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & h_1 & h_0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \cdots & h_3 & h_2 & h_1 & h_0 \end{pmatrix} \tag{3.35}$$

The *pyramid algorithm* provides an efficient algorithm to compute the DWT of a time series. The pyramid algorithm can be expressed in linear matrix operations as follows:

Step 1: Define the $N \times \frac{N}{2}$ matrices

$$\mathcal{W}_1 = \begin{pmatrix} h_0 & h_1 & h_2 & h_3 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & h_0 & h_1 & h_2 & h_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & h_0 & h_1 & h_2 & h_3 \\ h_2 & h_3 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & h_0 & h_1 \end{pmatrix} \tag{3.36}$$

$$\mathcal{V}_1 = \begin{pmatrix} g_0 & g_1 & g_2 & g_3 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & g_3 & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & g_0 & g_1 & g_2 & g_3 \\ g_2 & g_3 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & g_0 & g_1 \end{pmatrix} \tag{3.37}$$

where h_1 and g_1 are found from (3.8) and $g_l = (-1)^{l+1}h_{L-1-l}$.

Step 2: Multiply the time series vector $\boldsymbol{X}$ with $\mathcal{W}_1$ and $\mathcal{V}_1$, respectively, which yields the first order wavelet coefficients and scaling coefficients:

$$\boldsymbol{W}_1 = \mathcal{W}_{\mathbf{1}}\boldsymbol{X} \tag{3.38}$$

$$\boldsymbol{V}_1 = \mathcal{V}_1\boldsymbol{X}. \tag{3.39}$$

Divide N by 2, goto step 1 and apply the filters to the data vector $\boldsymbol{V}_1$.

Step j**:** Let $N := N_j = N/2^j$, goto step 1 and apply the filters to the data vector $\boldsymbol{V}_{j-1}$. Repeat wavelet transform of $\boldsymbol{X}$ is given by

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{W}_1 \\ \vdots \\ \boldsymbol{W}_J \\ \boldsymbol{V}_J \end{pmatrix}. \tag{3.40}$$

Unfortunately, the DWT requires that the number of data points is a power of 2. This requirement is relaxed for the Maximum Overlap Discrete Wavelet Transform (MODWT), which is well-defined for any sample size N. The MODWT is also suitable for multiresolution analysis. Define the *maximal overlap discrete wavelet transform wavelet filter,* $\{\tilde{h}_l\} : \tilde{h}_l \equiv h_l/\sqrt{2}$ and the *maximal overlap discrete wavelet transform scaling filter,* $\{\tilde{g}_l\} : \tilde{g}_l \equiv g_l/\sqrt{2}$ so that

$$\sum_{l=0}^{L-1} \tilde{h}_l = 0\,, \tag{3.41}$$

$$\sum_{l=0}^{L-1} \tilde{h}_l^2 = \frac{1}{2}, \tag{3.42}$$

$$\sum_{l=-\infty}^{\infty} \tilde{h}_l\tilde{h}_{l+2n} = 0, \tag{3.43}$$

for all nonzero integers n. This gives the first level MODWT ($J_0 = 1$)

$$\widetilde{W}_{1,t} = \sum_{l=0}^{L-1} \tilde{h}_l X_{t-l \mod N}\,, \tag{3.44}$$

$$\tilde{V}_{1,t} = \sum_{l=0}^{L-1} \tilde{g}_l X_{t-l \mod N}\,, \tag{3.45}$$

for $t = 0, \ldots, N-1$. By repeating this operation on $\tilde{V}_{1,t}$ gives the details on successively longer scales.

In the multiresolution analysis, the partial time series on different scales add up to yield the original time series itself. Illustrations of these partial time series show a strictly positive result as required. In the following, multiresolution analyses of Markovian, self-similar and aggregate traffic are illustrated. A Daubechies MODWT of width $L = 4$ is used. The figures show, from the top, fluctuations on levels $J = 1, 2, \ldots 5$, the *details,* and the *smooth* on level $J = 5$. Multiresolution analyses up to level $J_0 = 5$ of Poisson and MAP traffic are shown in Figure 3.5 and 3.6, respectively. Note the rapidly decreasing amplitudes on large scales. This indicates that the traffic is fairly regular and without causing long periods of saturation in a queue, provided that the intensity is less than the processing capacity ($\lambda < s$).

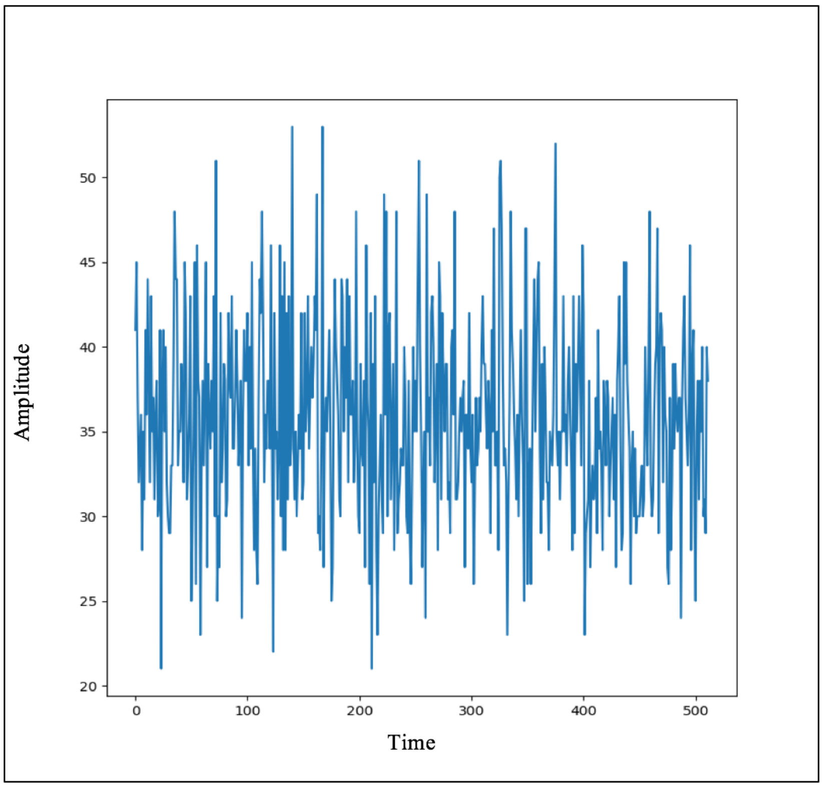


Figure 3.5: Multiresolution analysis of Poisson traffic

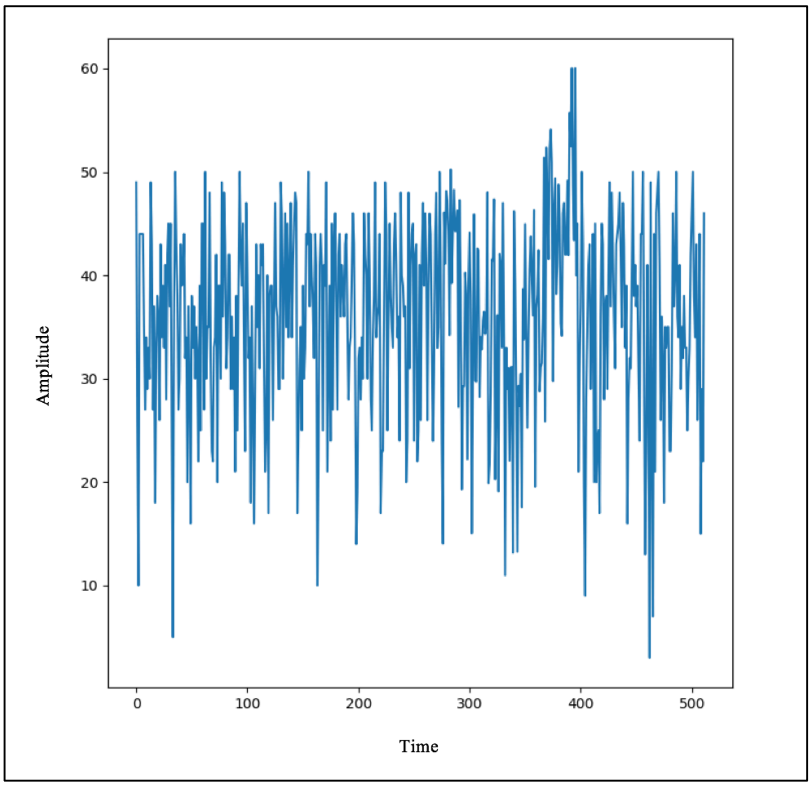


Figure 3.6: Multiresolution analysis of MAP traffic

For the fBm, the multiresolution analysis (Figure 3.7) shows large variations on both short and long scales. Even if the average load is lower than the processing capacity in the queue, there are relatively long periods of time where the workload in a queue builds up and causes congestion. Comparing the multiresolution analyses of the total aggregate traffic shown in Figure 3.8 and some of the Bellcore used in [31] and shown in Figure 3.9, reveals similar behaviour on both short and long scales. Notably, the amplitude on longer scales does not decrease significantly, but remains between 10-20% of the first details.

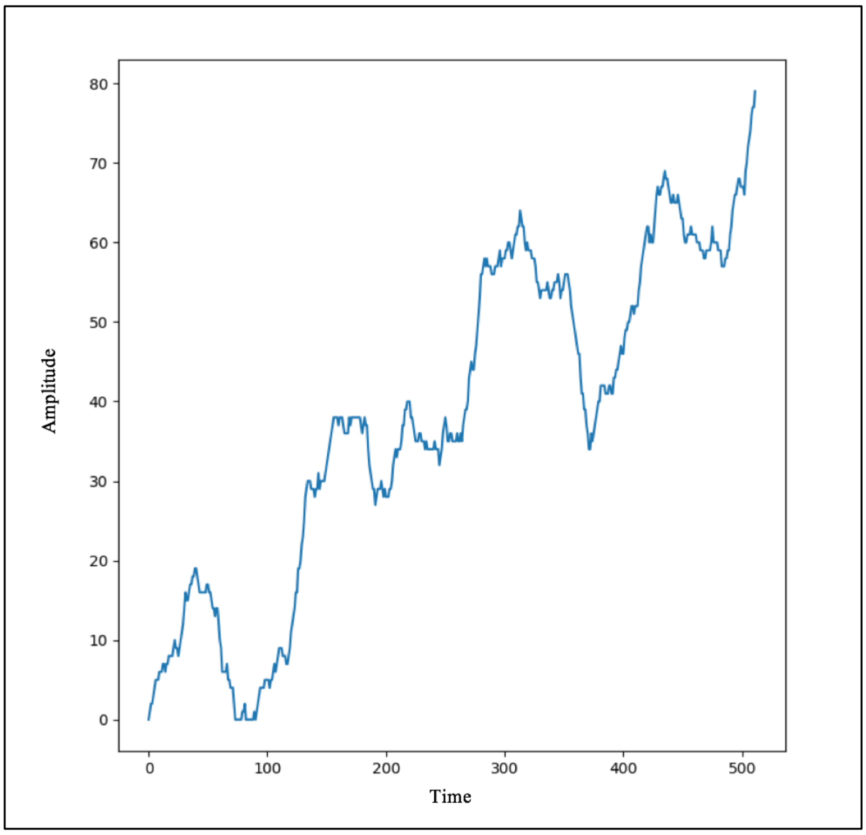


Figure 3.7: Multiresolution analysis of fBm traffic

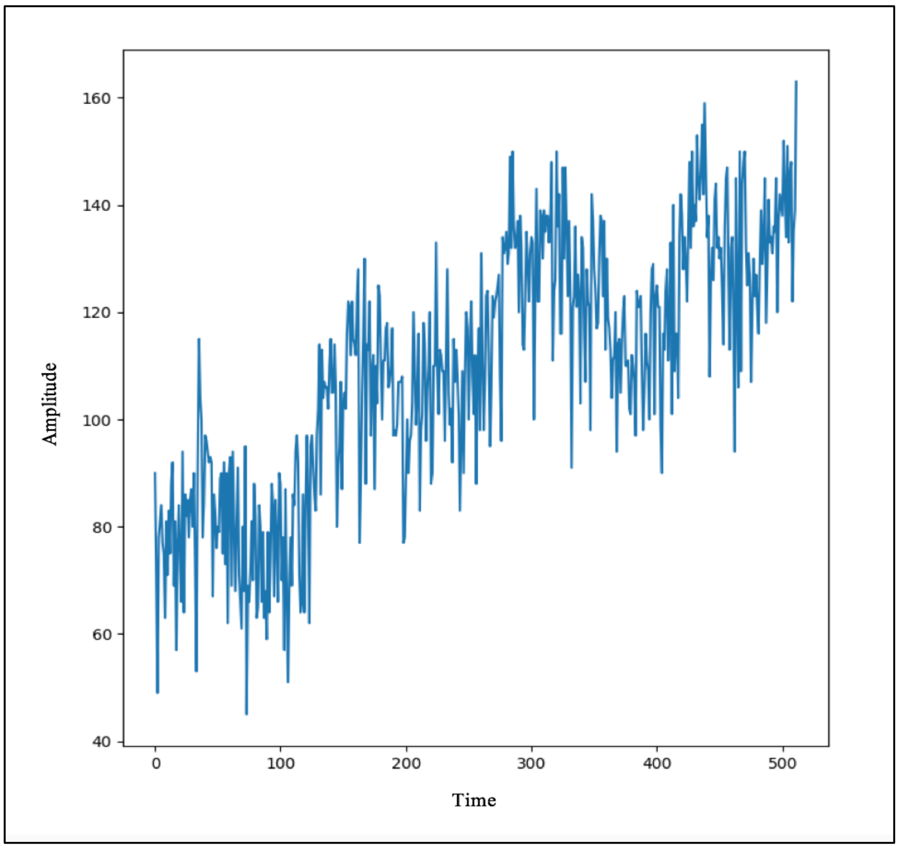


Figure 3.8: Multiresolution analysis of total simulated traffic

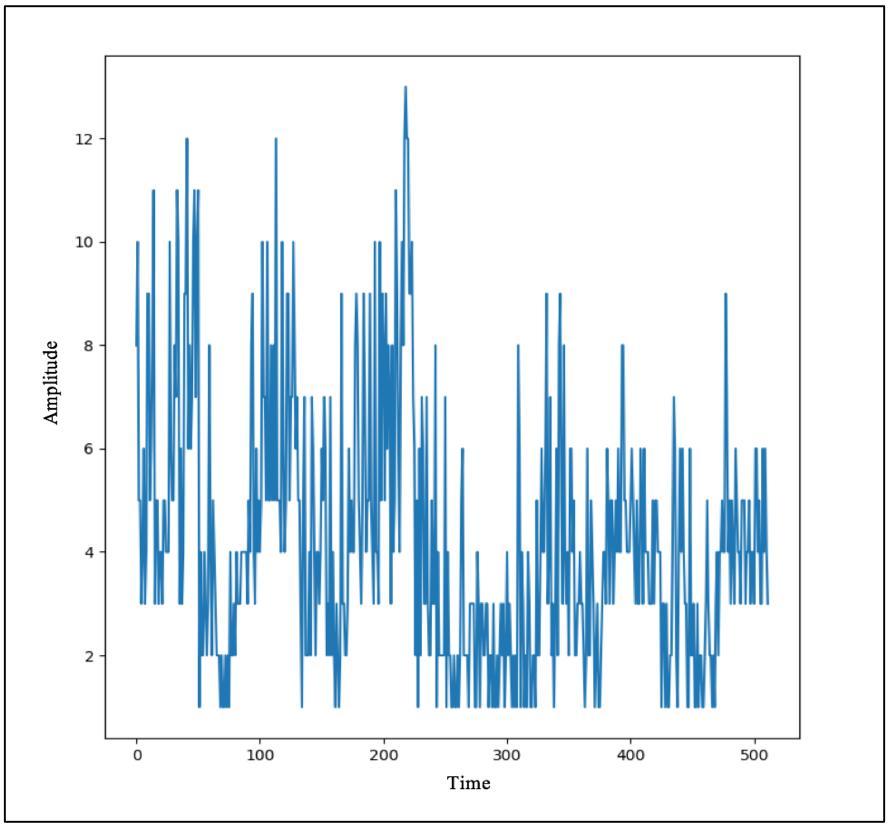


Figure 3.9: Multiresolution analysis of Bellcore traffic [57]

3.6 Chapter Summary

This chapter conceptually using entropy as a single traffic aggregation decision parameter. Entropy is not necessarily the best, but it is a single characterizing quantity and therefore convenient to use as a decision parameter. Most importantly, however, is that it can be efficiently measured in high-speed networks. At gigabyte rates, packets arrive at nanosecond intervals, and therefore streaming data algorithms need to be used for its estimation [59][60][62]. Using the entropy is convenient because some parsimonious characterization of the traffic which then would allow for truly real-time congestion control is needed. The investigation into traffic aggregation is to improve throughput and performance. As in the multiresolution analysis, a time series X is expressed as a sum of J vectors, each of which contains a time series related to X on a particular scale. This decomposition shows variations in X from high to low frequency, which is very useful to illustrate the characteristics of different traffic types [58].

Chapter 4

OPTIMAL ROUTING FOR CONGESTION CONTROL

4.1 Introduction

Optimal routing is a network optimisation which provides route optimisation on load balancing mechanisms. The routes are chosen so that the aggregation along the routes either minimises the total end-to-end delay, or the maximum end-to-end delay. The optimal routing provides "fairness", that trading average performance for better worst performance. At the end of the day, as long as service level agreements are fulfilled, carriers would target the worst performance.

4.2 The Workload Management Process

Consider a single server queue in discrete time under a first-come first-served (FCFS) queueing discipline. Assume a single class of jobs, which can be justified if different traffic types are mixing into a single traffic stream without any priorities assigned to any of the traffic types. Now the following two fundamental and rather natural assumptions in a system is said to be work conserving if

(1) No servers are idle when there is at least one job waiting,

(2) The service times are independent of the queue length.

For a discrete-time queue, a time stamp t represents a time interval between t and $t+1$. Thus, let A_t be the amount of additional jobs arriving at the queue at time t, and S_t be the amount of jobs processed by the server at this time. Then the delay of jobs in a queue, represented by the queue lengths Q_t, can be described by

$$Q_{t+1} = \max\{0, Q_t + A_t - S_t\}, \tag{4.1}$$

known as Lindley's equation [38]. This equation is the basis of the simulator, described in [Section 5.8]. Note that the queue length never can be negative, so it is not possible to save processing power when the queue is idle.

This thesis defines work load as the content of the queue plus arriving jobs minus processed jobs. The net change in work load can be written $U_t = A_t - S_t$, and Equation (4.1) can be expressed as

$$Q_{t+1} = \max\{0, Q_t + U_t\}. \tag{4.2}$$

Sometimes this recursion is used from a past moment in time up to the present time. Then the time index representing the past time is negative. Consider the first step in the recursive from $t = -1$ to $t = 0$,

$$Q_0 = \max\{0, Q_{-1} + A_{-1} - S_{-1}\}. \tag{4.3}$$

By letting $U_t = A_{-t} - S_{-t}$, changing of sign in the index of U for convenience and iteration (4.3), make

$$\begin{aligned} Q_0 &= \max\{0, U_1 + Q_{-1}\} && (4.4)\\ &= \max\{0, U_1 + \max\{0, Q_{-2} + U_2\}\} && (4.5)\\ &= \max\{0, U_1, U_1 + U_2 + Q_{-2}\}. && (4.6) \end{aligned}$$

The first equation says that the queue length at time t is the work arriving minus the work processed in one-time step, plus the previous queue length. On the second line, the expression for Q shifted in time by using the recursion, and on the third line, the $\max\{.\}$ operator is "moved out" according to the relation $U_1 + \max\{0, U_2 + Q_{-2}\} = \max\{U_1 + 0, U_1 + U_2 + Q_{-2}\}$. By continuing the recursion, a sequence with an increasing number of terms is obtained, representing the net change in work load at time t. Denoting the sum of work load changes by $W_t = U_1 + U_2 + \cdots U_t$, make

$$Q_0 = \max\{W_0, W_1, \dots, W_t + Q_{-t}\}, \tag{4.7}$$

where W_0 is defined as $W_0 = 0$. W_t is called the *workload process*. If there is some finite time $-\tau$ in the past when the queue was empty, then

$$Q_0 = \max\{W_0, W_1, \dots, W_\tau\}. \tag{4.8}$$

An example of a work load process is shown in Figure 4.1.

Supposing that the queue has been running for a long time, and that it is stable, that is

$$\boldsymbol{E}(A_t) < \boldsymbol{E}(S_t) \tag{4.9}$$

holds. The stability condition implies that the mean arriving work is less than the mean processed work. This assumption is necessary – otherwise a queue with a finite buffer would constantly overflow.

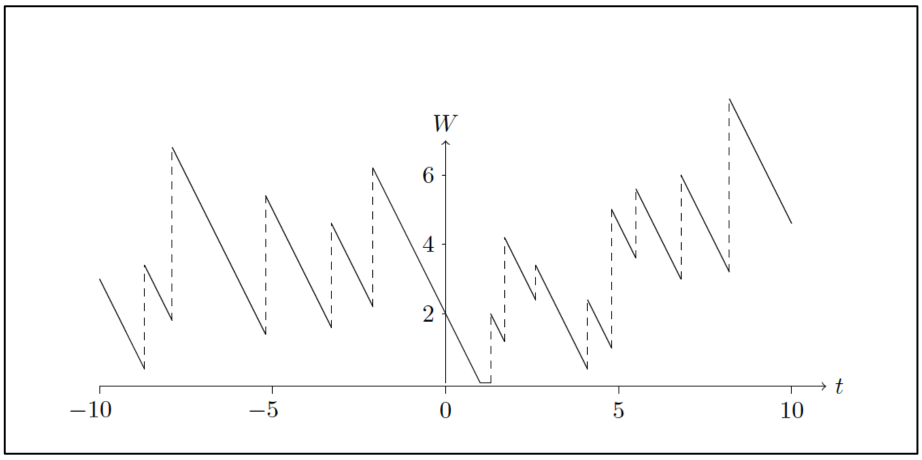


Figure 4.1: The workload process for a GI/G/1 queue [29].

If Equation (4.9) holds, then the queue must be empty at some point in time. It then follows that the steady state queue length is independent of the initial queue length. In this case, the steady state queue length Q can be written

$$Q = \max_{t \geq 0} W_t. \tag{4.10}$$

provided that the stability condition (4.9) is satisfied. The equation (4.10) is derived in the reference. The maximum (or supremum) over t in the limit must be constant when the workload W is the aggregation of arrivals minus departures, and the remaining queue content carries over from event to event. This reflects the steady-state queue length [33]. This is a general result for a GI/G/1 queue (where GI is short for Generally distributed and Independent arrivals). For first-come first-served queueing systems the delay is equal to the work load function at the time of the arrival of a job.

4.3 Theory of Large Deviations

This thesis shows that the queue length distribution is related to the work load process. Since the events that occur relatively seldom is interested, such as buffer overflow, the theory of large deviations can be used to obtain some approximate results. Large deviations are used to analyse probability distributions at their *tails,* far away from the distribution mean. Suppose to have a queue with a buffer of size b. Taking the probability on both sides of Equation (4.10) gives the packet loss probability

$$\boldsymbol{P}(Q > b = \boldsymbol{P}\,(\sup_{t\geq 0} W_t > 0). \tag{4.11}$$

Now, the inequality can be used as

$$\boldsymbol{P}(\sup_{t\geq 0} W_t > b = \boldsymbol{P}(\cup_{t\geq 0}\,\{W_t > b\}) \sup_{t\geq 0} \boldsymbol{P}(W_t > 0), \tag{4.12}$$

which for large b leads to the approximation

$$\boldsymbol{P}\left(\max_{t\geq 0} W_t > b\right) \approx \sup_{t\geq 0} \boldsymbol{P}\,(W_t > b). \tag{4.13}$$

The approximation implies that when the probabilities $\boldsymbol{P}(W_t > b)$ decay sufficiently fast with increasing b, the left-hand side of Equation (4.13) can be approximated by its largest term. This probability can be interpreted as the fraction of time when then queue length Q exceeds b. The asymptotic form of this probability is given by *Cramér's theorem.*

Theorem 4.2.1. *Let $X_1, X_2, X_3, \ldots$ be a sequence of bounded, independent and identically distributed random variables with mean μ, and let S_n be the sum of (the first) n variables, and*

$$M_n = \frac{1}{n} S_n = \frac{1}{n}(X_1 + \cdots + X_n) \tag{4.14}$$

denote the empirical mean of S_n. The the tails of the probability distribution of S_n decay exponentially with increasing n at a rate given by a convex rate function I(x)

$$\boldsymbol{P}(M_n > x) \asymp e^{-nI(x)} \quad for\ x > \mu \tag{4.15}$$

$$\boldsymbol{P}(M_n > x) \asymp e^{-nI(x)} \quad for\ x < \mu \tag{4.16}$$

The sign $\asymp$ (asymptotically equal to) means that the expressions on the left hand side tends to the right hand side as n tends to infinity. For a proof, see for example [33].

There is a close relationship between the rate function and the scaled CGF, denoted $\Lambda(\theta)$. The CGF is the logarithm of the *moment generation function,* $M(\theta)$, which is defined as

$$M(\theta) = \boldsymbol{E}(e^{\theta X}) = \int e^{\theta x} f(x)dx, \tag{4.17}$$

where X is a random variable with probability density function $f(x)$. The CGF is then $\Lambda(\theta) = \ln M(\theta)$. These functions can be used to compute the moments or cumulants of a distribution. The values at the origin are

$$M(0) = 1 \tag{4.18}$$

$$\Lambda(0) = 0, \tag{4.19}$$

and the mean and variance can be computes as

$$\mu = \boldsymbol{E}(X) = \frac{dM(\theta)}{d\theta}|_{\theta=0} = \frac{d\Lambda(\theta)}{d\theta}|_{\theta=0} \tag{4.20}$$

$$\sigma^2 = Var(X) = \frac{d^2M(\theta)}{d\theta^2}|_{\theta=0} = \frac{d^2\Lambda(\theta)}{d\theta^2}|_{\theta=0} + \left(\frac{d\Lambda(\theta)}{d\theta}|_{\theta=0}\right)^2 . \tag{4.21}$$

A convenient property of the scaled CGF is its additivity,

$$\Lambda_{X+Y}(\theta) = \log \boldsymbol{E}\left(e^{\theta(X+Y)}\right) = \log\left(\boldsymbol{E}(e^{\theta X})\,\boldsymbol{E}(e^{\theta Y})\right) \tag{4.22}$$

$$= \log\left(\boldsymbol{E}(e^{\theta X})\right) + \log\left(\boldsymbol{E}(e^{\theta Y})\right) = \Lambda_X(\theta) + \Lambda_Y(\theta). \tag{4.23}$$

4.3 The Chernoff Formula

In order to compute the rate function in Cramér's theorem (4.2.1) for a sequence of independent random variables, *Chernoff's Formula* can be used. The formula is derived by finding an upper bound on the tail probabilities. Let $X_1, X_2, \dots, X_n$ be independent and identically distributed random variables with average $M_n = \frac{1}{n}(X_1, X_2, \dots, X_n)$. An upper bound on the probability $\boldsymbol{P}(M_n > x)$ can be defined as follows. Let $I_A(.)$ be the indicator function of a set $A \subset \mathbb{R}$ such that

$$I_A(x) \triangleq \begin{cases} 1 & if\ x \in A \\ 0 & otherwise \end{cases} \tag{4.24}$$

Consider an interval $A = [a, \infty)$ on the x-axis. Then $I_A(x)$ is a step function with value one if $x \in [a, \infty)$ and zero otherwise. Figure 4.2 shows graphically that for any number a and positive number $\theta, I_{[a,\infty)} \leq e^{\theta x}/e^{\theta a}$. Note that $\boldsymbol{E}\big(I_{[na,\infty)}(nM_n)\big) = \boldsymbol{P}(nM_n > na)$, and

$$\boldsymbol{P}(M_n > a) = \boldsymbol{P}(nM_n > na) \tag{4.25}$$

$$= \boldsymbol{E}\left(I_{[na,\infty)}(nM_n)\right) \tag{4.26}$$

$$\leq \boldsymbol{E}\big(e^{\theta n M_n}/e^{\theta n a}\big) \tag{4.27}$$

$$= e^{-\theta n a}\boldsymbol{E}\big(e^{\theta(X_1 + \cdots + X_n)}\big) \tag{4.28}$$

$$= e^{-\theta n a}\left(\boldsymbol{E}\big(e^{\theta X_i}\big)\right)^n \tag{4.29}$$

where the random variables X_i are independent and identically distributed is assumed. Denoting the scaled cumulant generating function by $\Lambda(\theta) = \ln\left(\boldsymbol{E}\big(e^{\theta X_1}\big)\right)$, this thesis have $\boldsymbol{P}(M_n > a) \leq e^{-n(\theta a - \Lambda(\theta))}$. Since this holds for each θ can be optimised over θ which gives

$$\boldsymbol{P}(M_n > a) \leq \min_{\theta > 0} e^{-n(\theta a - \Lambda(\theta))} = \exp\left(-n \max_{\theta > 0}\big(\theta a - \Lambda(\theta)\big)\right). \tag{4.30}$$

If a is greater than the mean m, a lower bound is given by

$$\boldsymbol{P}(M_n > a) \asymp \exp\left(-n \max_{\theta > 0}\big(\theta a - \Lambda(\theta)\big)\right). \tag{4.31}$$

The result of this argument is known as Chernoff's Formula.

Theorem 4.3.1 (Chernoff's Formula)**.** *The rate function can be calculated from the cumulant generating function $\Lambda(\theta)$ using*

$$I(x) = \max_{\theta}\{x\theta - \Lambda(\theta)\}, \tag{4.32}$$

with $\Lambda(.)$ defined by

$$\Lambda(\theta) \triangleq ln\left(\boldsymbol{E}\big(e^{\theta X_i}\big)\right) \tag{4.33}$$

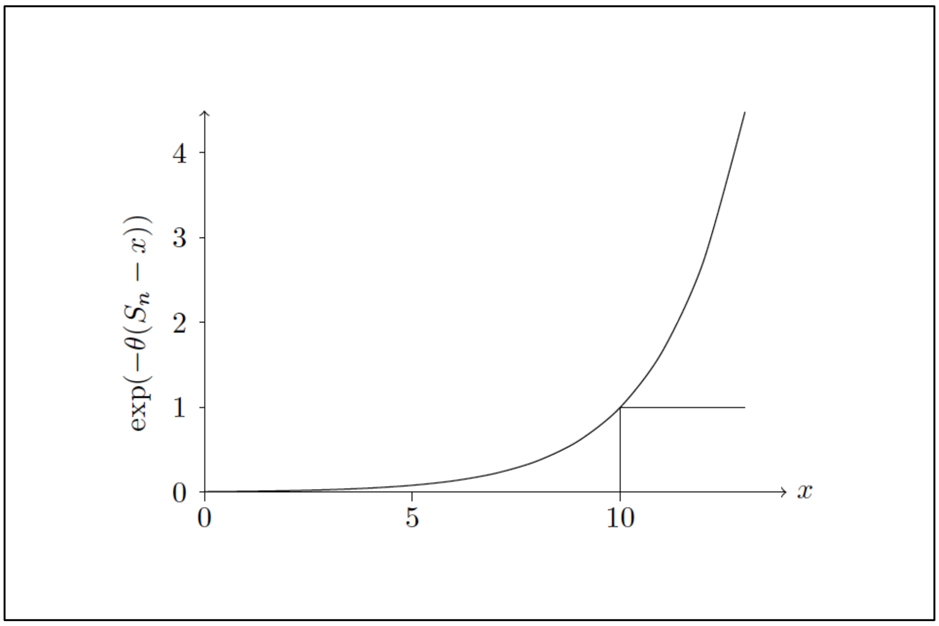


Figure 4.2: Chernoff's formula (with $\theta = 0.5$ and $x = 10$).

The asymptotic results in Theorem 4.2.1 and Equation (4.32) can now be used to estimate the queue length Q, and find an approximate probability distribution of $\boldsymbol{P}(Q \geq b)$ for large b. Figure 4.3 shows $\log \boldsymbol{P}\,(Q > b)$ for an *M/D/1* queue and an *M/M/1* queue. For both queues the traffic load is $\rho = \frac{\boldsymbol{E}(A_t)}{\boldsymbol{E}(S_t)} = 0.7$. the packet loss rate is asymptotically exponential in b for large b in both cases, but with different slopes.

The exponential decay may be expressed as

$$\boldsymbol{P}(Q > b) \asymp e^{-\delta q}\,, \tag{4.34}$$

with decay rate δ determined from the rate function of the workload process by

$$\delta = \min_{x} \frac{I(x)}{x}. \tag{4.35}$$

There is an important relationship between the rate function, which describes the macroscopic properties of the traffic, and entropy, describing its microscopic states. The entropy can be interpreted as the information content in a distribution, or its "randomness". The more similar to a uniform distribution, the higher the entropy. Formally, the entropy for a discrete distribution X is defined as

$$H(X) = -\sum_{i} p_i \log p_i, \tag{4.36}$$

where p_i is the probability of occupying state i. In practice, the number of packets arriving at any time instant represent a state where p_i is the frequency of occupying a particular state divided by the sample size. The entropy has been proposed as a traffic descriptor in the literature. Crosby et al. [6] suggest that the entropy is sufficient to estimate cell loss and delay in ATM networks. Tuan and Park [48] also uses entropy to measure the information content of long-range dependent traffic for classification of the traffic into load levels.

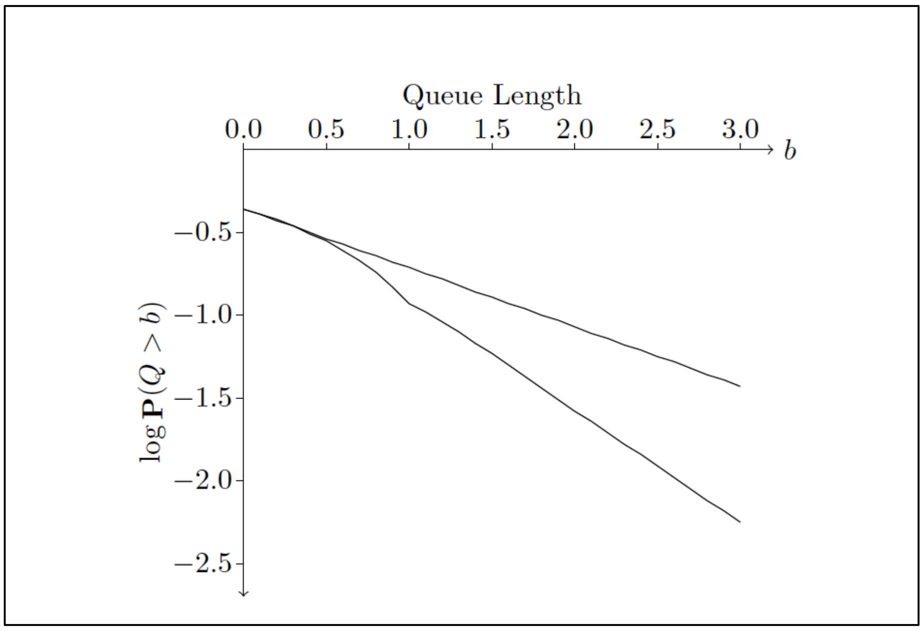


Figure 4.3: Comparison between an *M/D/*1 and an *M/M/*1 queue with the same traffic intensity $\rho = 0.7$ where the *M/D/*1 queue has the steeper decay.

4.4 Congestion Control

The aim of congestion control in communication networks is to provide services to its users that meet certain performance criteria, often represented by a set of QoS parameters. The performance criteria typically differ between services. One of the challenges in meeting QoS requirements is avoiding bandwidth starvation of certain traffic types by others. This may happen when one or more traffic types are given unconditional priorities. Another cause of congestion is traffic long-range dependence. Congestion control by using two fundamental techniques is studied for efficient resource utilisation – traffic aggregation and dynamic routing. Traffic aggregation refers to the resource utilisation in the nodes, that is, processing capacity and buffer size. Dynamic routing, on the other hand, aims at directing traffic so that the overall network performance is maximised. It affects both loads at the nodes and transmission resources. In this study, however, the transmission resources are assumed to have sufficient capacity for any traffic stream, and do not induce any significant propagation delay. This assumption is reasonable in optical Dense Wavelength Division Multiplexing (DWDM) backbone networks. WDM is promoted by the industry as the de facto preferred transport medium in 4G/5G networks. Several initiatives such as the 5G-XHaul are investigating the use of WDM in 5G. 5G-XHaul is a European Union funded project to investigate the use of WDM-PON technology for mobile backhaul and mobile fronthaul. The 5G-XHaul initiative is part of the EU Horizon 2020 5G Infrastructure Public-Private Partnership (5G PPP). For the purpose of the thesis, the assumption of using WDM allows disregarding capacity limitations on the links [55].

The distinguish between congestion control on flow level, such as TCP (also referred to as congestion avoidance), and congestion control on network level, being the focus of this investigation. On the network level, the static QoS guarantees implemented by resource reservation, or soft QoS guarantees implemented through traffic prioritisation and scheduling. As noted in [14], such QoS enforcement may adversely affect traffic not subjected to QoS restrictions and lead to bandwidth starvation. The described algorithm does not impose any QoS restrictions. It is designed to allocate resources so that it is being utilised in the best possible way. The only physical restriction affecting the traffic is the scaling of node resources with respect to traffic entropy.

Letting λ be the arrival rate and γ the throughput, a common type of feedback congestion control on flow level has the form

$$\frac{d\lambda}{dt} = \begin{cases} \backslash\epsilon, & if\ d\gamma/d\lambda, \\ -\alpha\lambda, & if\ d\gamma/d\lambda, \end{cases}, \tag{4.37}$$

where $\epsilon, \alpha > 0$ are constants. Thus, if an increase in traffic flow rate results in an increase in throughput $(d\gamma/d\lambda > 0)$, then the flow is increased linearly by ϵ, whereas if an increase in flow rate leads to a decrease in throughput $(d\gamma/d\lambda < 0)$, then the flow rate is decreased exponentially. Gerla and Kleinrock [19] describes different effects of congestion and propose mechanisms to mitigate these effects. Most notably are variants of resource reservation, where a single type of traffic is prevented from fully occupying a network resource (for example, a link or a buffer). Setting rules for resource allocation have been shown to be efficient in avoiding congestion and network instability effects. Rather than using a protocol-based congestion control like Equation (4.37), we investigate how available network resources best can be utilised is investigated and how that affects performance on a network level. By pointing it out, however, that combination of congestion control methods on different levels need to be implemented to have a robust network operation, such as admission control and rate control on flow level (such as TCP-controlled flows).

4.4.1 Congestion Control by Optimal Routing

A natural method for congestion control is using routes involving nodes with low load. Such routing achieves load balancing of the network resources. With a central knowledge of the load and average delay of each node, an end-to-end minimum delay route can in principle be found by solving a shortest path problem for each flow. However, shortest path algorithms (for example Dijkstra's algorithm) are based on link properties, whereas here have node properties. Therefore, construct a link cost matrix from the node delays as follows.

Denote the delay incurred by node i by δ_i. Assuming independence between nodes, so that the delays are additive, the delay on path p can be written

$$\delta_p = \sum_{i_p}^{N_p} \delta_{i_p} \quad , \tag{4.38}$$

where the nodes on path p are indexed $1, \ldots, N_p$. If these delays are mapped onto links, where each link along path p receives a fictitious delay Δ_{p_i, p_i+1}

$$\Delta_{i_p} = \frac{\delta_{i_p}}{2} + \frac{\delta_{i_p+1}}{2}, \tag{4.39}$$

see that

$$\delta_p = \sum_{i_p=1} N_p - 1\Delta_{i_p} + \frac{\delta_1}{2} + \frac{\delta_{N_p}}{2}. \tag{4.40}$$

Since the last terms are the delay at the ends of the path, the shortest path has not been affected. Applying Dijkstra’s algorithm on this link cost structure then gives the minimum delay path.

4.5 Chapter Summary

In this chapter, we justify out simulation set-up by some general arguments from queueing theory and large deviations theory. Firstly, we may conclude that exact calculation of the QoS in a network is hard, or more specifically #P-complete. This follows from the fact that finding the exact blocking in a general loss network is #P-complete [34], and that blocking, just like delay and packet loss, are related to the properties of queues offered traffic that is leaving other queues. Note that #P-complete problem is at least as hard as an NP-complete problem, and for such problems even approximations are difficult to find. Exact calculation of traffic and QoS parameters in a meshed network with heterogeneous traffic and dynamic routing is in practice impossible. We therefore revert to simulation to verify the efficiency of dynamic routing, as expressed by the QoS parameters delay, packet loss and throughput. These quantities can easily be deducted from the simulated queues representing routers or switches.

Chapter 5

Simulation and Results

5.1 Introduction

In this study, the end-to-end delay as the main optimisation target parameter and performance indicator are used. Two other system characteristics are also measured; packet loss and throughput. Packet loss is considered less important than delay due to the retransmission capabilities of TCP. Throughput, on the other hand, is a measure of resource utilisation rather than service quality. It is therefore of less importance when considering end-to-end connections in a network context. In this thesis, the investigation will be focus more to performance on network level rather than on flow level. In this context, throughput is not an additive measure, since large throughput downstream would not be useful when the throughput upstream is lower.

For the performance analysis, both the total delay and the maximum delay of end-to-end connections are determined. In addition of being a measure of service quality, the delay can also be seen as a measure of the load of the individual queues, since it is directly related to the average queue length. There is also an obvious trade-off between delay and packet loss. The probability of packet loss increases with decreasing buffer size, but at the same time, the delay decreases. Indeed, small buffers have been suggested for self-similar traffic [38]. The principle is that for such traffic, it is better to discard packets than causing overload with large delays as a consequence.

5.2 Simulation of Congestion Control

In the simulation, the performance improvements of the proposed traffic aggregation and routing strategies is shown. The use of simulation can be motivated by two facts. Firstly, an analytic treatment of the strategies is $\#\mathcal{P}$-complete and additional complexity is introduced by traffic long-range dependence, which consequently affects all routes and performance measures. Secondly, the control logic would be based on statistical network properties, so a simulation approach directly gives a realistic "blueprint" of the algorithm. To carry out simulation of congestion control strategies, several criteria needs to be defined

(1) Network topology
(2) Router capabilities
(3) End-to-end traffic distribution
(4) Simulated load from the different traffic types
(5) Aggregation and routing logic, and QoS evaluation

In simulating the effect of the congestion control logic, the network topology fixed is kept. In the simulations, a network topology consisting of seven nodes is used. Please refer Figure 5.1. These nodes are assumed to be identical for the sake of clarity of the performance evaluation results. The end-to-end traffic distribution is randomly sampled to allow for maximum flexibility in routing scenarios. Each router is fed with an individually simulated traffic load, consisting of Poisson, MAP, and fBm traffic. The intensity of the simulated traffic traces is the same for each node, again to simplify the interpretation of the simulation results.

The control logic performs measurements on incoming traffic streams and is assumed to possess information of the queue states of the nodes and the traffic distribution matrix. Based on this information, it configures the nodes depending on the aggregated traffic characteristics and determines end-to-end routes in the network. Finally, the QoS parameters delay and packet loss in each node are collected and the end-to-end parameters are determined. A comparison of both the total sum of end-to-end delays, packet losses, and throughputs, and the sums of maximum end-to-end delays, packet losses and throughputs are used for performance evaluation. The logical representation of the congestion controlled network is shown in Figure 1.1. It consists of a network of nodes (routers and switches) with high-speed optical links connecting to them, and a SDN Orchestrator (hosting the central logic) that is able to pull traffic statistics and the states of each node from the network, to configure buffer sizes and transmission rates in the nodes, and to modify their forwarding tables. For a full flexibility in traffic generation, node configuration, route optimisation, and detailed statistics, this thesis developed a simulation framework dedicated to the present study, rather than using SDN development platforms like Mininet or NS-3. The simulator is written in Python, and was run on a 64-bit dual-processor HP 6830s Linux Fedora 25 platform. Each simulation consists of generating three traffic traces, each of size 512 (that is, 512 time steps), performing three different aggregations, and calculating the resulting entropy for each node in the network. Being a power of two, this is a convenient size for wavelet decomposition (multiresolution analysis). Based on the entropy, the nodes are configured proportionally, and the queues are simulated by feeding them with the computed traffic aggregates.

Next, the performance measures are calculated and used to determine optimal routes. The route optimisation is iterative and uses a fixed-point equation which converges in a few iterations. In each step, the traffic aggregation at the nodes has to be recalculated. Each iteration on the network runs for approximately 100 seconds on the platform used. Each network scenario – route optimisation with respect to total end-to-end delay or maximum end-to-end delay – is simulated 100 times to generate statistical gain figures. The simulation is facilitated by using a *G/D/n* queue, which means that the time steps are equidistant with $t_0 = 0$ and $t_n \in \left((n-1)h, nh\right]$, where h is an arbitrary time unit. The packet loss and delay induced in each node is calculated from the simulated queue length distributions.

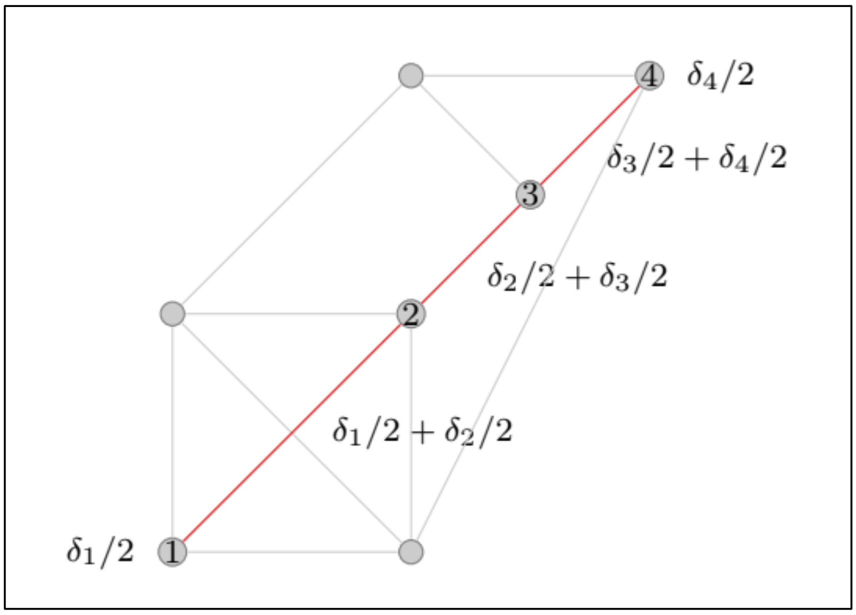

$\delta_4/2$
$\delta_3/2 + \delta_4/2$
$\delta_2/2 + \delta_3/2$
$\delta_1/2 + \delta_2/2$
$\delta_1/2$
1
2
3
4

Figure 5.1: Dijkstra's algorithm using delays along a path as cost, mapped onto the links (red).

5.3 Components of Network Simulation

In this section, the components used in this research are briefly explained. As mention in the subsection below, the reason why the proposed network topology is justified, how the router capabilities is being initialised, and the traffic distribution used in the simulation.

5.3.1 Network Topology

The network topology is chosen so that there are sufficiently many routing possibilities to yield interesting results and still small enough to allow for fast simulation. The speed of simulation is imperative in generating a large number of cases, which is necessary due to the self-similar nature of the traffic. The topology is a 3-connected network, which is described in [29]. The 3-connectivity ensures a certain level of resilience, which also implies path diversity. The network is modelled by a graph, on which shortest paths are determined. Since the network topology is assumed given, the shortest paths are determined with respect to delay, rather that distance, only the connectivity matrix is used in most of the simulations by assuming that the links have infinite capacity and induce zero delay. This is considered a good approximation for optical fibre networks carrying moderate data volumes. To be able to compute the shortest paths, however, the links are associated with a fictitious delay parameter deduced from the router delays, as described in section 5.6.

The network topology is a 3-connected graph, which means that between any two nodes there are (at least) 3 edge-disjoint paths. This fact, referred to as path diversity, is used in the dynamic routing, where the path with the lowest total delay is chosen to transport the traffic. The volume of traffic transported clearly affects the network load. The traffic matrix is a random matrix with uniformly distributed entries that is regenerated at each simulation run. The matrix is chosen so that for a large number of simulations, the effects of the traffic distribution is levelled out even with a very general assumption (uniform distribution). The order of flow mapping also influences the performance. The effect of this mapping is abated by iteration of a fixed-point equation.

5.3.2 Node Capabilities

The nodes are initially assigned a total processing and buffer capacity that can be configured into n queues, with the resources dynamically distributed among the queues. The number n of queues is here either 3, or 2, where 3 queues correspond to no traffic aggregation and each traffic type simply is fed onto its own queue, and 2 queues correspond to aggregation of two traffic types. Note that the single queue case is trivial from the point of view of aggregation strategy, that is, *how* traffic should be aggregated. The traffic separation by queues can in principle be implemented using (for example traffic type identification) as discussed in Section 2.3.1.

Whenever traffic aggregation is disabled, the processing and buffer capacity is equally divided onto the three queues. When enabled, resources are divided proportionally to the entropy of the aggregates. Traffic is represented by some capacity unit, which can be thought of as packets. Simulation is carried out for traffic traces of length $n = 512$, following Lindley's recursion

$$Q_{t+1} = \max\{0, Q_t + A_t - S_t\}, \tag{5.1}$$

where Q_t is the buffer content, A_t denotes arriving packets and S_t serviced packets between time t and $t + 1$. Whenever the number of arriving packets exceeds the free buffer space, so that $A_t > B - Q_t$, loss of $A_t - (B - Q_t)$ packets occur [33][54]. Assuming unit time increments, the queue delay is calculated as the expected queue length, determined for each queue. Letting $p(k)$ denote the probability of finding k packets in the buffer gives the delay

$$\boldsymbol{E}(d) = \sum_{k=1}^{B} kp(k) \tag{5.2}$$

Packet loss is measured by counting the number of packets exceeding system capacity, and throughput are the number of forwarded packets. The first equation can be taken as a definition of Lindley's equation. It means that the queue content at time t equals the previous queue content plus arrived traffic minus departed traffic, with the condition that the queue can never be negative. The second equation is a standard result for the mean number in a system [33][54][61].

5.3.3 Traffic Distribution

For each simulation scenario, the end-to-end traffic distribution matrix – expressed as a stochastic matrix – is simulated. For a stochastic matrix, the row sum must equal unity. The minimum assumption on the distribution is a uniform distribution $p_{ij} \sim U(0,1)$ of end-to-end destination probabilities, ensuring that the row sum is unity. The uniform distribution may not be appropriate for real networks, but in order to obtain as general results as possible, a minimum of network specific assumptions needs to be imposed. The same traffic distribution is used for each traffic type, for simplicity.

5.4 Traffic Simulation

The traffic is simulated independently for each type, and new traffic simulations are carried out for each new scenario and node. The three traffic types Poisson, MAP and fBm are simulated using the same traffic parameters in all cases to make the comparison of aggregated traffic more discernable. The time scale is a generic time step reflecting the packet arrival intensity and processing capacity of the nodes. To simplify the simulated queue characteristics and traffic aggregation, we express the work load as a number of packets, each assumed to be of equal length are expressed. These packets are generated by one of the three traffic processes (Poisson, MAP, or fBm) the servers are assumed to process packets with a constant bitrate, without any priority between traffic types. Using such an experimental framework, the traffic in a network is mainly determined by

- The intensity of the traffic types,
- The traffic distribution on a network level, given by a traffic matrix,
- The server capacities, and
- Traffic aggregation and routing strategies.

To study the performance gain on a network level as free from market specific assumptions as possible, the offered traffic intensity constant and node capacities uniform are kept. The traffic distribution is uniform in order to stochastically spread its effect. The important quantity is the quotient of the total mean arrival rate to the server processing capacity (Equation 3.1), which should be close to the node capacity in order to obtain performance measures of interest.

5.4.1 Simulation of Poisson Processes

The Poisson arrivals are simulated using the following algorithm. Let $\{N(t): t \geq 0\}$ be the counting process of a Poisson process with rate λ. Then $N(1)$ is Poisson distributed with mean λ. Let $Y = N(1) + 1$ and $t_n = X_1 + X_2 + \cdots + X_n$ denote the n^{th} arrival of the Poisson process, where the X_i are independent and identically distributed according to an exponential distribution with rate λ. Note that $Y = \min\{n \geq 1 : t_n > 1\} = \min\{n \geq 1 : X_1 + X_2 + \cdots + X_n > 1\}$ is a stopping time. Suppose the inverse transform method is used to generate independent and identically distributed exponential inter-arrival times X_i. Then can represent the variables as $X_i = -\left(\frac{1}{\lambda}\right) \ln(U_i)$. Re-writing the expression for Y gives

$$Y = \min\{n \geq 1 : \ln(U_1) + \ln(U_2) + \cdots + \ln(U_n) < -\lambda\} \qquad (5.3)$$

$$= \min\{n \geq 1 : \ln(U_1 U_2 \ldots U_n) < -\lambda\} \qquad (5.4)$$

$$= \min\{n \geq 1 : U_1 U_2 \ldots U_n > e^{-\lambda}\} \qquad (5.5)$$

Therefore simulate Y by generating independent uniformly distributed variates U_i, and taking the product of these variates until it first falls below $e^{-\lambda}$. The number of uniformly distributed variates in the product yields Y. Then get the desired Poisson variate as $X = N(1) = Y - 1$. Traffic traces of size 512 are generated in each simulation scenario.

5.4.2 Simulation of Markovian Additive Processes (MAP)

The MAP can be used to model and simulate bursty traffic, such as video sources. The general model has a number of states, where the transition between the states is controlled by a Markov chain. In this study, the MAP have two states only, a silent and an active state, denoted d and a, respectively. Denoting the traffic intensity by X_t, the Markov chain controlling the transition between the two states is defined by the transition probabilities

$$a = \boldsymbol{P}(X_t = 1 | X_{t-1} = 0), \qquad (5.6)$$

$$d = \boldsymbol{P}(X_t = 0 | X_{t-1} = 1). \qquad (5.7)$$

The Markov chain is then represented by the matrix

$$M = \begin{pmatrix} 1-a & a \\ d & d-1 \end{pmatrix}, \tag{5.8}$$

so that the steady state probabilities are given by

$$\pi_d = \frac{d}{a+d} \tag{5.9}$$

$$\pi_a = \frac{a}{a+d}. \tag{5.10}$$

The smaller the conditional probabilities a and d, the burstier is the resulting traffic. At each time instant, the conditional probabilities of the sources being silent or active are given by (5.6)-(5.7), keeping track of the states of the process at each time instant.

The process simulation uses a training period to "burn in", and a block of arrivals, which amounts to aggregation of individual MAP sources. In the simulation, aggregation of size 100 has been used. Traffic traces of size 512 are generated in each simulation scenario. During one time step in the queue, this thesis can simulate state change of the MAP by comparing the probabilities in the Markov chain with a uniformly distributed random number $u = U(0,1)$. If the chain is in active state and $u < a,$ it is switched to silent state. When in active state, the total packets of a block, say k packets, are added to the queue. When fed to the queue, a maximum amount of s packets are processed and removed from the queue in each time step. Simulation of queues with finite buffers are easily accomplished in discrete time, and various performance metrics can be calculated by introducing counters into the simulation process.

5.4.3 Simulation of fractional Brownian motion (fBm)

A fBm can be simulated using the Cholesky method. It uses the Cholesky decomposition of the covariance matrix, $C = \sum\sum'$, where $\sum$ is a lower triangle matrix of the covariance matrix given by Equation (3.23). It can be shown that such a decomposition exists whenever the autocovariance matrix is positive definite (and symmetric, which is true by its construction). The simulated packet arrivals are then obtained by multiplying the matrix $\sum$ by a vector of independent normal standard random variables η of suitable size, that is

$$\boldsymbol{X} = \sum\eta \tag{5.11}$$

It is necessary to restrict the resulting values to be equal to or greater than zero, since traffic obviously cannot be negative. The traffic load is chosen so that the peak rate exceeds the service rate s of the queue in order to obtain interesting node performance parameters, whereas the mean arrival rate must be slower than s. The Markov parameters are chosen as in [29] with $d = 0.072$ and $a = 0.028$. With a router capacity of $s = 60$ packets per time unit, this choice of parameters generates traffic with a peak rate exceeding s, but with a mean rate less than s. This exact method of simulating fBm is purely algebraic, based on the Cholesky decomposition of the covariance matrix. The simulated process can be negative, but such results are discarded in the simulation.

5.5 Traffic Aggregation

The traffic aggregation logic is based on comparing the entropy for aggregated and non—aggregated traffic streams. With three traffic types and two queues, two traffic types can be aggregated, which can be done in three ways, and compare the resulting entropies. The entropy is used in two steps. Firstly, the traffic types to aggregate is determined. The decision of which traffic types to aggregate is made so that the aggregate has an entropy as close as possible to the entropy of the non-aggregated traffic. The entropy of the single traffic types in different simulations is such that for Poisson traffic, it is high and nearly constant, whereas for fBm, the average is lower but with high variability. The entropy of MAP traffic is somewhere in between. Therefore, decision of aggregation is mainly driven by the behaviour of the fBm traffic. Next, the aggregated and non-aggregated traffic is mapped onto two queues in the node. Here, the entropy is used as a scaling parameter, where the capacity is allocated in proportion to the entropy. This is a simple heuristic, which is applied both to processing capacity and buffer space. More sophisticated control logic can be devised, but this simple principle suffices to study the effect of traffic aggregation on heterogeneous traffic.

5.6 Routing Strategy

For the analysis on a network level, this thesis assumes that paths can be chosen optimally with respect to delay. This means that an end-to-end connection is set up on the route which incurs the least amount of delay. Note that a shortest path can be defined with respect to any numerical weight, not only physical distance. Such weights may be cost, delay or some reliability measure. To find a minimum delay path, this thesis uses the algorithm by Dijkstra, see for example [29].

In finding an end-to-end optimal path, start from the first end node s, and scan its neighbours j for the lowest weight. Letting $d\ (s,j)$ be the total weight going from s to j, the algorithm successively uses the relation

$$d\ (s,j) = \min_{i \in S}\left(d\ (s,i) + d_{ij}\right), \tag{5.12}$$

where S is the set of nodes and d_{ij} is the weight on edge (i,j). The algorithm compares the minimum weight paths going from s to any node i, adding the weight going from i to j. Dijkstra's algorithm uses weights on edges, whereas the delay actually incurred in the nodes, so node delays need to be mapped onto edges. This can be done since the delays, just like distances, are positive and additive. The minimum delay route is determined locally, and by using this route for a particular end-to-end connection, the node delays would change. Since the global properties of minimum delay routing is the one to focus, iterative procedure is needed where minimum delay routes are re-calculated as traffic are added to the already determined minimum delay routes.

To find the optimal end-to-end routing from a network perspective, we use a fixed-point equation is used

$$F(\Delta_{min}) = \Delta_{min}. \tag{5.13}$$

Such fixed-point equations are commonly used to analyse blocking networks [29], where $F(.)$ is a function of the total blocking on a route, given by Erlang's B-formula. The algorithm (and equation) is inspired by the Erlang fixed point theorem, which is based on a fixed point iteration [63]. The blocking is a convex function of the load, just as the delay. Assume that at least one such fixed point exists and that the equation converges to this point (or points). This assumption is partly justified by the fact that the node delays are limited and a function of traffic load. Thus, assuming delays incurred by the traffic offered to the routers, this thesis sequentially determines the end-to-end minimum delay paths, adding traffic to the nodes along the path corresponding to an end-to-end traffic matrix. This gives the initial solution to the fixed-point equation (5.13). The minimum delay routes are then successively re-calculated, using the previously defined edge delays, updating the traffic on the routers along any computed path, and updating the edge delays accordingly. Interestingly, the fixed-point equation converges in many cases to two solutions, leading to a hysteresis in the network delay. This stability phenomenon has also been observed in blocking networks [20][29]. In blocking networks, which lack centralised control, trunk reservation has been suggested as a stabilising strategy [20]. In this case, the centralised logic selects the best network routes, using delay measurements from the nodes.

5.7 Quality of Service Evaluation

A node consists of two or three queues, which are simulated independently in the sense that once resources are assigned to a specific queue, these cannot be shared by another queue in the node. The queues are fed with the aggregate or single type traffic, represented by vectors of arriving packets. At each time instant, the queues are governed by Lindley's equation (4.1) with the constraint of limited buffer space B

$$Q_{t+1} = \begin{cases} \max\{0, Q_t + A_t - S_t\} & if\ Q_t + A_t - S_t \leq B \\ 0 & otherwise \end{cases} /., \tag{5.14}$$

which can be expressed $Q_{t+1} = \min\{\max\{0, Q_t + A_t - S_t\}, B\}$. The arrivals A_t are given by the vectors of offered load, and $S_t = s$ is a constant number of packets processed in each time step.

In the router simulator, at each time instant the number of lost packets is being counted, that is $Y = \max\{0, \hat{Q}_t - B\}$, where $\hat{Q}$ is the queue length in a queue with infinite buffer. These are the packets in excess of router capacity, that is, the packets that are being processed during the time step, and the free buffer space. By recording the number of packets in the system at each simulation step (the state occupancy), the delay can be determined. Whilst, from the recorded state occupancy, the state probabilities p_n, and the delay is given by

$$\delta_i = \sum_{n=0}^{C} 1 . p_n \tag{5.15}$$

packets per time step, and where C is the total router capacity. In the simulation, a M/D/1 queue is used, so the single queue services 1 packet per time step (and since it is deterministic, exactly one packet is processed in each small time interval). The difference in performance is measured using the standard "error" measure

$$\Delta m = \frac{m_o - m_r}{m_r}, \tag{5.16}$$

where m_o is the observed metric and m_r is the reference metric. When m_r is zero, the metric degenerates, and sets as

$$\Delta m = \begin{cases} 1 & if\ m_r = 0\ and\ m_0 > 0 \\ 0 & if\ m_r = 0\ and\ m_0 = 0 \\ -1 & if\ m_r = 0\ and\ m_0 < 0 \end{cases} \tag{5.17}$$

As a general rule, the performance measure is being improved so that $0 \geq \Delta m \geq 1$ to avoid large values whenever underlying metric is small. This also allows to represent the difference in performance as a percentage figure. In order to make the percentage figures comparable, the average number of affected packets is scaled; the number of buffered, lost and forwarded packets to the total number of packets, respectively.

5.8 The Effect of Traffic Aggregation

The effect of traffic aggregation on need level is analysed, comparing two aggregation strategies with the case of no aggregation, and recording delay, packet loss and throughput. Each case is simulated 1000 times, where each case consists of the three traffic types of size 512 time steps. Each case therefore consists of 512,000 simulation points. Traffic aggregation in combination with optimal routing is discussed in Section 5.9.

5.8.1 Node Level Traffic Aggregation

Traffic aggregation on node level includes the operations of merging two of the three traffic streams and configuring the node resources proportionally. This thesis compares two cases of traffic aggregation; aggregation of real-time traffic, always aggregating Poisson and MAP traffic, and dynamic aggregation, where traffic types are aggregated so that the entropy of the two resulting traffic streams are equal as possible. The entropy-based traffic aggregation is analysed in isolation to be able to assess its efficiency and impact on performance on node level. The aggregation decision is based on the entropy of each of the streams of voice, video, and data traffic. Aggregation is performed so that the entropy of the traffic load in each channel is as equal as possible.

For each queue with capacity *C* (including buffer space),

$$\sum_{j=1}^{N} n_j \alpha_j \leq C, \tag{5.18}$$

where *N* is the number of sources in an aggregate, and α_j is the effective bandwidth of the traffic flow j. Do not try to estimate the statistical multiplexing gain, but the gain in the QoS triple consisting of delay, packet loss and throughput. In each simulation, the total and maximum delay, the total and maximum packet loss and the total and maximum throughput are collected, and compares that with the situation where no aggregation is performed, using three queues, and resources are allocated equally between these queues. The improvement is referred to as *gain*. The gain in throughput and performance are compared by comparing the strategies of

(1) No aggregation, static mapping of traffic to equally configured queues
(2) Aggregation of real-time traffic sources (Poisson and MAP) mapped onto queues with proportionally scaled resources
(3) Dynamic aggregation, where traffic is aggregated to make the entropy as equal as possible between the two queues, and proportionally scaled resources

By measuring the change in total and maximum delay, packet loss and throughput when compared to no aggregation, this thesis has descriptors of the relative performance of each strategy. Clearly, the larger relative improvement the better. There is, however, justification to order the performance statistics after importance, having;

(1) ***Maximum delay***. The statistic measures the worst end-to-end delay, which is the main performance indicator to be improved. It is, however, closely related to the total delay, and the former should not be improved at the expense of the latter.
(2) ***Maximum and total packet loss***. For real-time services, packet loss can lead to severe quality degradation. For best effort traffic, however, packet loss is typically compensated by retransmissions controlled by TCP. Also note the duality between delay and packet loss; the larger the buffer, the longer delays and the smaller packet loss. The change in total packet loss – if negative – should be small due to its adverse effect on real-time traffic. Packet retransmission is not considered in the study, since it is traffic dependent and leads to a bias in the throughput figures.

(3) ***Total throughput***. The statistic represents the utility of shared resources, and the larger the increase, the better. The improvement is achieved through statistical multiplexing of the traffic or increased buffering.

5.8.2 Real-Time/Non Real-Time Traffic Aggregation

It has been suggested in the literature [9] that aggregation of real-time sources would give the best QoS. The aggregation strategy is to aggregate real-time traffic, justified by their similar characteristics and QoS demands, while keeping non real-time traffic separate and denote this strategy RT/NRT for brevity. The node resources are scaled in proportion to the entropy of the two traffic streams. On node level, a comparison of the RT/NRT with the case of separate queues is shown in Figure 5.2.

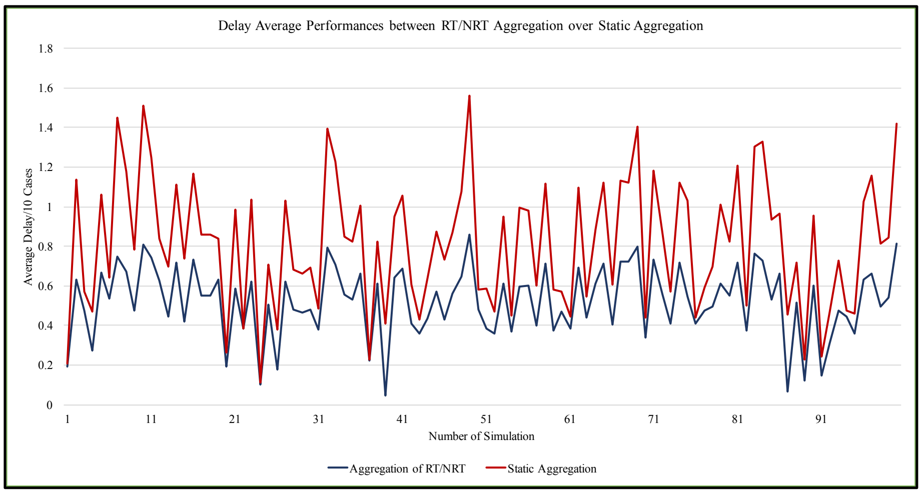


Figure 5.2(a): Delay Average Performance between RT/NRT Aggregation over Static Aggregation

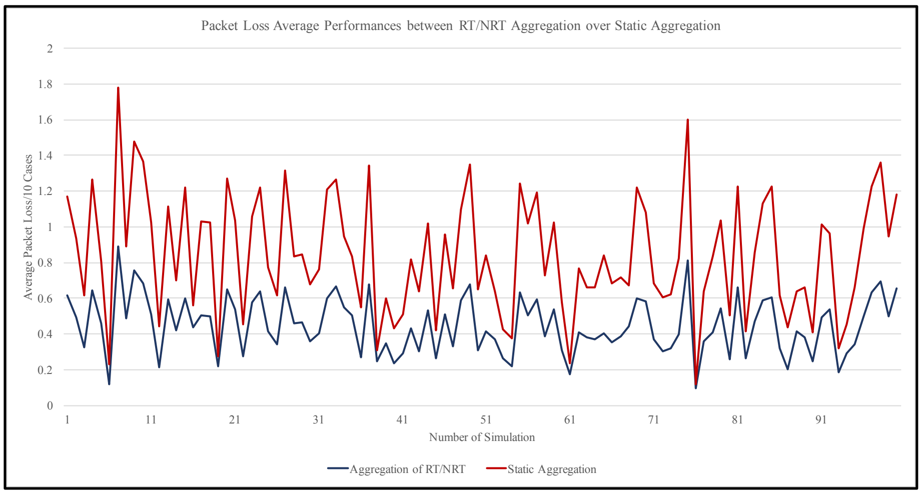


Figure 5.2(b): Packet Loss Average Performance between RT/NRT Aggregation over Static Aggregation

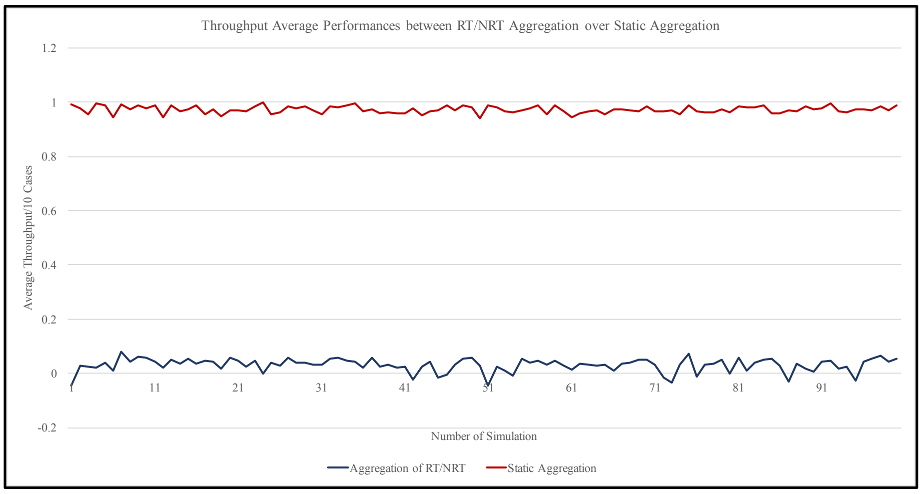


Figure 5.2(c): Throughput Average Performances of RT/NRT Traffic Aggregation and Static Aggregation

These graphs illustrate the results of independent simulations and is not a time average. Therefore, the results vary noticeably. The purpose is to show the relation between static and dynamic aggregation. The results indicate an increase in delay and maximum packet loss, by the negative relative changes, and an increase in maximum throughput. Since both the average and the maximum delay increases, the overall performance of real-time traffic is deteriorating when compared to queueing without traffic aggregation. This is a consequence of the resource scaling. When more traffic is buffered, the delay increases. By the effect of traffic scaling, the expectation of that processing capacity should be increased faster than buffer size using this strategy.

The increase in maximum throughput is associated with the aggregation of real-time traffic, and can be attributed to statistical multiplexing of real-time sources. This is an expected result, as aggregation of Markovian sources and resources improves the efficiency. At the same time, since the change in average throughput remains very small, the other queue must experience a similar decrease in throughput. There is no or small net gain in throughput in this case. While the maximum throughput increases, there is negligible improvement in total throughput which indicates a shift in processing resources to the real-time traffic.

5.8.3 Dynamic Traffic Aggregation

A comparison of the dynamic aggregation strategy with no aggregation is shown in Figure 5.3.

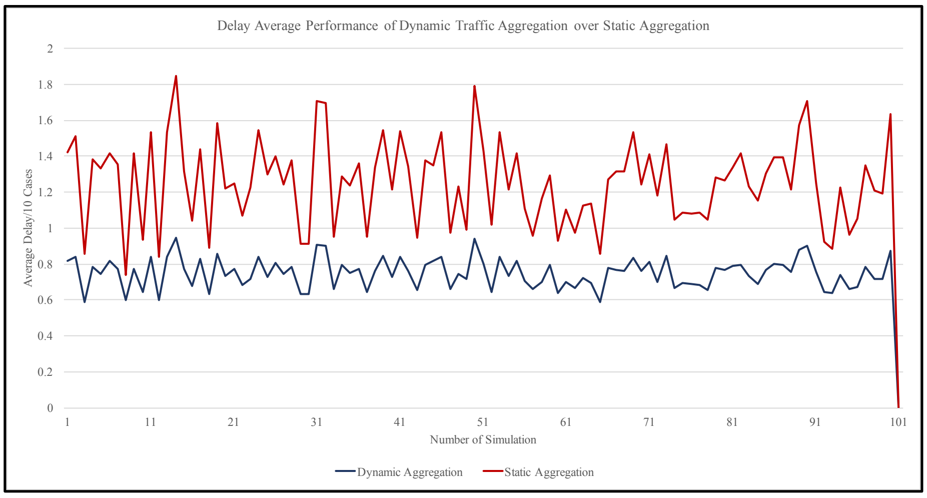


Figure 5.3(a): Delay Average Performances between Dynamic Traffic Aggregation and Static Aggregation

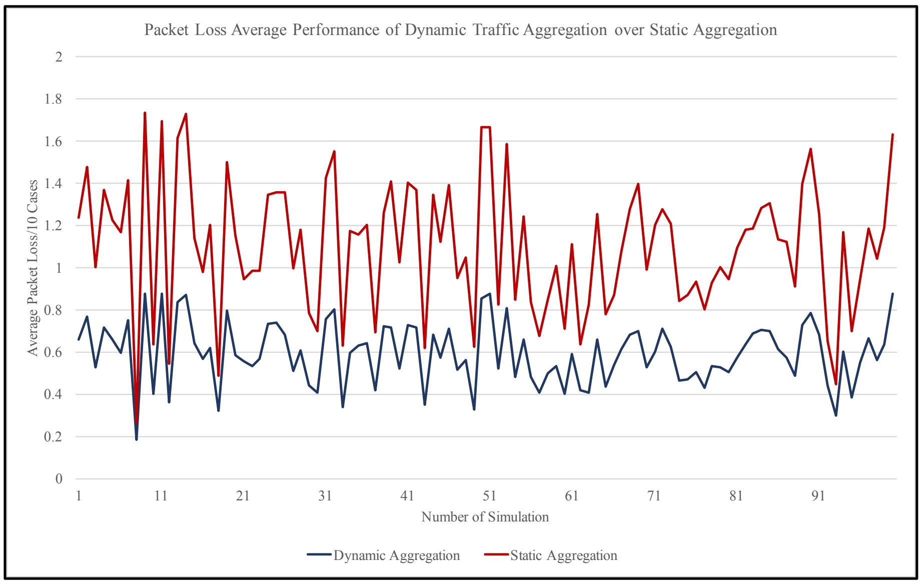


Figure 5.3(b): Packet Loss Average Performances between Dynamic Traffic Aggregation and Static Aggregation

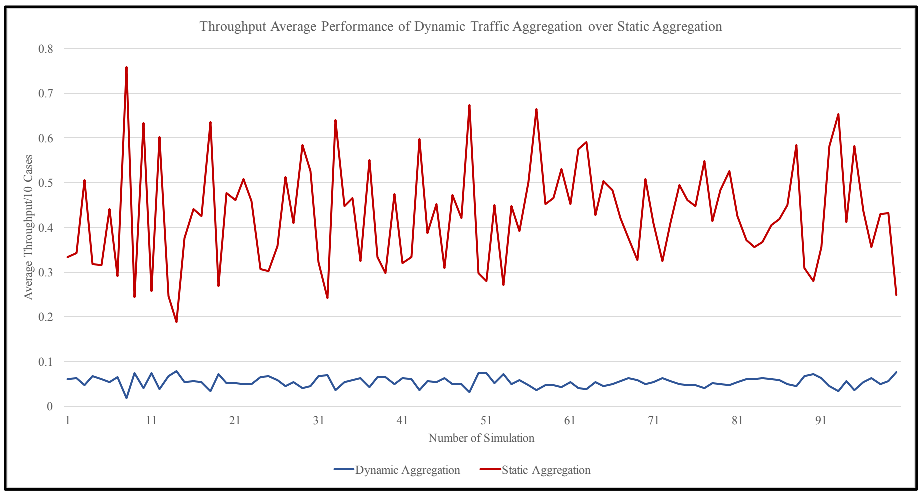


Figure 5.3(c): Throughput Average Performances between Dynamic Traffic Aggregation and Static Aggregation

A fully dynamic traffic aggregation enabled by SDN is more flexible than any deterministic strategy, and therefore has the potential to be much more efficient. In this strategy, this thesis aggregates the two traffic streams that gives an as even value of entropies as possible. There is a discernable improvement in total and maximum packet loss as well as maximum and throughput. Note that the gain in total throughput is higher and the gain in maximum throughput is lower than for RT/NRT aggregation. The increased throughput and improved packet loss follow from a more agile resource scaling strategy.

5.8.3.1 Relative Difference between Dynamic and RT/NRT Aggregation

It is instructive to compare the dynamic with the RT/NRT strategy. The result is shown in Figure 5.4.

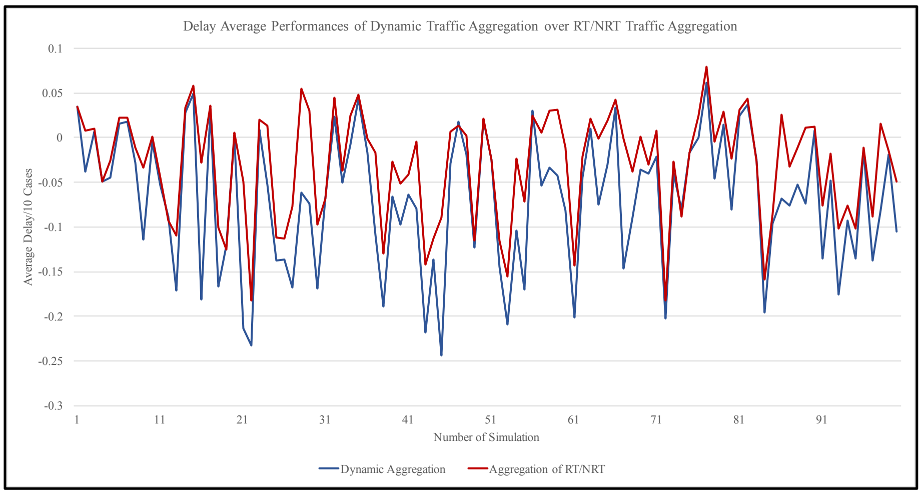


Figure 5.4(a): Delay Average Performances between Dynamic Traffic Aggregation and RT/NRT Aggregation

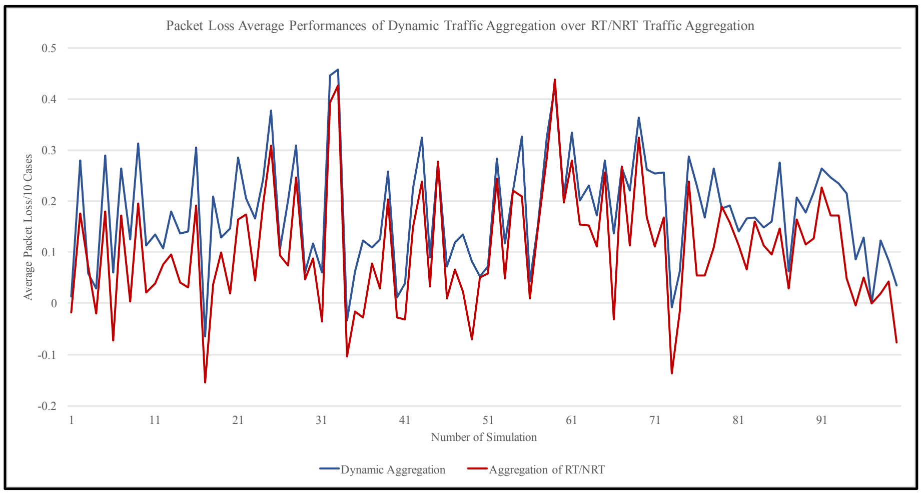


Figure 5.4(b): Packet Loss Average Performances between Dynamic Traffic Aggregation and RT/NRT Aggregation

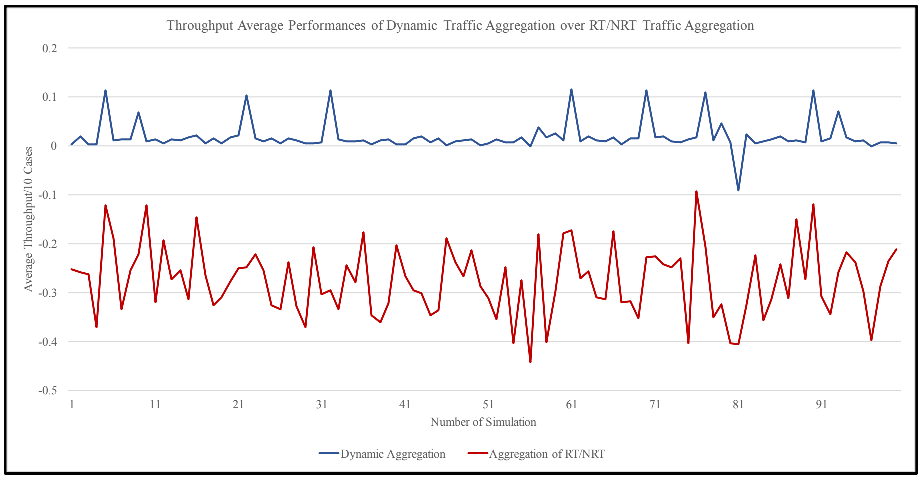


Figure 5.4(c): Throughput Average Performances between Dynamic Traffic Aggregation and RT/NRT Aggregation

The dynamic aggregation strategy shows a net improvement of packet loss and a slightly higher total throughput over the RT/NRT strategy. The increase in total throughput and decrease in maximum throughput in the comparison indicates a higher degree of fairness in QoS provisioning between the traffic streams. A higher maximum throughput is obtained when real-time traffic has precedence, as an effect of the smaller fluctuations on longer time scales.

The improvement in total delay can be motivated as follows. The aggregation of Markovian sources (audio and video) tends to be well-behaved, and therefore benefit from economy of scale in a larger queue. The average delay should therefore be lower and the gain positive. The slight deterioration for the maximum delay, likely to be that of long-range dependent (LRD) traffic may be due to resource scaling. That is, after the aggregation, the LRD traffic may be allocated less resources than it would have in it's static dedicated queue. This would also mean that more resources are allocated to the Poisson-MAP aggregate, lowering the delay further. The variance is large as a consequence of the self-similar traffic.

To summarise, the simulation results show a marked improvement in terms of packet loss and/or throughput. The presentation of these statistical figures is justified by the fact that the variability of self-similar traffic is decreased by buffer limitations, which potentially would affect LRD traffic at all nodes along its path. The main idea is that resource utilisation improves with aggregation, so that *similar* traffic types share a queue, and thereby benefit from statistical multiplexing. For dissimilar traffic types, on the other hand, starvation of resources of one type by another may decrease performance. It is therefore suggested to separate self-similar traffic from Markovian traffic, and allocation resources in proportion to the load.

5.9 The Effect of Optimal Routing

Controlled flow routing can be used to further take advantage of the resources available in the network. This thesis investigates routing optimised with respect to the total delay, that is, the sum of delays incurred at all nodes in the network, and sum of the maximum delays. For the search of an optimal flow assignment, the algorithm by Dijkstra is used. Dijkstra's Algorithm finds the shortest path between two nodes s (the start node) and t (the destination node) by scanning the neighbours at each step, and builds a tree of shortest paths to all nodes from s. It uses the "distances" of the edges, where the distance can be any positive real number, which be called as the edge weight. Thus, the weight can be Euclidean distances, number of hops, or, with a slight modification, delay.

Initially, all weights are set to ∞. Starting from the start node s, the algorithm finds the nearest neighbour of s by comparing the edge weights. It then progresses to the nearest neighbour j and repeats the scanning. At each step, the weights are updated to be the smallest weight encountered so far, resulting from all paths traversed up to the current step. The pseudo code of Dijkstra's algorithm is as follows.

Algorithm 5.0.1 (Dijkstra's Algorithm)**.**

Given a (undirected) graph $G = (V, E)$, non-negative edge costs $c(.)$ and a starting vertex $s \in V$.

```
STEP 0:
        Set $S := \{s\}, l(s) := 0, l(i) = \infty, i \in V, i \neq s$
STEP 1 to |V| - 1:
        While $S \neq V$ do
            find $x$ such that $l(x) = \min\{l(y): y \neq S\}$
            set $S := S \cup \{x\}$
```

```
for all y ∈ $V - S$ do
  $l(y) := \min\{l(y), l(x) + d_{xy}\}$
end end
```

Output: the shortest total weights from s to all $i \in V$. •

In this algorithm, two important observations are made. Firstly, the shortest path need not be unique. However, all shortest paths must have the same total weight in the network. This thesis is interested in any such shortest path, by looking for a global optimum. Secondly, in its original form, the algorithm does not give the shortest path, but only the sum of weights along this path. The actual paths can be found by labelling each edge, and adding an edge *(i, j)* to an edge set *E* each time such an edge is traversed and a node *j* is added to the set *S.* After the destination node *t* has been reached, backtracking is used from *t* through the edges in *E* to find the path back to *s*. Now, in order to make two modifications to find the optimal flow assignment with respect to delay, firstly, the delays, which are properties of the nodes, need to be mapped to edge weights. Secondly, since the algorithm finds the shortest path with all weights given, an iterative procedure is needed to find a global flow assignment in the network where weights depend on the assignment. Please refer to Figure 5.1 as illustrated.

The first modification is based on the additivity of delays. Since the delays are additive, the total delay of an end-to-end connection can be expressed as

$$\Delta_{st} = \sum_{i=s}^{t} \delta_i, \tag{5.19}$$

where i starts and ends at the respective end points $\{s, t\}$. In this expression, Δ_{st} is the total end-to-end delay, and δ_i is the delay incurred in node i. By setting the edge weights to $d_{ij} = \frac{\delta_i}{2} + \frac{\delta_j}{2}$, see that

$$\Delta_{st} = \frac{\delta_s}{2} + \left(\frac{\delta_s}{2} + \frac{\delta_i}{2}\right) + \left(\frac{\delta_i}{2} + \frac{\delta_j}{2}\right) + \ldots + \left(\frac{\delta_k}{2} + \frac{\delta_t}{2}\right) + \frac{\delta_t}{2} =$$

$$\frac{\delta_s}{2} + d_{si} + d_{ij} + \ldots + d_{kt} + \frac{\delta_t}{2}.$$

The first and the last term do not influence the shortest path, since the start and end notes are fixed. To obtain a numerical value of the total delay, simply add these two terms to the results from Dijkstra's algorithm.

Now, the minimum delay route can be determined locally, for a particular end-to-end connection. This choice of path, however, assumes that the delays remain fixed in the network. Since in this thesis the global properties of minimum delay routing is studied, an iterative procedure is needed where minimum delay routes are re-calculated as traffic are added to the already determined minimum delay routes. The end-to-end distribution of traffic is given by an $n \times n$ probability matrix, where each row signifies the node i traffic is originating from and each column the node j, a percentage p_{ij} of the traffic is addressed to. This is called the *traffic matrix* in the following. For this matrix must have $p_{ij} = 0$ and $\sum_i p_{ij} = 1$. In each simulation, the p_{ij} are drawn from a uniform distribution with the condition that $\sum_i p_{ij} = 1$. This ensures that the assumptions on the end-to-end traffic distribution is minimal.

To find the globally optimal end-to-end routing from a network perspective, the fixed-point Equation (5.13) is used.

$$F(\Delta_{min}) = \Delta_{min}. \tag{5.20}$$

After initial generation and dynamic aggregation of traffic, the fixed-point equation iteratively simulates and maps delays as weights onto the edges, computes the minimum delay paths, and aggregates the traffic along these paths.

Algorithm 5.0.2 (Fixed-Point Optimal Route Algorithm)**.**

Given a network (graph) *G = (V,E)* with $n = |V|$ nodes, an n-dimentional traffic vector **t** and an $n \times n$ traffic matric *T*.

```
STEP 0:
        Populate $\boldsymbol{t}^0$ by generating traffic and apply dynamic traffic aggregation.

STEP 1 until convergence:
        for each node $i \in V$ do
          determine delays $d_i$ incurred
        end for each link $(i, j) \in E$ do
          set weights $w(i,j) = \frac{\delta_i}{2} + \frac{\delta_j}{2}$
        end for each node $i \in V$ do
          for each node $j \in V$ do
            Find minimum delay paths $P_{min}^{(i,j)}$ using Dijkstra's algorithm
            Set $\boldsymbol{t}(i) = \boldsymbol{t}^0(i) + \sum_j T\,(i,j)\boldsymbol{t}_{(j)}^{min}$   end end
    Output: minimum delay flow allocation.                                    •
```

Assumes that the fixed-point algorithm converges. Should it happen that the algorithm shows oscillating behaviour, the solution with the best characteristics is chosen. Since the minimum delay paths may be longer than the shortest paths in terms of hops, the effect of optimal routing next is being analysed.

5.9.1 Traffic Aggregation Under Minimum Total Delay Routing

Considering congestion control on a network level, optimal routes through the network can be chosen minimise the number of hops (ordinary shortest paths), the total end-to-end delay, or sum of maximum end-to-end delays. The gain is determined as the performance improvement with traffic aggregation at the nodes and optimal routing, compared to the situation with traffic aggregation at the nodes and shortest path routing. It should be noted, that in the network scenarios, nodes carry not only its own traffic, which is a random quantity.

The performance gains for the case of routing minimising the sum of end-to-end delays are shown in Figure 5.5. As for the gain due to traffic aggregation in individual nodes, the largest gain is in packet loss. The distribution of gains, however, depends on the node configurations and the traffic levels. The positive gains in delay, packet loss and throughput are significant with *p*-value less than 0.3%. The simulations result of the system are the average of 100 simulations. From the data set, the average and variance are obtained. Two-sided confidence interval is used with z = 3, corresponding to a probability of 99.74% from a standard normal distribution table. Consequently, the confidence interval is not significant at level p = 1.00-0.997 = 0.003.

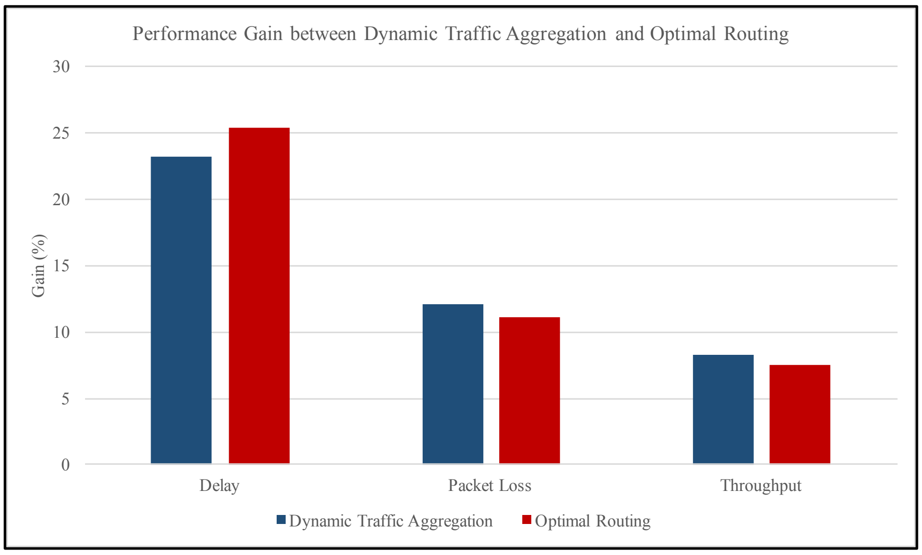


Figure 5.5: Performance Gain with Dynamic Traffic Aggregation and Optimal Routing

This result is compared with two cases: RT/NRT routing under optimal routing and dynamic aggregation under shortest path routing. The optimally routed RT/NRT aggregation is shown in Figure 5.6. A marked improvement in delay is showing, but declining in packet loss and throughput.

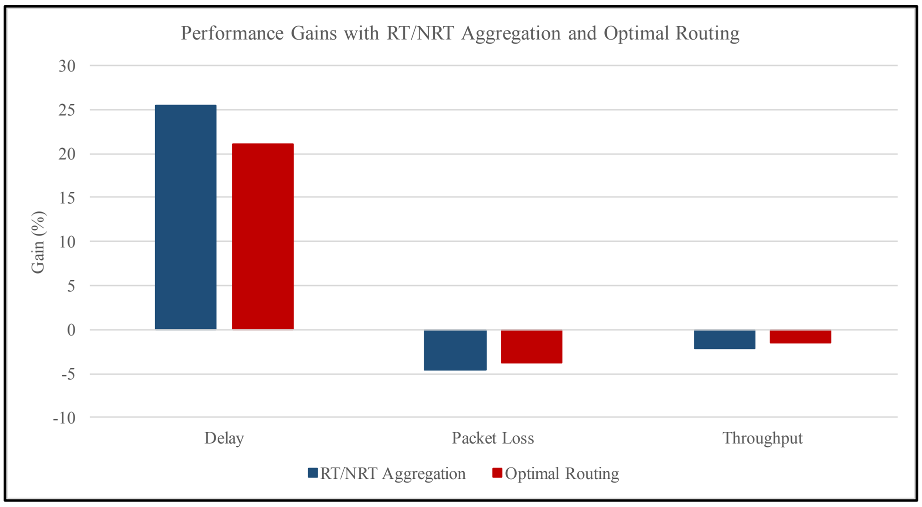


Figure 5.6: Performance Gains with RT/NRT Aggregation and Optimal Routing

5.9.2 Dynamic Traffic Aggregation Under Shortest Path Routing

The effect of dynamic traffic aggregation under optimal routing compared to simple shortest path routing is shown in Figure 5.7. These graphs illustrate the results of independent simulations and is not a time average. The scales are negative (relative the global optimum), and therefore the result for optimal routing is better than the result for static routing. Under optimal routing, the total delay and throughput is worse than for optimal routing, following from the fact that the average paths are longer than in shortest path routing. However, maximum delay and packet loss is improved, indicating an improvement in the worst end-to-end QoS. As a conclusion;

(1) When using optimal routing, the overall performance gain is better for dynamic traffic aggregation than for RT/NRT routing. This follows from the better resource utilisation in the nodes.

(2) Optimal routing can improve the worst end-to-end QoS at the expense of average values, following a dynamic load distribution in the network.

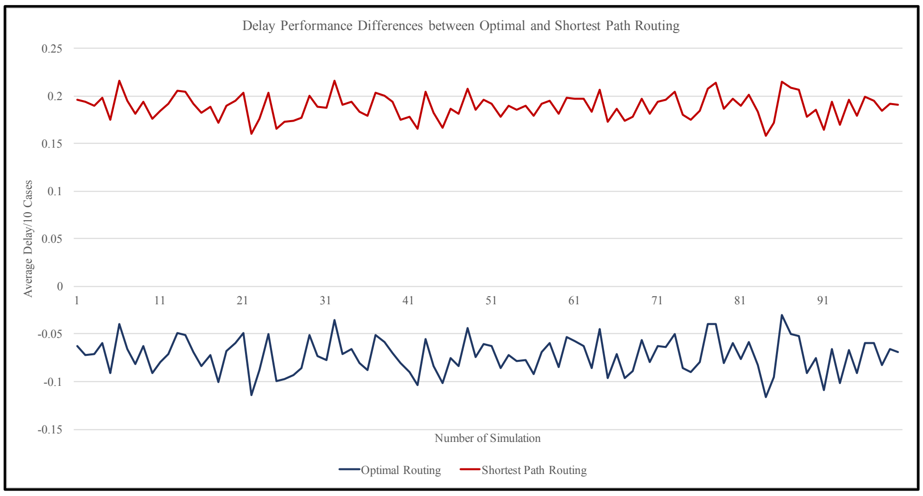


Figure 5.7(a): Delay Average Performances of Dynamic Traffic Aggregation Under Optimal Routing and Shortest Path Routing

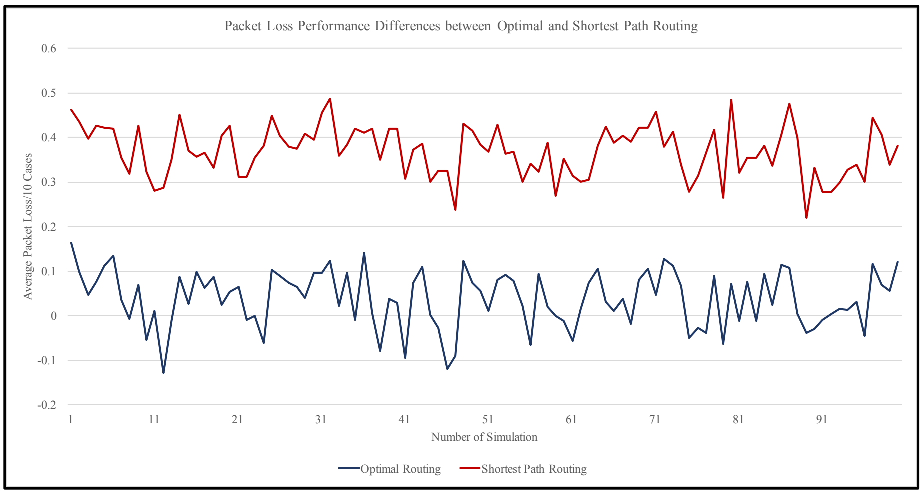


Figure 5.7(b): Packet Loss Average Performances of Dynamic Traffic Aggregation Under Optimal Routing and Shortest Path Routing

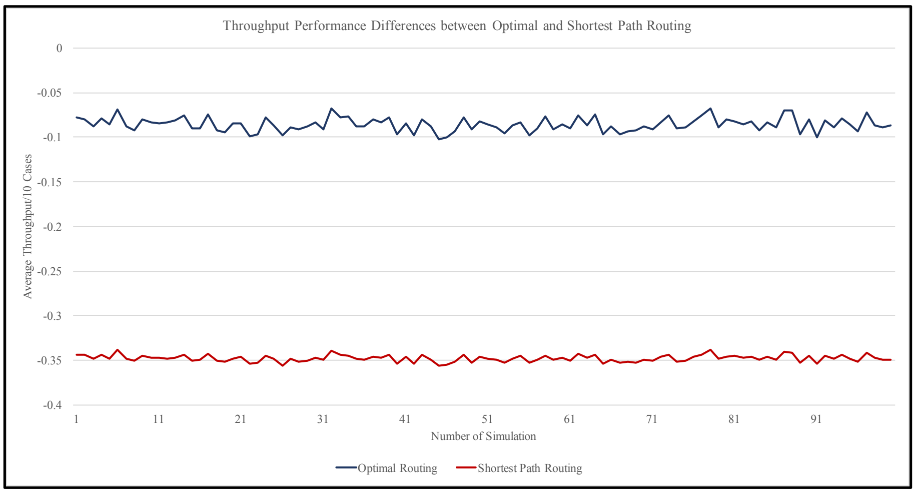


Figure 5.7(c): Throughput Average Performances of Dynamic Traffic Aggregation Under Optimal Routing and Shortest Path Routing

5.9.3 Traffic Aggregation Under Minimum Maximum (Min-Max) Delay Routing

When the optimal routes are chosen to minimise the maximum end-to-end delay, the gain in total and maximum delay is higher than in the previous case, as shown in Figure 5.8. By comparing Figures 5.5 and Figure 5.6, it can be seen that routes optimised with respect to maximum end-to-end delay gives higher overall gain. Using the maximum end-to-end delay as optimisation criterion has some obvious advantages. Firstly, minimising the maximum delay leads to a higher degree of fairness in the network. Secondly, from an optimisation point of view, the maximum delay is a stricter and explicit condition than the total delay, which should improve the convergence of the solution to a minimum.

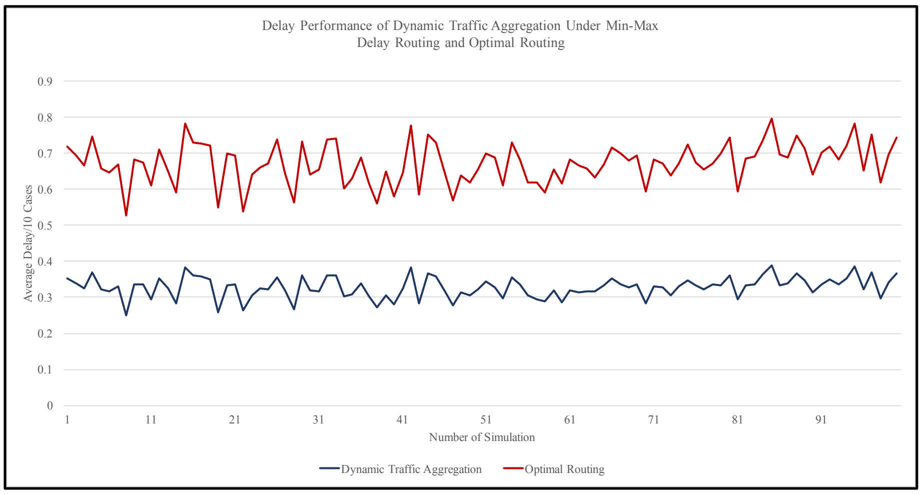


Figure 5.8(a): Delay Average Performances of Dynamic Traffic Aggregation Under Min-Max Delay Routing over Optimal Routing

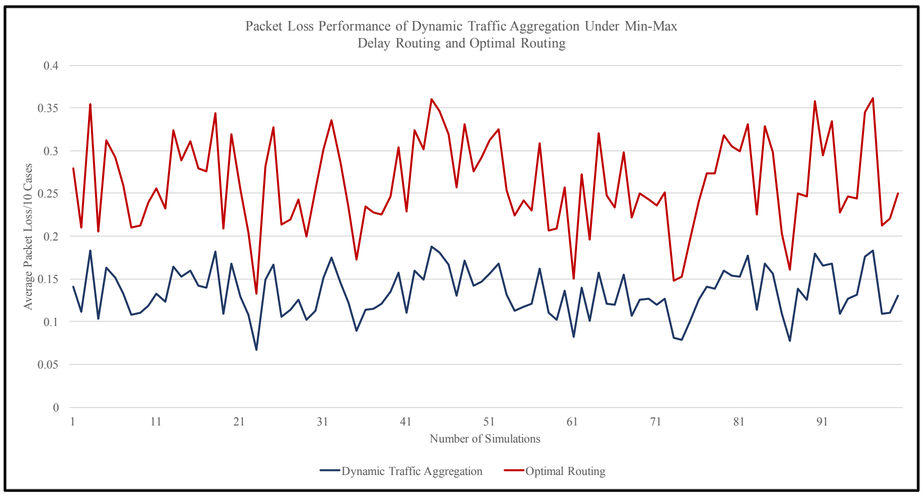


Figure 5.8(b): Packet Loss Average Performances of Dynamic Traffic Aggregation Under Min-Max Delay Routing over Optimal Routing

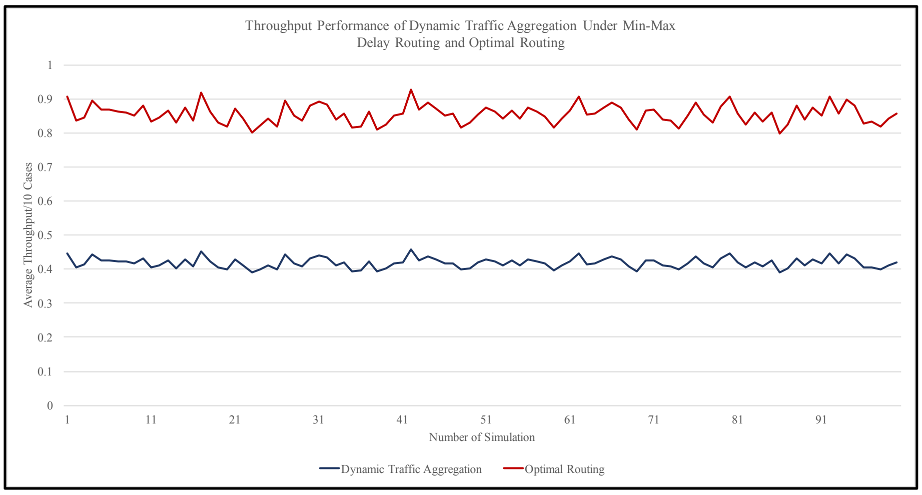


Figure 5.8(c): Throughput Average Performances of Dynamic Traffic Aggregation Under Min-Max Delay Routing over Optimal Routing

To summarise, dynamic traffic aggregation provides a more flexible utilisation of resources. At the same time, preserving fairness between traffic streams in imperative. On node level, traffic aggregation leads to gains in packet loss and throughput – in particular maximum throughput. When combined with optimal routing, a load balancing effect on network level is achieved, resulting in increased fairness without using any particular scheduling or priority principles. Routes optimised with respect to maximum delay gives better results than when optimising with respect to total delay, which is a direct consequence of improving the worst end-to-end performance.

5.10 Chapter Summary

This chapter summarise that by empirically, investigation into traffic aggregation and routing are to improve throughput and performance. We compared different aggregation and routing strategies and showed (by simulation) that aggregation in combination with Min-Max optimal routing gives substantial gain in all performance metrics, and was better than any other combination of studied actions.

Chapter 6

ALLOCATION OF NETWORK FUNCTION USING ANT COLONY OPTIMISATION

6.1 Introduction

The deployment of a centralised logic must be very resilient to failures of nodes and links. A promising strategy is therefore using NFV to allocate the logic dynamically, depending on availability and load on the nodes in the network. In this context, the network can be viewed as a "cloud" of resources, and it is assumed that a controlling entity is available to assign the control logic in an optimal fashion. If the failure probability of a node is q, and there are n nodes, the possibility to allocate the control function to any of the n nodes results in an operational probability of

$$p = 1 - q^n , \tag{6.1}$$

assuming that nodes fail independently from each other. Of course, this is a theoretical result, since the network would anyway be disconnected should a number $m << n$ nodes fail.

Resource allocation in clouds can be divided into different categories or steps. Mills et al. [35] evaluate different allocation heuristics and categorise the optimisation type based on initial placement, where new requests are allocated subject to available resources, consolidation, where a new request may modify existing allocation to achieve lower cost, and trade-off between SLA and cost. The cost/demand structure is also categorised into reservations, where the customer pays a fixed price for a service running for a specified amount of time, on-demand access, where customers put requests and the cost is dependent on utilisation, and spot market, where the price of a service also depend on demand. With resource allocation in a cloud is understood the allocation of VM to physical nodes, or hosts, where the hosts are characterised by the number of processors, and the amount of processing capacity and random access memory (RAM) (and possibly other parameters). The scheduling of tasks is not considered part of the project, but it is desirable that the framework should be possible to extend to this situation, if desired. An important step in defining an algorithm for the resource allocation problem is defining objective of the optimisation. Two different objectives of the customers and the cloud operator are identified. In a simplified setup, by assuming that for the customer, the cost is fixed, and so tries to maximise the utility of resources. For the operator, the total amount of resources is fixed, and the potential for adding more customer requests to the cloud depends on the assignment of the existing requests.

Resource allocation is Non-Deterministic Polynomial-Time ($\mathcal{NP}$)-hard, so the number of possible solutions grows exponentially with the number of servers and the number of customers. This is a variant of the bin packing problem, and can also be formulated as an integer program. In this thesis, the following terminology is used. Consider an IT cloud consisting of physical resources. The cloud is managed by the cloud operator, who fully controls the resources and how requests are assigned. Requests for resources are specifically VM, and the requesting entity (not necessarily human) is called a customer. The customer may have preferences where its request should be allocated. The final resource allocation, however, is determined by the cloud operator. In the context of the algorithm, the customer is also referred to as an *ant*.

The physical resources are residing in several levels. The lowest unit is referred to as a host. Hosts are aggregated into clusters – typically residing in the same location, and several clusters form a cloud. This thesis is not considered geographically distributed clusters, so a cluster and a cloud is essentially the same. The term cloud has been used as it is clearer from an engineering point of view. The resources together with the requests is referred to as a system when a general abstract term is needed.

6.2 Cloud Resources and Descriptors

Both the cloud resources and the requests are assumed to be described in sufficient detail so that an assignment can be made, and that this information is available to the algorithm at all times. The resources are quantifications of available physical properties, such as number of processors in the Central Processing Unit (CPU) core, CPU speed, amount of RAM, amount of disk storage space and network bandwidth. The idea of using a constantly updated database with a resource vector for each node [30], has been adopted as a general and technically appealing solutions to manage cloud state data. The network is clearly dynamic, so rather than allocating according to the physical resources of a node, it should be done with respect the instantaneously available *free* resources of a node. The result of the optimisation is an assignment of VM-node pairs.

Some of the resources are static, such as the CPU core, and can be included as side constraints (or rather, an infeasible assignment results in a zero probability). Other properties are dynamic and change with each assignment. Also here, by distinguishing between resources that set a definite limitation on the service capability, such as the amount of memory available, should the VM requirements exceed a node's capabilities, the node is considered not to be able to host that VM. The second type of resource - for example processing power or network bandwidth - scales gradually with the number of VMs. Service quality can then be seen as the expected average processing time (or throughput).

Assume that a VM can be specified in the same terms as a physical node. For simplicity, let the resource vector be the triple (core, cpu, memory), as described in [30]. Thus, a VM requirement can be directly compared to a node's available resources, and be allocated to any physical node having sufficient resources. Assume that a host can have a number of VMs, and this number is limited by the aggregate requirements on the node resources. Also that a VM occupies the resources it specifies [46], should a node not have sufficient amount of free resources, the assignment is infeasible and will be disregarded.

6.3 Optimisation Criteria

ACO is described in [10], where the optimisation method is applied to the TSP. The method has also been used on related bin packing problems [17]. Lee et al. [30] propose using a vector of different resources to describe the state of the system, and a scalar function to describe the instantaneous system state parsimoniously. The system variables can be stored in a central database for easy retrieval of cloud state data by the resource allocation function. In the scenario under consideration - initial placement of VM requests - customers would only benefit from trying to gain as good service quality as possible, since the price is fixed for a specified time. The cloud operator, on the other hand, may save cost by assigning resources optimally. The optimisation goal is to find in some respect best trade-off between cost and quality. A multi-optimal approach seems infeasible due to the hardness of the problem. Solutions optimal with respect to different criteria will tend to be vastly different, and there is no way to find a trade-off by interpolation due to the discrete nature of resource assignment. A second question is how to define service quality from a system perspective. With a central function for resource allocation, requests are assigned one by one, and simple heuristics would give no guarantee of fairness in service quality. In this paper, the minimum level of quality is therefore determined by the SLA of each request.

The actual ACO simulation is performed in C++. This implementation is off-line and intended to study the optimality properties of the algorithm. In CloudSim, however, the source code is Java, and the logic has to be on-line. The algorithm is therefore reformulated using a dynamic programming approach where the virtual machines are mapped onto servers with respect to available server resources. The CloudSim is run under Eclipse, with addition of the three policy rules (greedy, round-robin, and optimal). The number of ants equal is set to the number of nodes (5), and the number of cycles to 1000. The auxiliary parameters are set to alpha = 0.5, beta = 0.5, evaporation = 0.5 and tau = 0.1, for all i and j, rather conservatively. The attractiveness (visibility, eta) is the ‘distance matrix’ constructed from scaled processing capacity.

In the ACO, ants’ movements are governed by target probabilities that is a product of two parts. The first is an assignment probability that is proportional to the *attractiveness* of a match from the customer point of view (called *visibility* in [10]), and the second a memory of the best past assignments represented by the fictitious pheromone trail. As long as the SLA of a request can be fulfilled, the attractiveness is non-zero, otherwise it is zero. The probability of transition to another (including self) node is

$$p_{ij} = \frac{(\tau_{ij}(t))^{\alpha}(\eta_{ij}(t))^{\beta}}{\sum(\tau_{ij}(t))^{\alpha}((\eta_{ij}(t))^{\beta'}} \tag{6.2}$$

evaluated for feasible assignments, otherwise $p_{ij} = 0$. The pheromone $\tau_{ij}(t)$ and the attractiveness $\eta_{ij}(t)$ are time dependent, which is indicated by the argument in *t*. The first property changes with each cycle, and the second with each move within a cycle.

Several papers also consider optimisation with respect to geographically distributed clusters [22][34], and include network bandwidth as a system parameter. Another view is to divide requests into tasks, and schedule the tasks optimally [45]. Both consolidation and scheduling requires the possibility of migration of resizing of tasks. The optimisation algorithm described in this paper is adapted to the initial placement problem for the reservation type of cost and allocation in a single cluster. Consolidation and scheduling are not considered. The algorithm as such, however, is sufficiently general to be extended to more general resource allocation problems. From a global optimisation perspective, the system lets the customers find an assignment according to their preferences and the given constraints, and then select the best assignment out of a number *N* of trials.

6.3.1 Attractiveness

From the customers' point of view, it is reasonable to maximise their own benefit at each step. Assuming that the service comes at a fixed price per VM configuration, the customer would try to maximise the service quality accordingly. This is likely represented by response time, that is the sum of CPU processing time and network transmission delay. The transmission delay is not considered here, as it depends mainly on the infrastructure outside the cloud. The attractiveness refers to the property of the algorithm on which the probability of choosing a server is based. The free capacity measure is used to describe this property based on the principle that the more free capacity in a server, the more "*attractive*" it is for the customer to be allocated to.

The CPU capacity is typically measured in MIPS (Millions of Instructions Per Second). The effective processing power available to applications is dependent on system configuration and simultaneously running processes etc. In [30], the authors propose measuring the available CPU and RAM capacity by performing a matrix inversion operation and measure the execution time. This method would give an accurate instantaneous measurement on which the attractiveness can be based. For this discussion, however, it is sufficient to assume that this information is available. Thus, assume that a customer selects server based on the available processing power of the CPU (the total processing power adjusted for system processes and other VMs using it). The customer then sees the available processing power as a resource potentially available to itself.

6.3.2 Cost

Cost can be defined in terms of idle capacity, that is, unoccupied capacity that cannot be assigned to another VM due to limitations in some other resource type. The cost will depend on the applications, or in other words, the distribution of demands of arriving requests. The cost for a cloud operator can be expressed in the degree of infrastructure utilisation, or equivalently, return on investments. The operator wishes to allocate requests to resources in a ”*best fit*” manner, so that not more resources than necessary are occupied by an allocated request.

The greedy principle from the cloud operator's perspective is that the more VMs that can be allocated, the higher the utilisation and the return on investment. As a metric for system efficiency, the *energy* of the relative free resources is used here. The energy unit for processing capacity becomes (instructions)^2/s^2. The optimisation then follows the principle of minimum energy. The system energy is defined as

$$\mathrm{E} = \sum_{i=1}^{n}(C_i - \sum_{j=1}^{v_i} r_{ij})^2, \tag{6.3}$$

where C_i is the capacity of the server and r_{ij} is the VM capacity requirement of VM j on host i. The total requirement sums over the v_i VMs allocated to host i.

Rescaling Equation (6.3) gives

$$\mathrm{E} = \sum_{i=1}^{n} \left(1 - \sum_{j=1}^{v_i} r_{ij}/C_i\right)^2, \tag{6.4}$$

which is the objective function that will be minimised. Under this metric, it is more efficient to allocate available resources occupying the best fit between request and resource. The best fit is when the resource matches the VM specification exactly. Then, the energy of the match is zero.

6.4 Algorithm for Resource Optimisation

ACO is suitable for many optimisation problems that can be modelled by a graph, including resource assignment [10][17]. It is, however, in its original form modifying the edges, but not the nodes, in a graph. The algorithm is an adaptation of the ACO algorithm for solving the TSP, described in [10]. The assignment problem is modelled as a complete graph on the set n of nodes. Initially, the ants are distributed between the nodes in a round-robin fashion and could also originate from a source node (a ”nest”), but this is not necessary, as the algorithm only performs a single iteration in each cycle. The ants could also be distributed randomly, which would affect the order of assignment. In the example below, however, this has no or little effect. The ants move according to a matrix of transition probabilities, where self-loops are allowed, so that an ant may request its job to be assigned to the node it originally occupies. The probabilities are proportional to the attractiveness of the target and the pheromone level of the edge between the origin and the target. The algorithm therefore has to keep track of the resource requirements of each ant, and the instantaneous amount of free resources at each node.

As opposed to the TSP, where the attractiveness is fixed, the system state changes with assignment of a new job (the property of the ant, or customer). Therefore, after each move, the transition probabilities change and must be recalculated. The attractiveness of a server to a given customer decreases when resources are assigned another customer. As a measure of attractiveness, the (possibly scaled) available CPU processing power of the host is used. Whereas in the TSP, the goal is to find a path, the objective here is to find an assignment. Since the constraints ensure that all allowed assignments are feasible, the algorithm only runs for one iteration, where each ant is moved (including possibly back to its origin) exactly once. The number of cycles will have to be fairly large though, in order to find an optimum. After an iteration, a candidate assignment has been generated. The algorithm has to keep track of the so-far best assignment through the cycles.

The constraints also differ from the TSP case, where an ant is forbidden to visit a node already visited by a *tabu list.* In the present case, the tabu list simply consists of the nodes that cannot with remaining resources accommodate the ant, that is, the VM. This includes the host it is initially starting from. Next, the system cost c_N is calculated according to Equation (6.3). This energy can be a composite measure including other resources, such as RAM as suggested in [30]. For the purpose of describing the algorithm, however, the energy is only based on the CPU processing power.

The deposited amount of pheromone, $\Delta\tau$ on each edge is now dependent on system cost, rather than on a single ant's trail as in the TSP. This quantity is given by

$$\Delta\tau = Q/c_k, \tag{6.5}$$

where Q is a scaling constant and c_k the cost in cycle $k \in \{1, 2, \dots, N\}$. Since c_k can be zero, a maximum limit on $\Delta\tau$ is set to one. This limit is rather arbitrary, and is an additional system parameter that may affect the convergence properties of the algorithm.

Since the attractiveness changes dynamically throughout the algorithm, the transition probabilities are given by two matrices: the attractively matrix A, which changes throughout an iteration but is reset for each cycle, and the pheromone level matrix P, that remains constant throughout a cycle, but is updated after each cycle. The cost is used to update matrix P, first by multiplying all previous pheromone levels p_{ij} by the evaporation constant $(1 - \rho)$, and then by adding $\Delta\tau$ onto edges describing assignments made in the iteration. The minimum cost c_{min} and the corresponding assignment obtained so far is recorded after each cycle, the matrix A and the vectors of free node resources and assignments are restored to their initial values, corresponding to no yet assigned VMs.

Algorithm 6.0.1 (Resource Allocation)**.**
Given matrices of server capabilities *S* and VM requirements *V*.

```
STEP 0: (initialise)
Let $J$ be a list of initial node assignments, and set algorithm parameters $\alpha$, $\beta$, $\tau_0$, $\rho$ and $K$,
the number of cycles. Set the matrix of free resources to $A = S$ and the matrix of
pheromone concentrations $P = (\tau_0)$, the matrix where all entries equals $\tau_0$. Set $c_{min} = \infty$

STEP $k = \{1, 2, . . . ,N\}$: (iterate)
    while $k <$ K (the number of cycles) do:
      Randomly select a node $i$ and a customer request,
      Divide a customer request into tasks (ants). For each ant $j$:
        Calculate the probabilities $p_{ij}$ (Eq. (6.2) based on $A$ and $P$,
        By simulation, select a move of ant $j$, assign to the selected target node if
        resources are available; assign resources, update $A$ (end).
      When all ants have been moved once:
        Calculate the cost $c_k$ (defined by the energy in Eq. (6.3) of the assignment,
        and update matrix $P$ by $\Delta\tau$.
      If $c_{min} < c_k$, let $c_{min} = c_k$ and $J_{min} = J_k$. Reset $J$, $A = S$,
      and let $P = (1 - \rho)$ $P$. (end)
     Delete customer request (end)
    end;

 Output $c_{min}$ and $J_{min}$ , the optimal assignment.                    •
```

It should be noted, that the algorithm assigns VMs all at once, so for the CloudSim experiment described below, the assignment rule has to be formulated so that VMs can be assigned sequentially. Also, the result of the algorithm depends on the random number generator, and so to find an optimum the algorithm may have to be run several times, and possibly with different seeds. The memory induced by the pheromone trails may therefore take many runs to change from a suboptimal assignment to an optimal.

6.5 Experiments

To test the algorithm, a small cluster with hosts similar to the one described in [30] (Section 6.5, Experiments, Tables 6.3 and 6.4) is used. The simplicity of this scenario with five servers having different characteristics and a single type of VM, makes manual comparison with other assignment schemes straightforward. To evaluate different assignment strategies, it is a good idea to have hosts with different characteristics but identical VMs that clearly shows how different strategies being assigned. The algorithm as such could easily be extended to larger and more general cases. Like in [10], the number of ants (VMs) is set equal to the number of nodes. The properties of the host servers in the cluster are listed in Table 6.1, and of the virtual machines in Table 6.2.

To compare the algorithm with other assignment schemes, this thesis compares the result with the round-robin, and a customer greedy heuristic schemes. In the round-robin scheme, the VMs are simply distributed one at each node, and the relative free capacity

Table 6.1: Cluster specification: MIPS and RAM capacities.

Host ID	Core	MIPS	RAM
0	1	1000	2048
1	2	500	2048
2	2	300	2048
3	1	2000	2048
4	2	300	2048

Table 6.2: Virtual machine specification: requirements on MIPS and RAM.

VM ID	Core	MIPS	RAM
0-4	1	300	512

is shown in Table 6.3. In Tables 6.3, 6.4 and 6.5 the entries for each host are the percentage of free capacity, calculated as 1- (occupied capacity)/(total host capacity). The cost as defined in Equation (6.3), that is, the sum of the squared entries, is 1.3725. Taking the number of processors into account, the energy is 2.2025.

Table 6.3: Efficiency of round-robin assignment.

Host ID	**No. VM**	**Free Capacity w/o PEs**	**Free Capacity With PEs**
0	1	0.7	0.7
1	1	0.4	0.7
2	1	0.0	0.5
3	1	0.85	0.85
4	1	0.0	0.5
Energy		1.37	2.20

The round-robin and the greedy algorithms are deterministic, whereas the ACO algorithm is random. The algorithm may therefore give a different result at each run. This depends on the random number generator. Since the round-robin assignment scheme is deterministic and not an optimisation method, it is likely to perform poorly when there are VM with different requirements. In this example, however, the assignment is good, under the energy metric. By letting each customer choose server according to the largest amount of available processing capacity, the assignment is as shown in Table 6.4. The cost in this case is 3.65. The same value is achieved when taking the number of processors into account, since there is one VM per host. The greedy scheme is essentially what would be expected from a single iteration of the algorithm.

The algorithm applied to the same problem gave the assignment shown in Table 6.5. It should be noted that lower capacity hosts (1, 2 and 4) are assigned VMs, but not node 3. The minimum energy obtained is 1.32. After having reached the minimum energy, the algorithm was run for up to $N = 10000$ without showing any further improvement. Taking the number of processors into account, the energy is 2.15 for this policy.

Table 6.4: Efficiency of greedy assignment.

Host ID	**No. VM**	**Free Capacity w/o PEs**	**Free Capacity with PEs**
0	1	0.7	0.7
1	0	1.0	1.0
2	0	1.0	1.0
3	4	0.4	0.4
4	1	1.0	1.0
Energy		3.65	3.65

Table 6.5: Efficiency of ACO assignment.

Host ID	**No. VM**	**Free Capacity w/o PEs**	**Free Capacity with PEs**
0	2	0.4	0.4
1	1	0.4	0.7
2	1	0.0	0.5
3	0	1.0	1.0
4	1	0.0	0.5
Energy		1.32	2.15

The parameter values used are $\alpha = 0.5$, $\beta = 0.5$, $\rho = 0.1$ and $\tau_0 = 0.1$. The cut-off limit for the inverse of the cost was set (rather arbitrarily) to unity. Elaborating on the system parameters would probably influence the convergence of the algorithm greatly, but has not been studied in detail in this project. The convergence shown in Figure 6.1 shows one possible run trace of the algorithm. It could, for example, also jump up and down before settling down.

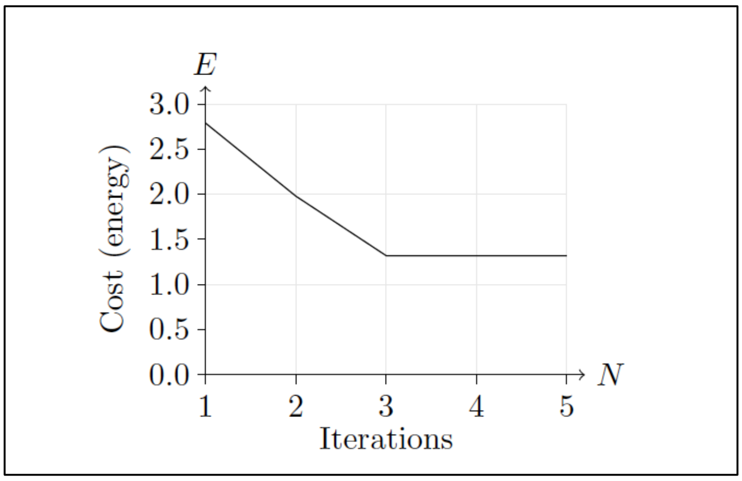


Figure 6.1: The convergence of the algorithm in the example.

The Figure shows that the first value is somewhat greedy (but still random) and come at a rather high cost, but once the minimum is reached, it stays there.

6.5.1 CloudSim Implementations

In the CloudSim experiment the goal was to keep things as simple as possible apart from the hosts and VMs. Only one user, one datacenter and one broker was therefore initiated. The VMs represent the ants, and the cloudlets jobs assigned to the VMs. The implementation uses 10 cloudlets, like in the code by [41]. These can of course be chosen differently, but the cloudlets are basically just a test to see that the cloud works. The CPU capacity is measured in MIPS (million instructions per second) in CloudSim - per processor (Pe). Thus, MIPS and RAM are properties used of the hosts in the datacenter (bandwidth and storage are not used). For the algorithm can use any unit, either GHz or MIPS, as exchangeable and give the same results. Host IDs are set manually, so these are defined according to Table 6.1.

The three assignment strategies where implemented in CloudSim. CloudSim (3.0.3) in run under the Eclipse Mars environment. Additional java files used are a cloud simulation file, specifying details of VMs and hosts, and a VM allocation policy class containing the assignment logic. The CloudSim environment is used to verify the online version of the algorithm, and produces the same result as the offline C++ implementation. The cloud was defined as described in Tables 6.1 and 6.2. The round-robin assignment was implemented by [41] and has been used for comparison. The simulation uses ten cloudlets of equal small sizes to illustrate that the clouds with the given assignment policies work properly. The round-robin assignment is shown in Table 6.3. For the greedy algorithm, the assignment policy is implemented so that each VM is assigned the host with the largest available MIPS (CPU capacity), possibly after some VM already has been assigned to the host. The assignment is in agreement with Table 6.4. The optimal assignment is implemented so that VMs can be assigned sequentially. This is necessary, because the algorithm described makes a repeated assignment of all VMs at once, whereas in CloudSim, VMs are assigned on a first-come first-served basis. Therefore, optimal result of the algorithm needs to reformulate as a policy, and in order to do so, we may use dynamic programming is used.

Consider the energy Equation (6.4). This is the cost to minimise by assignments of VMs to hosts and also have the rather obvious restrictions on the problem

$$\sum_{j=1}^{v_i} r_{ij} \leq C_i, \tag{6.6}$$

$$r_{ij} \geq 0, \tag{6.7}$$

for all *i* and *j*. The dynamic program can be written

$$V_k(y) = \min\left\{V_{k-1}(y), V_{k-1}(y) + (1 - \frac{r_k}{C_k})^2\right\} \tag{6.8}$$

where $y = \sum_{i=1}^{n}\left((1 - \sum_{j=1}^{v_i} r_{ij}/C_i)^2\right)$ is the relative free capacity at each instant, and r_k is a new VM to be allocated to a host with capacity C_k in step *k*. The dynamic programming formulation of the problem is the base for the implementation of the policy in CloudSim, since it is sequential as compared to the algorithm which is parallel. Thus, the energy Equation (6.4) is therefore implemented only implicitly in the simulation.

The algorithm described assigns VMs so that the largest decrease in energy occurs for each VM assignment to a host. For example, the decrease is much greater when the free capacity decreases from 1 to 0.5, than when the capacity decreases from 0.5 to 0. Thus, heuristically (and knowing the capacities of the hosts), if the proportion of available MIPS goes below the threshold (50%), the second best match is used.

The policy aims at both minimising unutilised capacity in a host. It also aims at distributing VMs so that the load is less than unity. Since the energy in Equation (6.4) is a sum of squares, this leads to a minimum energy assignment. The result of the simulation is in agreement with Table 6.5. The assignment is in practice made as follows. For the first VM, the host with the lowest capacity able to accommodate the VM is found, and an assignment is made. In this case, the free capacity decreases from 1 to 0.5, or the energy contribution (which is the squared value), from 1 to 0.25. The second VM could be assigned to the same host. Then the energy contribution would decrease from 0.25 to 0. But assigning it to another host with the same capacity would give an energy decrease from 1 to 0.25 as well, which is lowers the energy sum more than the decrease from 0.25 to 0 which is achieved when the VM is assigned to the first host.

Continuing in this way, a situation where the decrease in energy is larger when assigning a VM to a host which already has a VM assigned, rather than assigning it to another host with larger free capacity. So for host with ID 0, the decrease in energy is larger by assigning two VMs to this host, as compared to assigning one VM to host 0 and one VM to host 3. The cost in the implemented in CloudSim therefore looks for the host for which the VM allocation gives the largest decrease in energy. This is used in the CloudSim policy as an optimisation parameter to find the best match. Figure 6.2 shows the energy for each of the three assignment strategies for an increasing number of standard size VMs in Table 6.2. The algorithm described in this study (green line) has lower energy than the other two, although the round-robin strategy (blue line) is close to optimal.

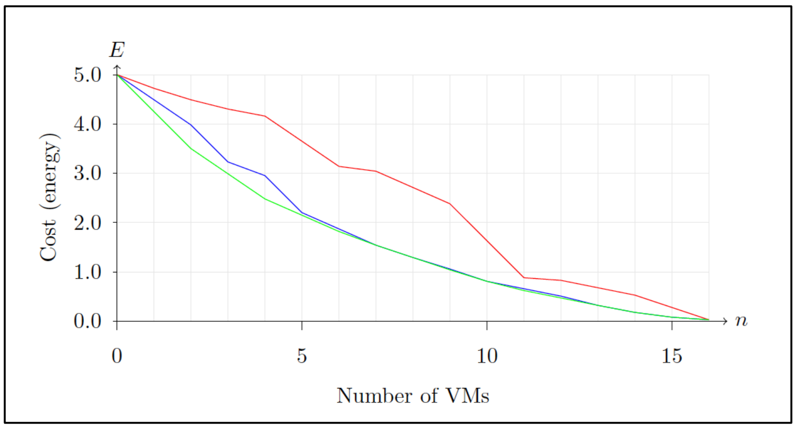


Figure 6.2: Comparison of the efficiency the algorithms; greedy (red), round-robin (blue) and AOC (green).

6.6 Chapter Summary

This chapter concludes that the proposed algorithm using ACO shows a better results compared to the most used algorithm in the market; Greedy and Round-Robin Algorithm. Since ACO is a meta-heuristic (randomised algorithm) it is difficult to give complexity estimates. However, in practice the algorithm can be reformulated and implemented as a dynamic program.

Chapter 7

CONCLUSIONS AND FUTURE WORK

In this final chapter, all the works in this thesis are concluded and in brief, explain regarding the future works that may improve the methods of the system in the future study.

7.1 Conclusions

This thesis investigates a logic for congestion control on network level based on SDN and NFV and showing that traffic aggregation on node level combined with optimal routing with respect to delay. Traffic aggregation is performed on node level using entropy of aggregate traffic streams as decision variable for aggregation and resource allocation. The optimal paths are selected to minimise either the total end-to-end delay or the maximum end-to-end delay. Note that path optimisation with respect to maximum end-to-end delay gives slightly better results compared to optimisation with respect to total delay and that the congestion control should be combined with admission control at access points and end-to-end flow control, such as TCP (or a modification of TCP).

By comparing three policies for assigning VMs to hosts; round-robin, a customer greedy heuristic, and an optimised allocation policy derived from an ACO algorithm, these have been simulated in CloudSim. To measure a cloud operator's cost, an energy function has been used, and the main objective has been cast-off to find as assignment policy that minimises this energy. When comparing the three assignment policies, the round-robin can be said to be both simple and efficient (has low energy) in this setup. The greedy assignment, where a customer can choose to allocate a VM where there are most available resources is rather expensive.

An optimal assignment can be found using the algorithm described in this thesis, which minimises the energy used to measure cloud operator costs. The benefit for the cloud operator is to minimise the possibility to add further VMs to the cloud without performance degradation or delays. All assignment policies have been simulated and tested in CloudSim, and the processing of cloudlets have in this setup been shown to be equally efficient.

7.2 Future Work

The congestion control algorithm proposed in this thesis use traffic measurements and predicted network states for aggregation and routing decisions. It gives measurable performance gains at reasonable traffic levels. It should therefore be combined with admission control to protect the network from severe overload and end-to-end flow control to achieve fairness and rate limitation on flow level. For future study, the relationship between different types of congestion control (hop, end-to-end, access, and transport level) should be investigated.

Furthermore, since different services are sensitive with respect to different QoS measures, a composite metric for route optimisation should be investigated. It should also be intrusive to analyse control actions on different time scales. This may affect stability as well as the efficiency of the congestion control. Although long range dependent traffic is likely to attain a load level close to previous one, predictive congestion control can likely be improved by adequate modelling and computing forecast intervals, which should improve the resource utilisation further.

References

[1] Z. Abbasi, F. Noroozi, and A. Sharifi, “Congestion Level Prediction of Self-Similar Traffic Based On Wavelet Transform,” 2005.

[2] A. Adas, “Traffic models in broadband networks,” *IEEE Commun. Mag.*, vol. 35, no. 7, pp. 82–89, 1997.

[3] M. B. Al-Somaidai, “Survey of Software Components to Emulate OpenFlow Protocol as an SDN Implementation,” *Am. J. Softw. Eng. Appl.*, vol. 3, no. 6, p. 74, 2014.

[4] J. Beran, *Statistics for Long-Memory Processes*. 1998.

[5] E. Buffet and N. G. Duffield, “Exponential Upper Bounds via Martingales for Multiplexers with Markovian Arrivals,” vol. 31, no. 4, pp. 1049–1060, 2014.

[6] S. Crosby, I. Leslie, and J. T. Lewis, “Bypassing Modelling : an Investigation of Entropy as a Traffic Descriptor in the Fairisle ATM network,” pp. 1–10, 1995.

[7] M. E. Crovella and A. Bestavros, “Self-similarity in World Wide Web traffic: evidence and possible causes,” *IEEE/ACM Trans. Netw.*, vol. 5, no. 6, pp. 835–846, 1997.

[8] M. E. Crovella and A. Bestavros, “Explaining World Wide Web Traffic Self-Similarity,” *October*, pp. 1–19, 1995.

[9] K. Dolzer, W. Payer, and M. Eberspächer, “A simulation study on traffic aggregation in multi-service networks,” *IEEE Int. Conf. High Perform. Switch. Routing, HPSR*, pp. 157–165, 2000.

[10] M. Dorigo, V. Maniezzo, and A. Colorni, “Ant system: Optimisation by a colony of cooperating agents,” *IEEE Trans. Syst. Man, Cybern. Part B Cybern.*, vol. 26, no. 1, pp. 29–41, 1996.

[11] D. D’souza, K. P. Sundharan, S. Lokanath, and V. Mittal, “Improving QoS in a Software-Defined Network,” pp. 1–9, 2016.

[12] N. G. Duffield, “Exponential bounds for queues with Markovian arrivals,” *Queueing Syst.*, vol. 17, no. 3–4, pp. 413–430, 1994.

[13] N. G. Duffield, N. G. Duffield, J. T. Lewis, N. O’Connell, R. Russell, and F. Toomey, “Entropy of ATM Traffic Streams: A Tool for Estimating QoS Parameters,” *IEEE J. Sel. Areas Commun.*, 2006.

[14] H. E. Egilmez and S. T. Dane, “OpenQoS: An OpenFlow controller design for multimedia delivery with end-to-end Quality of Service over Software-Defined Networks,” *Signal Inf. Process. Assoc. Annu. Summit Conf. (APSIPA ASC)*, pp. 1–8, 2012.

[15] N. Feamster, J. Rexford, and E. Zegura, “The Road to SDN: An Intellectual History of Programmable Networks,” *ACM Sigcomm Comput. Commun.*, vol. 44, no. 2, pp. 87–98, 2014.

[16] W. C. Feng, K. G. Shin, D. D. Kandlur, and D. Saha, “The Blue active queue management algorithms,” *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 513–528, 2002.

[17] S. Fidanova, “ACO algorithm for MKP using various heuristic information,” in *Numerical Methods and Applications*, 2003, pp. 438–444.

[18] A. E. García and K. D. Hackbarth, “Next generation IP access networks planning: Approximated methods,” *WSEAS Trans. Commun.*, vol. 5, no. 3, pp. 535–542, 2006.

[19] M. Gerla and L. Kleinrock, “Flow Control: A Comparative Survey,” *IEEE Trans. Commun.*, vol. 28, no. 4, 1980.

[20] R. J. Gibbens, P. J. Hunt, and F. P. Kelly, “Bistability in communication networks,” *Disord. Phys. Syst.*, no. 1973, pp. 113–128, 1990.

[21] R. J. Gibbens, S. Sargood, C. Van Eijl, F. Kelly, H. Azmoodeh, R. Macfadyen, and N. Macfadyen, “Fixed-point models for the end-to-end performance analysis of IP networks,” *13th ITC Spec. Semin. IP Traf c Manag. Model. Manag.*, pp. 1–8, 2000.

[22] H. Goudarzi and M. Pedram, “Maximizing profit in cloud computing system via resource allocation,” in *Proceedings - International Conference on Distributed Computing Systems*, 2011, pp. 1–6.

[23] Z. J. Haas and J. H. Winters, “Congestion Control By Adaptive Admission,” *Proc. IEEE Int. Conf. Comput. Commun.*, pp. 560–569, 1991.

[24] Y. Hu, D. M. Chiu, and J. C. S. Lui, “Entropy based adaptive flow aggregation,” *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 698–711, 2009.

[25] X. Huang, C. Lin, F. Ren, G. Yang, P. D. Ungsunan, and Y. Wang, “Improving the convergence and stability of congestion control algorithm,” in *Proceedings - International Conference on Network Protocols, ICNP*, 2007, pp. 206–215.

[26] V. Jacobson, “Congestion avoidance and control,” *ACM SIGCOMM Comput. Commun. Rev.*, vol. 18, no. 4, pp. 314–329, 1988.

[27] F. Keti and S. Askar, “Emulation of Software Defined Networks Using Mininet in Different Simulation Environments,” in *Proceedings - International Conference on Intelligent Systems, Modelling and Simulation, ISMS*, 2015, vol. 2015–Octob, pp. 205–210.

[28] J. Kristoff, “TCP Congestion Control,” 2002.

[29] C. Larsson, “*Design of Modern Communication Networks – Methods and Application, Academic Press,”* 2014.

[30] H. Lee, Min, L. Y. Jeong, and H. Jin, “Performance analysis based resource allocation for green cloud computing,” *J Supercomput DOI 10.1007/sl 1227-013-1020-x*, no. September, pp. 1013–1026, 2013.

[31] W. E. Leland, M. S. Taqqu, and D. V. Wilson, “On the Self-Similar Nature of Ethernet Traffic (Extended Version),” *IEEE/ACM Trans. Netw.*, vol. 2, no. 1, pp. 1–15, 1994.

[32] J. V. Lévy and R. Riedi, “Fractional Brownian motion and data traffic Modeling: The Other End of the Spectrum,” *Fractals Eng.*, no. August, 1997.

[33] J. T. Lewis and R. Russell, “An Introduction to Large Deviations for Teletraffic Engineers,” *Program*, pp. 1–45, 1997.

[34] K. Mills, J. Filliben, and C. Dabrowski, “Comparing VM-placement algorithms for on-demand clouds,” in *Proceedings - 2011 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2011*, 2011, pp. 91–98.

[35] X.-N. Nguyen, D. Saucez, C. Barakat, and T. Turletti, “Optimizing rules placement in OpenFlow networks,” in *Proceedings of the third workshop on Hot topics in software defined networking - HotSDN ’14*, 2014, pp. 127–132.

[36] J. Ni, T. Yang, and D. H. K. Tsang, “Source modelling, queueing analysis, and bandwidth allocation for VBR MPEG-2 video traffic in ATM networks,” *IEE Proc. - Commun.*, vol. 143, no. 4, pp. 197–205, 1996.

[37] S. A. M. Ostring, “Dual Dimensional ABR Control Scheme Using Predictive Filtering of Self-similar TrMiC,” 1999.

[38] K. Park and W. Willinger, *Self-Similar Network Traffic and Performance Evaluation*, vol. 53. 2000.

[39] D. B. Percival and A. T. Walden, *Wavelet Methods for Time Series Analysis*. 2000.

[40] A. Pulipaka, P. Seeling, and M. Reisslein, “Traffic models for H.264 video using hierarchical prediction structures,” in *Global Communications Conference (GLOBECOM), 2012 IEEE*, 2012, pp. 2107–2112.

[41] Raju, “CloudSim Example with Round Robin Data center broker & Round Robin Vm Allocation Policy with Circular Hosts List,” 2016. .

[42] B. Ryu, D. P. Connors, and S. Dao, “Modeling and simulation of broadband satellite networks part 1: Medium access control for QoS provisioning,” *IEEE Commun.*, vol. 37, no. 3, pp. 72–79, 1999.

[43] K. Saravanan and Suresh R.M., “Service Oriented Real-Time Buffer Management for QoS on Adaptive Routers,” *J. Thoretical Appl. Inf. Technol.*, vol. 55(2), 2013.

[44] S. Shenker, “A Theoretical Analysis of Feedback Flow Control,” in *The Conference on Communications Architecture and Protocols (SIGCOMM)*, 1990, pp. 156–165.

[45] J.-T. Tsai, J.-C. Fang, and J.-H. Chou, “Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm,” *Comput. Oper. Res.*, vol. 40, no. 12, pp. 3045–3055, 2013.

[46] T. Tuan and K. Park, “Congestion control for self-similar network traffic,” *Network*, vol. 0, pp. 447–480, 1998.

[47] R. Wallner and R. Cannistra, “An SDN Approach: Quality of Service using Big Switch’s Floodlight Open-source Controller,” *Proc. Asia-Pacific Adv. Netw.*, vol. 35, p. 14, 2013.

[48] K. Wu and D. S. Reeves, “Capacity Planning of Survivable MPLS Networks Supporting DiffServ,” pp. 1–20.

[49] “Software-Defined Networks and OpenFlow,” *Internet Protoc. J.*, vol. Volume 16, no. No. 1 https://www.cisco.com/c/dam/en_us/about/ac123/ac147/images/ipj/ipj_16-1/161_sdn_fig03_lg.jpg.

[50] L. Liu, T. Tsuritani, I. Morita, H. Guo, and J. Wu, “Experimental validation and performance evaluation of OpenFlow-based wavelength path control in transparent optical networks,” *Opt. Express*, vol. 19, no. 27, p. 26578, 2011.

[51] J. Dix, “The promise of software defined networking,” 2013. [Online]. Available: https://www.networkworld.com/article/2167667/lan-wan/lan-wan-the-promise-of-software-defined-networking.html.

[52] “Seamless Legacy to SDN transformation.” [Online]. Available: https://pfsw.com/solutions/sdn-nfv/.

[53] “SDN vs NFV vs Traditional Networking,” 2015. [Online]. Available: http://sateeshkolagani.com/?p=221.

[54] Iversen, V.B., “Handbook: Teletraffic Engineering,” 2013. ITU-D. Technical University of Denmark. pp. 21-22, 884-93

[55] “5G PPP Architecture Working Group, View on 5G Architecture,” 2016.

[56] Stewart, W.J., “Introduction to the Numerical Solution of Markov Chains,” 2014. Princeton University Press, USA.

[57] W.E. Leland, M.S. Taqqu, W. Willinger, D.V. Wilson, “On the self-similar nature of Ethernet traffic (extended version),” Transactions on Networking 2 (1), February 1994.

[58] D.B. Percival, A.T. Walden, “Wavelet Methods for Time Series Analysis,” Cambridge University Press, USA, 2000.

[59] A. Lall, V. Sekar, M. Ogihara, J. Xu, H. Zhang, “Data streaming algorithms for estimating entropy of network traffic,” Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS’06), 2006, pp. 145–156.

[60] A. Lakhina, M. Crovella, and C. Diot., “Mining anomalies using tra_c feature distributions,” Proceedings of ACM SIGCOMM, 2005. (mining-anomalities-TR.pdf)

[61] Gross, D., Harris, C.M., “Fundamentals of Queueing Theory,” third ed. John Wiley & Sons Inc., USA., p. 275 and p.10, 1998.

[62] A. Wagner, B. Plattner., “Entropy Based Worm and Anomaly Detection in Fast IP Networks” In Proceedings of IEEE International Workshop on Enabling Technologies, Infrastructures for Collaborative Enterprises, 2005. (wetice05_entropy.pdf)

[63] Kelly, F., and Yudovina, E., “Lecture Notes on Stochastic Networks,” Cambridge University Press, 2014.

Appendix 1

Mathematical Symbols – Explanations

$$Z = \frac{Var(X)}{\boldsymbol{E}(X)}$$

$\boldsymbol{E}(X)$ denotes the expectations and $Var(X)$ denotes the variance of a random variable or process X.

$$\lambda \in \left[\bar{X} \pm z_{\alpha/2} s . n^{-1/2}\right]$$

(change notation) λ lies in the interval $\left[\bar{X} - z_{\alpha/2} s . n^{-1/2}\right], \left[\bar{X} + z_{\alpha/2} s . n^{-1/2}\right]$, and $z_{\alpha/2}$ is the upper $(1-\alpha/2)$ quantile of the standard normal distribution and $s^2 = (n-1)^{-1} \sum_{i=1}^{n} (X_i - \bar{X})^2$ is the sample variance estimating σ^2.

$$p(k) \approx H(2H-1)k^{2H-2},$$

$\approx$ should be interpreted as "is approximately", such as the ordinary least squares line approximating a random but correlated set of points.

$$Q_{t+1} = \max\{0, Q_t + A_t - S_t\},$$

The max{ } operator takes on 0 whenever $Q_t + A_t - S_t < 0$, and $Q_t + A_t - S_t$ otherwise (similarly for min{ }). The operator may take an index, denoting the variable that should be changed to attain the functional maximum (or minimum), such as

$$\boldsymbol{P}(M_n > a) \leq \min_{\theta > 0} e^{-n(\theta a - \wedge(\theta))} = \exp\left(-n \max_{\theta > 0}(\theta a - \wedge(\theta))\right).$$

$$\boldsymbol{P}(Q > b) = \boldsymbol{P}\left(\sup_{t \geq 0} W_t > b\right).$$

The $\boldsymbol{P}$ operator denotes the probability (with respect to some probability measure). $\sup_{t \geq 0} W_t > b$ is the supremum (the largest value) of W_t for any $t > 0$ such that $W_t > b$. This is a slight abuse of notation: Should the value of W_t never exceed b, the conditional supremum would be undefined, but its probability zero.

$$\boldsymbol{P}\left(\sup_{t\geq 0} W_t > b\right) = \boldsymbol{P}(\cup_{t\geq 0}\{W_t > b\}) \geq \sup_{t\geq 0}\mathbf{P}(W_t > 0),$$

In this equation, the notation $\cup_{t\geq 0}\{W_t > b\}$ are all the events where $W_t > b$. You may think of it as the sum of the time segments (t_1, t_2) , $(t_3, t_4), \ldots, (t_n. t_{n+1})$ where for $t \in (t_i, t_{i+1})$ it is true that $W_t > b$, and the sum divided by the total time of consideration T giving the probability.

$$\boldsymbol{P}(M_n > x) \asymp e^{-nI(x)} \quad for\ x > \mu$$

$$\boldsymbol{P}(M_n < x) \asymp e^{-nI(x)} \quad for\ x < \mu$$

The notation $\asymp$ means “is asymptotically equal to”, so that the right-hand side approaches the left-hand side as (in this case) $n \rightarrow \infty$ (as n approaches infinity). It does not indicate how fast the left-hand side approaches the right-hand side, but rather indicates the mathematical form “for large n”.

$$M(\theta) = \boldsymbol{E}\left(e^{\theta X}\right) = \int e^{\theta X} f(x)dx,$$

The symbol d signifies the differential operator (derivative) and is not in italics to distinguish it from the variable d, which can be anything.

$$I_A(x) \triangleq \begin{cases} 1 & if\ x \in A \\ 0 & otherwise \end{cases}$$

The symbol $\triangleq$ denotes identity, and can be read as “is defined by”.

$$\boldsymbol{E}\left(I_{[na,\infty)}(nM_n)\right) = \boldsymbol{P}(nM_n > na)$$

The symbol $I_{[na,\infty)}$ is the indicator function, taking value nM_n in the interval $[na, \infty)$ and 0 otherwise.