Topology Control for Resource Management in Microwave Backhaul Networks

by

Alexander Deng

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Engineering
in Network Engineering.

Victoria University of Wellington
2017

Abstract

Microwave backhaul networks are the dominant technology used to connect together access and core networks for their flexibility and cost-effectiveness in deployment. Unfortunately, microwave backhaul networks are susceptible to interference and are statically managed leading to poor Quality of Service (QoS) in the form of high delays and loss as well as being inefficient on energy. The use of Software Defined Networking (SDN) is proposed to address these problems by dynamically managing resources to work around the interference and remove static allocations. Two new algorithms, CUT and OptiCUT were designed to compute an optimal topology, to minimise loss and delay while at the same time reducing power consumption.

Acknowledgments

Most importantly, I would like to express my thanks to my supervisor Dr. Bryan Ng. He motivated me throughout the year and and went beyond what he was required by keeping me on track to the end goal. I would also like to acknowledge Prof. Winston Seah for his guidance throughout the past year.

I would also like to thank Aviat Networks and Darran Hunt (formally of Aviat Networks) for providing ideas and support for the project undertaken in this thesis.

A big thanks to my sister for proofreading my thesis for me. I know it wasn't easy reading this. Finally, I would like to thank Maria Libunao for supporting me throughout my Masters.

Contents

Chapter 1

Introduction

A backhaul network is used to connect the access and core networks together. Microwave radio is currently the favoured technology used, making up over 50% of all backhaul networks, beating out fibre and copper fixed lines [14, 64, 1]. They have several advantages which have been exploited by backhaul providers. Most importantly, microwave backhaul networks are flexible and cost-effective in deployment [39, 64, 1]. Next, microwave backhaul networks are used almost exclusively in developing countries as the infrastructure for telecommunications providers with limited wired infrastructure, or where it is physically infeasible to have a wired backhaul [47, 39]. Also, they utilise licensed bands which results in regulated interference on the channel as only licensed users can operate on these bands.

Microwave backhaul networks have several disadvantages that lead to Quality of Service (QoS) issues such as packet loss, delay as well as inefficient usage of energy [39]. Firstly, microwave backhaul networks are highly susceptible to environmental factors and resource constraints. This causes the link to be unstable in that the capacity it offers fluctuates leading to packet loss and increased delay [38, 36]. Secondly, the resources in a microwave backhaul network are statically managed leading to capacity wastage, which causes a balancing problem when incorrect allocations

are made to different applications [7, 5]. This in turn also leads to packet loss and delay. Finally, the static management of resources leads to inefficient power consumption which is a valid concern when the equipment of these microwave backhaul networks are deployed in locations that have no mains power. These disadvantages often lead to QoS issues that plague backhaul operators [7].

1.1 Motivation

With the predicted yearly increase in mobile traffic, microwave backhaul providers now have the need to efficiently utilise the resources in their networks [11, 63]. The use of smartphones has become an essential part of everyday life for many people. An increasing amount of traffic from these users is from video streaming, accounting for over 50% of all mobile data traffic [12]. This increase in video streaming has directly resulted in a 43% increase in average data usage per month in 2015 [12]. This is predicted to further tax the backhaul networks due to the increased levels of traffic [39, 65, 27].

The increase in mobile traffic is predicted to be a significant problem for microwave backhaul networks as they do not have unlimited capacity to easily cope with this increase. The capacity available for a microwave backhaul is directly linked to available spectrum. Spectrum is a limited and costly resource and future networks cannot increase capacity by utilising more spectrum [24]. Future networks require resources to be utilised as efficiently as possible; reallocating unused capacity will be key to keeping up with the new predicted levels of traffic [39].

To keep up with the increase in traffic and to utilise spectrum and capacity more efficiently, reuse of spectrum has been proposed. Microwave radio has been proposed to decrease the size of cellular sites into small cells (in current 4G LTE and future 5G networks) [18, 24], for instance, densely built areas such as on top of buildings in a city [10] where it is con-

sidered impractical or economically infeasible to be connected to a wired backhaul.

The centralised nature of SDN allows for a more effective way of managing capacity in a microwave backhaul network. The centralised architecture that SDN enables allows for information to be obtained and aggregated on a single entity [29, 50]. The concepts can be applied onto a microwave backhaul network to better manage the resource and increase utilisation to address the disadvantages that are associated with microwave technologies.

The research project has been conducted in collaboration with the Wellington office of Aviat Networks with some input from the Santa Clara office. Aviat Networks is a company that designs and markets a line of microwave backhaul routers, transmitters and radio cards. The algorithm developed for this Masters project is to be run and evaluated on hardware that is sold worldwide to backhaul providers using microwave radio technology.

1.2 Goal and Objectives

The research goal of this project is to improve QoS (packet delivery ratio and delay) and energy efficiency by dynamically controlling capacity in a microwave backhaul network in response to live demand and channel conditions. Objectives 1 and 2 are the main objectives of this thesis. They are to be met before objective 3 is considered. The output of the thesis is an algorithm to meet all 3 objectives.

In order to meet this goal, these objectives are to be met:

- **Objective 1: Improve packet delivery ratio**

  This objective focuses on improving the packet delivery ratio (or lowering packet loss) in a microwave backhaul network. An algorithm is to be developed to route traffic through a microwave back-

haul that is aware of network resource constraints and allocations. The algorithm will dynamically alter allocations in response to live demand.

- **Objective 2: Reduce delay within backhaul network**

  This objective aims to reduce the delay incurred when transporting traffic from access to core networks via a microwave backhaul network. Targets have already been defined for LTE networks such as a target of 10ms for video traffic [14].

- **Objective 3: Reduce energy usage** [1]

  This objective focuses on reducing power usage in a microwave backhaul network. Links are statically left on in current networks. The algorithm is to reduce the number of links in the network if possible.

1.3 Outline of Thesis

The remainder of the thesis is structured as follows:

Chapter 2: Background

This chapter provides the basis on which the work is being done such as information on graphs and pre-existing work in the literature.

Chapter 3: Problem Formulation and Notation Definitions

This chapter formulates the problem the thesis is solving. Both an informal description and a mathematical description of the problem being solved is given.

[1] The microwave routers are often not connected directly to mains power. They are often run on fossil fuels that need to be carried up to remote locations for generators.

Chapter 4: Network Model and Execution Environment

This chapter introduces the network model used to represent the microwave backhaul network and the environment which the algorithms and simulations are run on. Details on how the coordinates of the nodes are used as part of a wireless model to determine connectivity is given.

Chapter 5: Topology Control Algorithm Design

This chapter details the algorithms produced to improve packet loss, delay and power consumption.

Chapter 6: Evaluation

This chapter evaluates the algorithms produced using several key metrics.

Chapter 7: Conclusion

This chapter concludes the the thesis and gives insights into potential future work that can be done to extend this thesis.

Chapter 2

Background and Related Work

The focus of this thesis is to improve QoS and energy efficiency in a microwave backhaul network. Many networking problems are represented in terms of a graph theory problem. Graph theory problems are used to solve capacity, delay and energy problems using various techniques. Topology design, planning, creation and control and, routing and path finding are topics of interest to this thesis as they use graph theory techniques to represent and solve their problems. The use of these techniques in SDN networks are currently in infancy stages. The remainder of this chapter discusses the existing work in the literature of these topics.

2.1 Use of Graphs in Networking

Networking problems are commonly represented using graphs. The fundamental units that make up a graph are vertices (or nodes) and edges are used to represent routers/switches and the links that connect them respectively [55]. Matrices are rectangle arrays of numbers that are used to model the vertices and edges that make up a graph [55]. For graph theory, the matrices used to represent the graphs are square, meaning there is an equal number of rows and columns. Specifically, adjacency matrices are used in graph theory to represent the graph. An $n * n$ matrix (where n

is the number of vertices in the graph) is created. The element in the i-th row and j-th column represents whether the vertices are adjacent. 1 represents a direct connection between a pair of vertices (they are adjacent or neighbours) and 0 means there is not.

Graphs have been used for many years since the Internet's growth in networking problems such as path finding (Dijkstra's, Prim's, Kruskal's, Open Shortest Path First(OSPF)) and network simplification (spanning tree) [66, 49, 17]. During the rapid growth in the 1990s it became increasingly difficult to track and visualise the Internet and the World Wide Web (WWW) [17]. Graph theory was used to keep track of this growth, allowing for it to be visualised in diagrams. Today, it is used in path finding and modelling complex networking problems.

Such methods have been adopted into algorithms that allow business to take place using Border Gateway Protocol (BGP). The basis of this thesis takes these graph theory concepts, and applies them to microwave backhaul networks to solve the QoS problems and improve energy efficiency.

2.2 Topological Design: Initial Design and Deployment

Topology design is a field of networking that deals with optimising the initial design and deployment of a network using graph theory to model the problem [25, 34, 32, 53]. The target objectives of the deployments are often based on monetary cost, throughput and network efficiency. A summary can be seen in Table 2.1.

Gerla and Kleinrock [25] described the topology design problem as a way to generate topologies that minimise line cost in Advanced Research Projects Agency Network (ARPANET) (predecessor to the Internet). The authors alternate design objectives such as minimising network delay and maximising throughput. Additional objectives such as meeting capacity is

given as a subject to constraint in the formulation of a graph problem. The authors used capacity assignment, flow assignment independently and together to compute the topology.

Srivatsa *et al*. [60] proposed a basic heuristic to generate perturbations of a topology using node numbering. Their solution is limited in that they did not consider any cost involved with the topology that is output from their algorithm. Gavish [23] used combinatorial optimisation of a graph to find a topology that balances investment cost and queuing delay (queuing delay decreases as investment increases). The algorithm the author used to generate "feasible" solutions took 40 minutes to execute and calculate the topology which was high. Pierre *et al*. [53] solved a similar problem, using simulated annealing to find a least cost network that ensured delay and reliability constraints were met. The results showed that in a 10 node network, it took three minutes to find a local optimum topology (not the global optimum). The authors [53] found that better topologies with a lower cost could be found with more trials of the annealing but did not detail the effect it had on execution time. Simulated annealing is a heuristic based search approach and as such the authors noted the results varied with trials. These solutions generate topologies that meet the objectives defined by the authors' but the execution times of their algorithms are high despite generating non-optimal topologies.

To improve the execution times of algorithms used in topology design, Kumar and Banerjee [30] proposed using a multi-objective evolutionary algorithm to find a topology that meets delay and monetary cost objectives. The authors found that to further reduce delay, the monetary cost would increase as more physical links would have to be deployed in the topology. As with most Artificial Intelligence (AI) based algorithms, the topologies found in Kumar and Banerjee's [30] algorithm heavily depend on the initial training data and the accuracy of the heuristic used.

Topology types have a large effect on the cost of the topology calculated for deployment. Li *et al*. [35, 34] proposed a heuristic model to plan

for a microwave backhaul network to be deployed in a tree topology, given trade-offs between (i): the number of hops, (ii): the sum of link distances, (iii): the number of long links, (iv): the number of small angles and (v): the number of link crosses in the topology. Similarly, Kuo *et al*. [31], and Nadiv and Naveh [44] studied cost-effective backhaul topologies for providers. The authors [31] compared tree, mesh and ring based topologies using monetary cost and resilience to links going down and traffic demand increasing. The existing research shows that tree-based backhaul topologies yields the lowest cost for deployment but has poor resilience to links going down as the topology becomes disconnected when a single link goes down.

Schlinker *et al*. [57] created Condor, a pipeline that generates and evaluates network topologies. The network architect defines topology designs in a in a declarative "Topology Description Language" and the topology synthesiser generates topologies based on the design objectives (i.e. bandwidth, resiliency to failure, cost, complexity, scalability) set by the architect. The topologies calculated is visualised using Pythons Networkx library, allowing the network architect to modify the design objectives or choose the required tradeoff. The downside of Condor is that it requires human interaction, meaning it is not suitable for use as a topology control algorithm in dynamic wireless networks.

Topology planning in microwave backhaul networks focuses on deployment of nodes and links based on required capacity and monetary cost. It does not adjust the existing topologies to reduce power consumption. Such techniques can be used in new algorithms to simplify the topology (nodes and links in use) of existing topology deployments. The next section discusses existing work in controlling existing topologies to reduce power consumption and ensure capacity requirements.

2.3 Topology Control: Dynamically Changing a Topology

Topology control is similar to topology design. Instead of creating the initial topology, a preexisting topology is simplified in the form of a graph theory problem [58]. The purpose of this is to reduce the computational complexity of routing algorithms. It is also commonly used to reduce power consumption by removing links or powering off routers in a network. The existing algorithms from topology design and planning can be used although their execution times are high. A summary can be seen in Table 2.1.

Topology control is a well studied field in wired networks and wireless sensor networks (WSN). Unlike microwave backhaul networks where the nodes are generally left on 24/7, WSNs puts nodes to sleep depending on power constraints or how the topology is being controlled [68]. Both Ben-Othman *et al*. [4], and Ramanathan and Rosales-Hain [54] proposed the use of power adjustment in order to perform topology control. Yonezu *et al*. [69] used MiDORi to aggregate the traffic in a topology into a subset of the links in the network to reduce power consumption. MiDORi monitors the traffic load in the network and calculates the optimal topology in regards to power consumption and changes the topology to what was calculated. The authors found that they could reduce power consumption by 60% during low loads and 40% during high loads. As the authors' algorithm is iterative and exhaustive, the calculation time becomes high once the number of nodes in the network increases beyond 40.

In wired backbone networks, Chiaraviglio *et al*. [8] reduce power consumption while maintaining link utilisation at 50%. The authors use QoS constraints in the form of flow conservation and maximum link utilisation with integer linear programming to turn off routers and links to alter the network topology. However, the problem they solve falls under the capacitated multi-commodity minimum cost flow problems (CMCF) which

is NP-hard. This is why they used Greedy heuristics to find the set of links and routers that need to be powered on to meet the constraints. Cuomo *et al*. [16] utilised graph pruning to reduce power consumption. They found that power consumption reduced by 10.4% to 13.9% depending on link utilisation. As utilisation or connectivity degree requirement increased, the authors found that links could not be pruned off. Packet loss was observed in the test cases though.

Wireless backhaul networks have the same objectives as wireless networks in which topology control algorithms are used to reduce energy consumption and maximise capacity [35, 34, 46]. An integer linear programming problem is used by Nepomuceno [46] to adjust radio configurations to minimise power consumption in grid-based microwave backhaul topologies. The authors do this by adjusting modulation and coding along the links. Genetic algorithms have been used in wireless networking to perform topology control [43, 6]. Genetic algorithms appear to have low execution times in computing near optimal topologies for power saving or capacity aware routing but there is no mention in [43] of training times and the accuracy based on training data.

Topology control has been used in the literature for fault tolerance and reliability of a network when links have gone down. Szlachcic [61] and, Szlachcic and Mlynek [62] designed genetic algorithm based solutions to ensure average delay in the topologies calculated. Szlachcic [61] found their solution performed better than simulated annealing but depended heavily on population and training. The solution proposed by Szlachcic and Mlynek [62] uses number of links to evaluate the effectiveness of their algorithm. Topologies are chosen based on cost once delay has met the requirement threshold. Kamiyama [28] utilised binary partition to enumerate over potential topologies to solve monetary cost and quality objectives. Using complete enumeration as the baseline, Kamiyama [28] found that the execution time to find all topologies that meet the objectives would increase exponentially to non-polynomial times once topology sizes in-

creased beyond seven nodes (in the order of years). The authors found that using a combination of binary partitioning and limiting the number of topologies found before ending execution allowed for them to reduce execution to polynomial times but the topologies found were not optimum.

Fencl *et al*. [19] used genetic algorithms to generate 4-connectivity topologies. The authors required higher degrees of connectivity between the routers for redundancy in non-loss tolerant networks. Fencl *et al*. [20] extended their work and used an approach based on genetic algorithms to calculate network topologies that meet a k-connectivity requirement. The authors constrain their work to having a fixed topology as a physical change in the topology (removal of a node or physical link that can be used) will change all logical topologies calculated by their algorithm. Average delay is used to evaluate the topologies found. The authors algorithm calculates 100 topologies before ending execution and does not check if the topology found is optimal or minimal. The complexity of the algorithm is O(N^4) (where N is the number of nodes) so it will not be suitable for large topologies with a high number of nodes as execution time will be too high,

Hsu *et al*. [26] used a modified version of Dijkstra's algorithm to ensure maximum tolerable delay in wireless mesh networks. The utilisation of the links in the topology is monitored, and when a delay threshold is exceeded, the modified version of Dijkstra's algorithm is executed, and a topology change occurs based on the routes calculated in order to ensure the delay requirement is met.

2.4 Path Finding and Routing

Within a topology, path finding and routing has been used in the literature to reduce power consumption and delay. Path finding and routing algorithms are generally quicker to execute compared to the algorithms used for topology design. Typical routing algorithms are based on span-

ning tree. The problem with spanning tree-based algorithms is that they are generally based on using a single weight metric and do not consider capacity constraints. A summary of surveyed work can be seen in Table 2.1.

2.4.1 Reducing Power Usage

Lin et al. [38] utilised path finding to balance power consumption with delay and utilisation in a backbone network. Their solution did not consider disjoint paths. Extending this previous work, Lin et al. [36] use path finding in order to save power in networks with bundled links (aggregated links to increase capacity). A heuristic algorithm based on Dijkstra's and Yen's *k*-shortest paths algorithm was used to calculate a single path for the demands of each source-destination pair of routers in the network to minimise the number of powered on links. The shortcomings of the authors work is that they did not consider dynamic topologies as their solution was for a wired backhaul network and that the time complexity of the algorithm is worse than $O(n^3)$ (where n is the number of nodes). Lin *et al*. [37] further extend their work by calculating two disjoint paths for each source-destination pair of nodes in the network. Their solution improves reliability but increases power consumption as more links are left powered on compared to when a single path is found.

2.4.2 Delay and Capacity

Narlikar *et al*. [45] used a heuristic-based routing algorithm to dynamically adjust the effective topology of a wireless backhaul network by changing the routes between the routers. The authors utilise scheduling and change routes by allowing certain links to be active during certain timeslots to guarantee capacity and delay. The authors [45] did not appear to consider when demand was low and the scheduler would still change the topology. Lin et al. [38] improve delay by splitting up flows and sending them

through more than one path when a given delay constraint is now met through a single path. The authors [38] have not considered packet loss when a path can no longer be used when a flow has been split up.

Bojic *et al*. [6] utilise path finding to improve capacity usage in small cell wireless backhaul networks. Dynamic paths are calculated in ring and mesh topologies to distribute the load of traffic to optimise the use of capacity. The authors noticed an increase in capacity but they did not consider the effect this had on the delay of the traffic.

2.5 SDN in Backhaul Networks

SDN is a recent networking architecture that separates the control and data functions of network appliances to enable a network to be logically centralized as well as having a programmatic approach to building, operating and managing a network [42, 29, 6]. OpenFlow is currently the most common protocol used to realise an SDN network [29]. Currently, the use of SDN in microwave backhaul topology control is in its infancy stages and work is limited.

2.5.1 Topology Discovery in SDN

Choi [9] incorporated a SDN-based centralised server in a Multi-Protocol Label Switching (MPLS), mobile backhaul based network for collecting neighbour information and topology construction. The Link Layer Discovery Protocol (LLDP) is used for neighbour discovery and Generalized Multi-Protocol Label Switching Path Computation Element (GMPLS PCE) based Topology discovery protocol (G-TOP) is used for topology construction. A client process running on the routers of the topology responds to request messages from the server. Pakzad *et al*. [51] also used LLDP to perform topology discovery in OpenFlow networks. The authors found that there were a high number of LLDP messages sent to the controller in

Table 2.1: Table summarising work in existing research. Objective refers to the target optimisation of the algorithms used by the authors'. An execution time of 'High' refers to the algorithm not being able to run in polynomial time.

Author(/s)	Objective	Method	Optimality	Execution Time
Topology Design				
Gerla and Kleinrock, 1977 [25]	Delay, throughput, reliability	Heuristics	Non-optimum	High
Gavish, 1992 [23]	Cost, queuing delay	Combinatorial optimisation	Non-optimum	High
Pierre *et al.*, 1995 [53]	Delay, reliability	Simulated annealing	Non-optimum	High
Kuo *et al.*, 2010 [31]	Cost	Network structure	N/A	N/A
Kumar and Banerjee, 2011 [30]	Cost, delay	Evolutionary algorithm	Non-optimum	Low
Li *et al.*, 2014 [35]	Cost	Heuristics	Non-optimum	Low
Schlinker *et al.*, 2015 [57]	Bandwidth, reliability, cost, complexity, scalability	Heuristics	Non-optimum	High

Li *et al*., 2016 [34]	Long links, traffic hops, link crosses, link distances	Heuristics	Non-optimum	Low
Topology Control				
Szlachcic, 2006 [61]	Delay	Genetic algorithms	Non-optimum	Low
Fencl *et al*., 2008 [19]	4-connectivity	Genetic algorithms	Non-optimum	Low
Hsu *et al*., 2008 [26]	Maximum tolerable delay	Modified Dijkstra's	Optimum	High
Szlachcic and Mlynek, 2009 [62]	Delay	Genetic algorithm	Non-optimum	Low
Chiaraviglio *et al*., 2009 [8]	Power consumption, utilisation	Greedy heuristics	Non-optimum	Low
Kamiyama, 2009 [28]	Cost, quality	Binary partitioning	Non-optimum	High
Yonezu *et al*., 2010	Power consumption	Enumerative	Optimum	High

Fencl *et al.*, 2011 [20]	Connectivity	Genetic algorithm	Non-optimum	High
Cuomo *et al.*, 2012 [16]	Power consumption	Graph pruning	Non-optimum	Low
Path Finding and Routing				
Narlikar *et al.*, 2010 [45]	Capacity, delay	Heuristic	Non-optimum	Low
Lin et al., 2012 [38]	Power consumption, delay	Heuristic	Non-optimum	Low
Lin et al., 2013 [36]	Power consumption	Heuristic	Non-optimum	Low
Bojic *et al.*, 2013 [6]	Capacity	Genetic algorithm	Non-optimum	Low
Lin *et al.*, 2014 [37]	Reliability	Heuristic	Non-optimum	Low

the standard implementation used in OpenFlow. The authors proposed a new implementation based on the original LLDP to pre-install flow rules to reduce the number of LLDP messages. Pentikousis *et al*. [52] created Mobileflow to realise their definition of a Software-Defined Mobile Backhaul. The authors defined MobileFlow Forwarding Engines (MFFEs) that are responsible for auto topology discovery by sending updates to the MobileFlow Controller (MFC). The MFC computes paths on the MFFE topology based on the information that it has stored. Pentikousis *et al*. [52] proposed a solution that appeared to be unique but the concepts were no different to a standard OpenFlow-based network, with it's entities given different names.

2.5.2 Obtaining Network and Topology Information in SDN

The Open Networking Foundation (ONF) [50] performed a collaboration to trail three use cases in a Proof of Concepts (PoC) that utilised an SDN-based approach for managing a wireless transport network. Their findings from the trails show that the hardware can be programmed to allow information to be aggregated on a central entity to perform decisions on a global scale. The focus of this trail was not to manage a topology to improve power consumption and QoS. However, the findings show that an SDN-based architecture can be used to aggregate the information needed to perform topology control.

Similarly, Santos and Kassler [56] use the same concepts as the ONF [50] to manage a wireless backhaul network. The SDN controller obtains statistics from the nodes in the topology to manage the entire network. Subsequently, route recalculations and topology changes are performed based on the information the controller has obtained. Fernandez-Fernandez *et al*. [21] perform energy aware routing using SDN. The authors did not obtain a optimal set of links to power off despite having a global view of the network. Lu *et al*. [40] proposed a network manager called HybNET

to manage network infrastructure. As part of HybNET, there is a built in topology management entity that collects physical topology information. With the information collected, there is a path finding entity that computes paths for the attached hosts to determine the topology needed. Fu *et al*. [22] proposed a hybrid hierarchical control plane that contains topology and routing management, similar to Lu *et al*. [40]. Dijkstra's algorithm is used as part of the routing manager to calculate paths using the information obtained from the topology manager. New paths are computed when the topology manager has detected a change to the topology. The authors of [40] and [22] did not use objective metrics typically used in topology control algorithms such as energy consumption or delay but their work does provide insights into the potential solutions in SDN-based mobile backhaul networks.

2.5.3 Topology Control in SDN

Wette and Karl [67] use OpenFlow enabled switches to perform virtual network topology configuration in Wavelength Division Multiplexing (WDM) networks. The authors solution routes as much traffic as possible through a single lightpath. If all traffic can be routed through a subset of existing lightpaths, the remaining lightpaths can be removed from the topology. Such techniques can be used in wireless based networks. FlowVisor has been used to ensure bandwidth and physical topologies in virtual networks [33, 59]. Within a production network, a slice of the available resources can be given to a virtual topology in order to perform simulations such as links going down. As multiple operators use the same base stations, being able to independently change a topology without affecting another operator is important as many operators cannot afford their own infrastructure.

2.6 Summary

In this chapter, background information on graphs and their use in networking applications was given. Existing work in the literature in the context of *topology design, topology control, path finding and routing, SDN in backhaul networks* was surveyed to use as inspiration for the algorithm proposed in this thesis. A summary of surveyed work in these fields can be seen in Table 2.1. Within these fields, there are algorithms to design, monitor, compute and alter topologies to meet objectives chosen by the authors.

In the next chapter, the problem is formulated mathematically to be solved using algorithms and techniques inspired by the literature.

Chapter 3

Problem Formulation and Notation Definitions

In this chapter, the notation used in the thesis is listed, where the terms used also elaborated. Following previous work in the literature, the mathematical representation of the problem to be solved and the representation of the network is given. Previous work yields sub-optimal topologies and require high execution times of the algorithm. The solution in this thesis will aim to have a low execution time and yield the optimal topology once the algorithm has been run.

3.1 Definitions of Terms

- The term **node** refers to the fundamental unit in graph theory that is connected together using edges. Nodes refer to the routers inside a microwave backhaul that make up the topology. The two will be used interchangeably in this thesis.
- The term **delay** refers to the transmission delay involved in transmitting traffic across a link.
- The term **link** is used to refer to the the microwave links between

the routers. A link may be composed of several microwave links aggregated together to form a single logical link with higher capacity.

- **Prune** refers to removing edges from a graph or in the case of a topology, the removal of link(s).
- **Outdoor unit (ODU)** refers to the transmitters that the microwave links are made up of. A single microwave link consist of one to eight ODUs.

3.2 Assumptions

1. Bidirectionality is assumed on the microwave links. This means both routers directly connected via a microwave link can transmit and receive at the same time.
2. The topologies are represented in undirected graphs. Traffic can be transferred in both directions along a link.
3. The indexes *i,j* and *j,i* into a matrix refer to the same value. For example current traffic demand, denoted below in Section 3.4.1 as this value reflects the total traffic along the link (traffic going both directions). This means effectively half the matrix is used.
4. The ODUs that are aggregated together to logically make up a link have identical MCS rates.

3.3 Representation of Variables

For the purpose of this thesis, all graphs are represented as matrices. The input variables are represented using adjacency matrices except the elements in the matrices will not only be 1's and 0's. The input variables model capacity, demand, delay and etc., will have varying values. A value

of zero in an element of the matrix will always mean that the vertices the indexes correspond to are not adjacent/neighbours and a non-zero value represents adjacency. Section 3.4.1 details the matrices used to formulate the problem.

3.4 Problem Formulation

The problem is formulated descriptively and formally using mathematical notation in this section.

3.4.1 Problem Description

A description of the problem can be written as: **Given** (i): a physical microwave backhaul topology comprising of microwave routers as nodes, and the links between them, (ii): the capacities of the links between the microwave routers, (iii): the total traffic demand along the links between the routers and (iv): the maximum tolerable delay, **Find** a least cost topology containing the nodes and the subset of links that must be powered on in order to meet capacity and delay demands, minimise delay and reduce power consumption.

3.4.2 Mathematical Description

A mathematical description of the problem is detailed as follows. **Given** a graph $G = (N, E)$, where $N = \{n_1, n_2, ..., n_n\}$ is the set of nodes in the graph, $|N| = n$ is the number of nodes (routers in the network), $E = \{e_1, e_2, ..., e_n\}$ is the set of edges in the network, $|E| = m$ is the number of edges (links between the routers), **Find** a graph $G' = (N, E')$ where $E' \in E$, such that delay is minimised, meets the maximum tolerable delay T_{max} and packet loss is minimised given the capacity and demand constraints.

Below are the constraints and variables imposed on the topology in the form of matrices. Each matrix corresponds to an input variable and is

indexed with *i* and *j* which are indexes to a matrix representing the link between routers *i* and *j*. This is denoted by

$$X = \begin{array}{c} \\ 1 \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & j & \dots & n \end{array} \\ \begin{pmatrix} x_{11} & x_{12} & x_{13} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & \dots & x_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & x_{n3} & \dots & x_{nn} \end{pmatrix} \end{array},$$

where *X* corresponds to one of the below variables used as inputs.

Using the representation denoted above as *X*, the below variables are used as inputs into the algorithm.

- The capacity (available capacity) supported on the link between routers *i* and *j*, $C_{i,j}$ in C, whereby C is defined as:

$$C = \begin{array}{c} \\ 1 \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & j & \dots & n \end{array} \\ \begin{pmatrix} c_{11} & c_{12} & c_{13} & \dots & c_{1n} \\ c_{21} & c_{22} & c_{23} & \dots & c_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_{n1} & c_{n2} & c_{n3} & \dots & c_{nn} \end{pmatrix} \end{array}.$$

  A value of 0 in the capacity matrix at link C_{ij} represents that there is no capacity between routers *i* and *j*. This is due to one of two factors: there are no physical microwave link(s) between routers *i* and *j* or routers *i* and *j* are no longer directly connected due to noise and fading. The unit of each value in the matrix is measured in Mbps.

  This is calculated with:

  - Modulation and coding scheme (MCS),

$$C_1 = \begin{array}{c} \\ 1 \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & j & \dots & n \end{array} \\ \begin{pmatrix} c1_{11} & c1_{12} & c1_{13} & \dots & c1_{1n} \\ c1_{21} & c1_{22} & c1_{23} & \dots & c1_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c1_{n1} & c1_{n2} & c1_{n3} & \dots & c1_{nn} \end{pmatrix} \end{array},$$

where each element in the matrix is an integer $[m] = 1, ..., 8$, which corresponds to an MCS bitrate in Mbps. The MCS rate is calculated using link budgeting, in order to take into account radio transmitter and wireless channel characteristics to determine received power on the receiving end.

– Number of active links,

$$C_2 = \begin{array}{c} \\ 1 \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & j & \dots & n \end{array} \\ \begin{pmatrix} c2_{11} & c2_{12} & c2_{13} & \dots & c2_{1n} \\ c2_{21} & c2_{22} & c2_{23} & \dots & c2_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c2_{n1} & c2_{n2} & c2_{n3} & \dots & c2_{nn} \end{pmatrix} \end{array},$$

where each element in the matrix is an integer, $[l] = 1, ..., 8$ representing the number of links supported. Each $C2_{ij}$ follows a uniform distribution $\sim U(1, 8)$ with each value generated with equal probability of $\frac{1}{8}$

- The current traffic demand on the link between routers *i* and *j*, f_{ij} in f, whereby f is defined as:

$$f = \begin{array}{c} \\ 1 \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & j & \dots & n \end{array} \\ \begin{pmatrix} f_{11} & f_{12} & f_{13} & \dots & f_{1n} \\ f_{21} & f_{22} & f_{23} & \dots & f_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{n1} & f_{n2} & f_{n3} & \dots & f_{nn} \end{pmatrix} \end{array},$$

which is calculated by multiplying a percentage with available capacity on the link between routers *i* and *j* with every $f_{i,j} \sim N(\mu, \sigma^2)$. μ is the control variable representing the average current traffic demand. This means every f_{ij} cannot be below 0 and cannot exceed the the capacity of the link C_{ij}. Mathematically this is $0 \leq f_{ij} \leq C_{ij}$. The unit of each element in the matrix is measured in Mbps.

- New demand requirements to be satisfied between each pair of routers *i* and *j* γ_{ij} in γ, whereby γ is defined as:

$$\gamma = \begin{array}{c} \\ 1 \\ i \\ \vdots \\ n \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & 2 & j & \dots & n \end{array} \\ \begin{pmatrix} \gamma_{11} & \gamma_{12} & \gamma_{13} & \dots & \gamma_{1n} \\ \gamma_{21} & \gamma_{22} & \gamma_{23} & \dots & \gamma_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \gamma_{n1} & \gamma_{n2} & \gamma_{n3} & \dots & \gamma_{nn} \end{pmatrix} \end{array},$$

  which is calculated by multiplying a percentage with residual capacity ($C_{ij} - f_{ij}$) (i.e $0 \leq \gamma_{ij} \leq C_{ij} - fij$) on the link between routers i and *j*, with every $\gamma_{ij} \sim N(\mu, \sigma^2)$. The control variable, μ represents the average new demand to be met. The unit of each element in the matrix is measured in Mbps.

3.5 Mathematical Formulation

3.5.1 Mathematical Formulation of Objectives 1 and 2

Given the definition and constraints of the problem in Section 3.4.1, the problem (for Objectives 1 and 2 in Section 1.2) can be formalised as follows:

Objective:

$$\min \frac{\sum_{k=1}^{|A|} \sum_{i=1,j=i+1}^{|A_k|} \frac{\frac{f_{P_i,P_j}}{C_{P_i,P_j} - f_{P_i,P_j}}}{\gamma_{P_i,P_j}}}{|A|}, \tag{3.1}$$

where:

A is a sequence containing paths for each source-destination pair of nodes,

$|A|$ is the length of A,

A_k is the $k-th$ element in A. It is a sequence of nodes that make up the path for a source destination pair, e.g $< A, F, C, G, E >$,

$|A_k|$ is the length of the A_k,

$i\ and\ j$ are indexes into A_k to obtain node pairs to get link information,

P_i, P_j are the i-th and j-th elements of A_k used to access the constraint matrices,

$i \neq j$,

i, j and j, i refer to the same pair of nodes, P_i, P_j in A_k,

Subject to:

$$0 \leq f_{ij} \leq C_{ij}, \tag{3.2}$$

$$0 \leq \gamma_{ij} \leq C_{ij} - f_{ij}, \tag{3.3}$$

$$\sum_{i=1,j=i+1}^{|A_k|} \frac{\frac{f_{P_i,P_j}}{C_{P_i,P_j} - f_{P_i,P_j}}}{\gamma_{P_i,P_j}} \leq T_{max}, \tag{3.4}$$

The objective in Eq. (3.1) minimises the average delay across the paths for all source destination pairs in the topology.

Equation (3.2) states that $f_{i,j}$, the existing flow demand on the link between routers *i* and *j* must be less than or equal to $C_{i,j}$, the capacity of the same link.

Equation (3.3) states that $\gamma_{i,j}$, the new flow demand between source node i and destination node j must be less than or equal to $C_{i,j}$ - $f_{i,j}$, the residual capacity between source node i and destination node j.

Equation (3.4) states that $\sum_{i=1,j=i+1}^{|A_k|} \frac{\frac{f_{P_i,P_j}}{C_{P_i,P_j} - f_{P_i,P_j}}}{\gamma_{P_i,P_j}}$, the delay along the path found for a source-destination pair of routers must be less than T_{max}, which is the maximum tolerable delay given as an input variable. This input variable signifies the delay that can be tolerated for the traffic in the

network. This represents certain QoS requirements for the traffic that is carried through the topology. $T_{path-i,j}$ is calculated by summing the delays $T_{i,j}$ across the links that are in the path for the source-destination pair. The delay for the link between nodes i and j, $T_{i,j}$ is calculated with $\frac{\frac{f_{i,j}}{C_{i,j}-f_{i,j}}}{\gamma_{i,j}}$ [25, 53, 3, 15].

Objective 2 is represented in Eq. 3.2 and 3.3. By ensuring the flow constraints are met, packet loss is minimised when a path is not calculated for source-destination pairs with demands that cannot be met.

3.5.2 Mathematical Formulation of Objective 3

Additionally, the third objective in Section 1.2 can be formalised with the following Equation.

Objective:

$$\min \sum_{k=1}^{n} \sum_{j=1}^{n} L_{ij}, \tag{3.5}$$

where:

L is a matrix containing the number of ODUs that make up the links in the topology,

n number of rows and columns in L,

L is initially initialised to $C2$, the links in the input topology,

$i \; and \; j$ are indexes into N to obtain the number of ODUs,

$i \neq j$,

i, j and j, i refer to the same entry in the matrix (as bidirectionality is assumed)

The objective in Eq. (3.5) minimises the number of links in the topology calculated. Priority is given to achieving Objectives 1 and 2 in Eq. 3.1 and meeting the constraints in Eq. 3.2, 3.3 and 3.4.

3.6 Summary

In this chapter, the terms used in this thesis are listed and defined, while the mathematical representation of variables are given. Additionally, the problem being solved in this thesis was given both descriptively and in mathematical formulation form. The three objectives set in Chapter 1 are denoted by two mathematical objectives in Eq. 3.1 and 3.5.

In the next chapter, the network model that is used to represent the problem formulation is introduced. The network model takes in the inputs described in the mathematical formulation, and uses an algorithm to meet the objectives.

Chapter 4

Network Model and Execution Environment

The problem formulated in the previous chapter is represented in a network model written in Python. This chapter gives details of the network model, such as how the input variables' representations are given. The network model provides the basis on which the algorithms detailed in the next chapter are run on. The model allows several aspects of the network to be measured and calculated such as wireless connectivity, power consumption and monetary cost measurement.

Together with the model, a discrete event packet level simulator was written to validate the results of the model. The simulator takes the calculated topology as an input to verify that packet loss and delay is minimised as required by objectives one and two.

Implementation details on the model, runtime environment and discrete event packet level simulator are also given in this chapter.

4.1 Network Model Components

The network model comprises of the input topology, the connectivity model which is then turned into input matrices. These matrices are then fed into

the algorithm described in Chapter 5 which outputs a set of routes and a minimum cost topology. The components of the network model can be seen in Fig. 4.1.

The remainder of the section will describe the input topology, which is used to calculate connectivity and the input matrices.

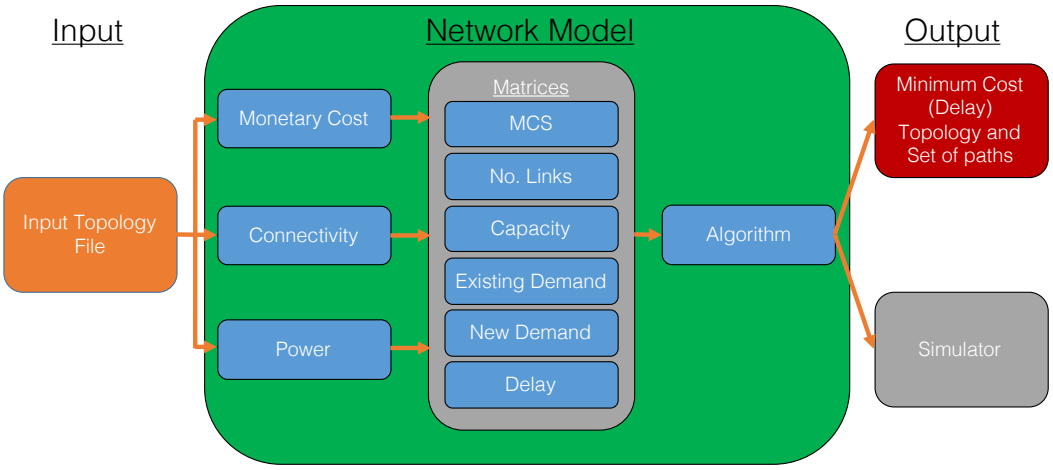

Figure 4.1: The components that make up the network model to represent the microwave backhaul which the algorithms detailed in Chapter 5 are run on.

4.1.1 Wireless Backhaul Network Topology for Evaluation

To model a microwave backhaul network within a program, information from a live topology is converted into an input file that can be parsed. The original data is defined in XML. This is parsed into a format that the network model can convert into a representation in code.

The format of the topology input files are defined as follows:

- **Line 1**: Contains all the nodes in the input topology. The names

declared in the file are used as the names in the topology diagram. A name to index mapping and an index to name map is created to access the matrices generated.

- **Line 2**: Contains the edge definitions between the nodes in the topology. The edges are formatted using "node_from-node_to". Bidirectionality is assumed meaning "node_1-node_3" and "node_3-node_1" refer to the same edge, although only one of the two is defined in the file.

- **Line 3**: Contains the cartesian coordinates of the nodes in the network. These are converted from (latitude,longitude) geographic coordinates with respect to a (0,0) cartesian coordinate point.

- **Line 4**: Contains the cost associated along the edges to transmit a Mb of traffic.

An example of an input topology file can be seen in Fig. 4.2. The file format follows the above.

```
            Topology.txt

Line 1:   node_1_name,node_2_name,...,node_n_name
Line 2:   node_x_name-node_y_name,...,node_p_name-node_q_name
Line 3:   node_1_name(x1,y1),...,node_n_name(xn,yn)
Line 4:   node_x_name-node_y_name($x),..., node_p_name-
            node_q_name($n)
```

Figure 4.2: Input topology file format that gets parsed and input into the network model.

4.1.2 Modelling Wireless Conditions

The MCS index is used to model the condition of the wireless channel the ODUs are transmitting on. In the model, a change in wireless channel

MCS Index	Modulation	Capacity (Mbps)	Signal Range (dBm)
0	Unreachable	0	$\geq$ -90
1	QPSK	62	-90 $>$x $\geq$ -85
2	16QAM	138	-85 $>$x $\geq$ -80
3	32QAM	216	-80 $>$x $\geq$ -75
4	64QAM	238	-75 $>$x $\geq$ -65
5	128QAM	311	-65 $>$x $\geq$ -55
6	256QAM	361	-55 $>$x $\geq$ -45
7	512QAM	405	-45 $>$x $\geq$ -35
8	1024QAM	424	$>$-35

Table 4.1: Table of MCS index corresponding to modulation schemes, rates and signal ranges used in the wireless model [48].

conditions is represented in the MCS matrix, C_1. There are 8 possible MCS indexes each corresponding to a integer $[m] = 0, ..., 8$. A higher MCS index indicates better channel conditions. A better channel condition yields higher capacity. The reverse holds, yielding with capacity decreasing as the channel condition worsens. In the event of poor channel conditions, the MCS index could decrease to represent a decrease on modulation and coding to account for the extra noise/fading etc. In the event that the link drops out, the MCS index will become 0. At this point the capacity of the link becomes 0.

The values corresponding to the MCS indexes, modulation schemes, capacity and signal range can be seen in Table 4.1. The values seen in Table 4.1 refer to the rates offered by Aviat Networks CTR8540 link of microwave routers using an 18 GHz ODU. The ODUs take up 55 MHz of spectrum, following standards defined by the European Telecommunications Standards Institute (ETSI).

To obtain the signal strength of the link between two nodes, the equation for link budgeting and path loss are described in Section 4.1.3. Once the signal strength is calculated, a value from Table 4.1 is used to deter-

mine the capacity of the link.

4.1.3 Link Budgeting Equation

The equation used to calculate received power on the receiver is described below [2, 32].

$$P_{rx} = P_{tx} + G_{tx} - L_{tx} - F_{fs} - L_m + G_{rx} - L_{rx}. \tag{4.1}$$

Where:

P_{rx} received power, is the received power or signal at the receiving node (measured in dBm),

P_{tx} transmit power, is the power which the sending node transmits the signal at (measured in dBm),

G_{tx} transmitter gain, (measured in dBi),

L_{tx} transmitter losses, (measured in dB),

F_{fs} path loss, (measured in dB),

L_m miscellaneous loss, (measured in dB),

G_{rx} receiver gain, (measured in dBi),

L_{rx} receiver losses, (measured in dB).

The input topology files currently do not have the following variables provided, so they were fixed inside the model for the simulations conducted in this thesis.

- P_{tx}, transmit power at 25 dBm,
- G_{tx} transmitter gain at 10 dBi,
- G_{rx} receiver gain at 10 dBi.

Path Loss Equation

Path loss in the wireless model for a microwave backhaul network is calculated using Friis Equation. This takes into account the euclidean distance between node placement. Friis Equation can be seen below [2, 32].

$$PathLoss = (\frac{4.\pi.d.f}{c})^2. \tag{4.2}$$

Where:

f is frequency, in Hz,

d is range or distance between nodes, in m,

c is speed of light

This equation can be simplified down to:

$$PathLoss = 20log_{10}(d) + 20log_{10}(f) - 27.55, \tag{4.3}$$

where:

d is measured in m,

f is measured in MHz.

4.1.4 Modelling Connectivity

The third line of the input file contains coordinates to the nodes in the network as described in Section 4.1.1. The coordinates are used to visualise the network inside the model and to compute whether the directly connected nodes have connectivity. The cartesian coordinates are used to calculate euclidian distances between the nodes.

Once euclidian distances are calculated between the nodes, connectivity can be determined using the link budgeting formula defined above in Section 4.1.3. The wireless model contained within the network model is used to determine connectivity using link budgeting and free space path loss. Connectivity in the model can be adjusted by introducing noise denoted by the L_{tx} or R_{tx} variables in Eq. 4.1.

4.1.5 Modelling Power Consumption

Power consumption in the network can be measured by the total number of ODUs in the network that make up the microwave links. This is a com-

monly used method in the literature to determine the energy efficiency of a backhaul topology [41]. The input topology is the baseline power consumption with every ODU powered on. From this output topology calculated by the algorithm, the total number of ODUs are used to calculate the power consumption. The baseline power consumption is compared against this value from the new output topology to determine if power has been saved.

No new links can be added into the topology unless a new topology is defined. This means the input topology is the upper bound in power consumption. If no links can be pruned from the topology (for example, there are no loops), then the output topology cannot use more power than the input topology as it will have the same number of active ODUs.

To further model power consumption, this can be converted into watts, the unit typically used to measure power consumption and $, dollars. The Aviat 18 GHz ODU used in the model uses 30 Watts when operating at its maximum transmit power which is assumed for the wireless model. This is multiplied by the cost of power at 30 cents per kWh to obtain the total system cost.

4.1.6 Modelling Cost

Work in the literature [53, 25] use monetary cost as the objective when creating the initial topology. This cost is the monetary cost for the operator in running the links (either in power or spectrum licensing). The numbers calculated are used by the operator to determine how much they should charge their customers that use the network to make a profit.

Calculating the cost in the resulting topology is performed by summing up the costs of transmitting the existing demand and the new demand. Line 4 of the topology input file defines the cost to transmit per Mb of traffic. The lower the total cost, the lower the expense is for the operator.

An example of the link costs can be seen in Fig. 4.3.

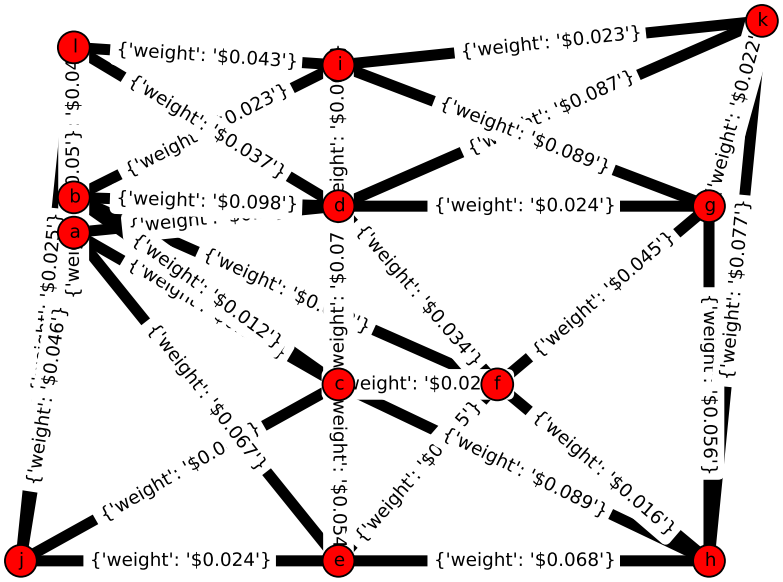

Figure 4.3: The monetary costs transmitting through each link in a topology. Each monetary value represents the cost to the operator when transmitting per Mb of data.

4.2 Discrete Event Packet Level Simulator

A packet level discrete event simulator was written in Python to validate the results obtained from the model. The output topology calculated by the algorithm is the analytical results to be validated with the packet level simulator. The simulator provides results to perform comparisons with the analytical model.

As shown in Fig. 4.1, the packet level discrete event simulator is external from the network model. The simulator takes in the outputs of the algorithm as its inputs. SimPy was used as the discrete event simulation

framework. Packets are generated and sent through the virtual network in the simulator. Queuing is applied at the routers to produce realistic results.

When the output topology and routes are calculated from the algorithm within the model, the simulator is initialised taking them in as arguments. The simulator generates fixed size packets of 1000 bytes according to the flow demands from the matrices (f and γ defined in Section 3.4) in the model following an exponential distribution. The packets with a certain pair of source-destination routers in the topology use the paths calculated from the algorithm. The results are recorded and averaged when the simulation is completed for comparison with the model.

4.3 Summary

In this chapter, the network model which represents the network is given. The network model facilitates the execution of the algorithm used to control the topology and allows evaluation tests to take place. The packet level simulator used to validate the results of the algorithm is also given in this chapter.

In the next chapter, the design of the algorithm executed within the network model as introduced in this chapter will be detailed. Microwave backhaul networks are dynamic, requiring topology control algorithms that can execute quickly.

Chapter 5

Topology Control Algorithm Design

In this chapter, the design of the algorithm used to minimise packet loss and delay is detailed. Microwave backhaul networks are dynamic and may require topology changes in the event of changing weather or traffic conditions. Simultaneously, packet loss and delay needs to be minimised yet the solutions in the literature yields sub-optimum topologies. The CUT and OptiCUT algorithms presented below use the brute force algorithm as the basis in design yet yield the same optimum topologies as output while reducing execution time.

5.1 Inputs and Outputs into Algorithms

This section lists the inputs that are used in the brute force, CUT and OptiCUT algorithms and the outputs after executing the algorithms.

Inputs into Algorithm

- $G = (N, E)$: the input topology containing the existing routers and links,

- C_1: Modulation and Coding Scheme matrix C_1 calculated ,
- C_2: Number of links matrix,
- f: Current demand matrix,
- γ: New demand matrix,
- T_{max}: Max tolerable delay variable (seconds).

Outputs of Algorithm

- $G' = (N, E')$: is the output topology from the algorithm containing the nodes of the original input topology, and a subset of the links from the original topology,
- $paths = \{G'_{a,a} = (N_{a,a}, E_{a,a}), \cdots, G'_{i,j} = (N_{i,j}, E_{i,j})\}$ *where* $\{E_{a,a}, \cdots, E_{i,j}\} \subset E$ *and* $\{N_{a,a}, \cdots, N_{i,j}\} \subset N$: is a set of paths for each pair of source-destination routers in the topology.

5.2 Path and Topology Computation using the Brute Force Algorithm

Using the brute force algorithm described in Procedure BruteForceAlgorithm yields the optimal topology and paths. Optimality in the context of this thesis means minimal loss and delay.

5.2.1 Brute Force Algorithm Details

The simplest method of calculating a minimum cost topology is to compute every possible path for each source-destination pair. The brute force method solution is based on a complete state enumeration [13]. The complete state enumeration method described in the literature is used to generate every possible combination of edges in a graph. A complete state

enumeration in the context of this thesis is modified to give every possible simple path (a path with no repeated nodes) for each source-destination router in the network with respect to an input topology containing the routers and links. This means that the algorithm computes paths using every link in the network. The brute force algorithm is the basis which the CUT and OptiCUT algorithms are designed from as it allows the minimal delay topology (and paths) to be calculated. The pseudo-code for the brute force algorithm is shown in Procedure BruteForceAlgorithm.

5.2.2 Analysis of Brute Force Algorithm

The brute force algorithm finds the optimal topology and consequently the optimum paths for each source-destination pair of routers in the input topology. This is due to the algorithm iterating over all possible paths, and then discarding all paths that cannot be used as a result of the residual capacity constraints. From these remaining paths, the path with the lowest total delay (sum of all delays across the links in the path found) is chosen for the demand to be met for a given source-destination pair if it meets the maximum tolerable delay given as an input.

The brute force algorithm was implemented in Python within the network model. To determine whether the algorithm can be optimised, a software profiler was used to analyse the number of function calls and total time spent within these functions.

The profiler is started right before algorithm the executed and stopped right after execution within the code of the network model. This measures the execution time the model spends on the algorithm and not when it is computing the matrices. The results of running the profiler can be seen in Fig. 5.1

In Fig. 5.1, cProfile shows that the brute force algorithm spends almost 80% of the measured total execution time on the functions *constrained_paths* and *_all_paths_constraint*. 2.026 and 2.152 seconds are spent in the *con-*

Procedure BruteForceAlgorithm

input : G, C_1, C_2, f, γ, T_{max} (variables shown above in Sec. 5.1)

output: $G' = topo$, $paths$ = $shortest\_paths$

```
1  C = C_1 * C_2
2  R = C − f
3  topo = new_graph();
4  shortest_paths = []
5  for each i, j pair ∈ N do
6      paths = compute_all_simple_paths(G, i, j);
7      paths_capacity = [];
8      paths_delay = [];
9      for each path in paths do
10         if meets_capacity(path, R, γ) then
11             add path to paths_capacity;
12         end
13     end
14     for each path in paths_capacity do
15         if meets_delay(path, T_max) then
16             add path to paths_delay;
17         end
18     end
19     min_path = lowest_delay(paths_delay);
20     add edges in min_path to topo;
21     add min_path to shortest_paths;
22 end
```

```
16374085 function calls (16374049 primitive calls) in 5.640 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
       36    2.026    0.056    5.640    0.157 problemSetupSpanningTreeAlgorithm.py:484(constrained_paths)
   493236    2.152    0.000    3.016    0.000 problemSetupSpanningTreeAlgorithm.py:807(_all_paths_constraint)
  5300082    0.318    0.000    0.318    0.000 {ord}
  2987280    0.277    0.000    0.277    0.000 {next}
3288494/3288458    0.223    0.000    0.223    0.000 {len}
   612875    0.178    0.000    0.178    0.000 {range}
   986400    0.146    0.000    0.146    0.000 {method 'pop' of 'list' objects}
  1718822    0.137    0.000    0.137    0.000 {method 'append' of 'list' objects}
   493200    0.116    0.000    0.116    0.000 {iter}
   493200    0.068    0.000    0.068    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:380(__getitem__)
       57    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:731(add_edge)
       36    0.000    0.000    0.000    0.000 problemSetupSpanningTreeAlgorithm.py:801(all_paths_constraint)
      216    0.000    0.000    0.000    0.000 {chr}
       57    0.000    0.000    0.000    0.000 {method 'update' of 'dict' objects}
       36    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:363(__len__)
       57    0.000    0.000    0.000    0.000 {method 'get' of 'dict' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Figure 5.1: Execution times (shown in the tottime column) of the brute force algorithm shown in Procedure BruteForceAlgorithm.

strained_paths and *_all_paths_constraint* functions respectively. This is the function that was targeted with improvements to be made in terms of execution in the CUT algorithm (shown in Procedure CUTAlgorithm).

5.3 CUT Algorithm

The brute force algorithms allows for the optimal topology in terms of delay to be obtained but its high execution time makes it unsuitable for a microwave backhaul network when a topology recalculation needs to be performed quickly. The bottleneck in setting up the topology is during the *compute_all_simple_paths* step in Procedure BruteForceAlgorithm as shown by measuring the execution time in Section 5.2.2. All possible paths from a source to a destination node is found before being processed to determine whether they are suitable, given the capacity and delay constraints. The goal of this algorithm was to improve execution time, whilst retaining optimality in terms of topology and paths calculated which the brute force algorithm holds.

5.3.1 CUT Algorithm Details

The CUT algorithm targets the *constrained_paths* and *_all_paths_constraint* functions which together makes up almost 80% of the total execution time

of the algorithm. The *_all_paths_constraint* function is called from the *constrained_paths* function to calculate every possible path from a source to destination router. By first pruning off the links that cannot be used due to capacity constraints denoted in Eq. 3.2 and 3.3, the search space for calculating the paths is reduced, which results in execution times of this new algorithm taking less time than the brute force algorithm.

5.3.2 Analysis of CUT Algorithm

Figure 5.2 shows the results of running cProfile to measure the execution time of the CUT algorithm. Compared to the brute force algorithm, the time the CUT algorithm spends on the *constrained_paths* and *_all_paths_constraint* functions have decreased from 2.026 and 2.152 to 1.165 and 0.527 seconds respectively.

The pruning step shown in line 5 of Procedure CUTAlgorithm removes all links that do not meet capacity before the *_all_paths_constraint* function is run (denoted by line 6 in Procedure CUTAlgorithm). This new algorithm improved the execution times of the *constrained_paths* function by over 42% and the *_all_paths_constraint* function by over 75%.

```
6540926 function calls (6540890 primitive calls) in 2.193 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
       36    1.165    0.032    2.190    0.061 problemSetupSpanningTreeAlgorithm.py:484(constrained_paths)
   119665    0.527    0.000    0.736    0.000 problemSetupSpanningTreeAlgorithm.py:807(_all_paths_constraint)
  3284844    0.180    0.000    0.180    0.000 {ord}
   789596    0.071    0.000    0.071    0.000 {next}
   239664    0.066    0.000    0.066    0.000 {range}
937228/937192    0.060    0.000    0.060    0.000 {len}
   627321    0.046    0.000    0.046    0.000 {method 'append' of 'list' objects}
   268470    0.038    0.000    0.038    0.000 {method 'pop' of 'list' objects}
   134235    0.019    0.000    0.019    0.000 {iter}
   134235    0.017    0.000    0.017    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:380(__getitem__)
       36    0.001    0.000    0.003    0.000 problemSetupSpanningTreeAlgorithm.py:456(create_pruned_graph)
     1005    0.001    0.000    0.001    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:731(add_edge)
     2112    0.000    0.000    0.000    0.000 {chr}
     1041    0.000    0.000    0.000    0.000 {method 'update' of 'dict' objects}
      324    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:409(add_node)
       36    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:260(__init__)
     1005    0.000    0.000    0.000    0.000 {method 'get' of 'dict' objects}
       36    0.000    0.000    0.000    0.000 problemSetupSpanningTreeAlgorithm.py:801(all_paths_constraint)
       36    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:363(__len__)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Figure 5.2: Execution times (shown in the tottime column) of the CUT algorithm shown in Procedure CUTAlgorithm.

Procedure CUTAlgorithm

input : G, C_1, C_2, f, γ, T_{max} (variables shown above in Sec. 5.1)

output: $G' = topo$, $paths$ = $shortest\_paths$

```
1  C = C_1 * C_2
2  R = C − f
3  topo = new_graph();
4  shortest_paths = []
5  for each i, j pair ∈ N do
6      pruned_graph = graph_capacity(N, E, R, i, j)
7      paths = compute_all_simple_paths(pruned_graph, i, j);
8      paths_capacity = [];
9      paths_delay = [];
10     for each path in paths do
11         if meets_capacity(path, R, γ) then
12             add path to paths_capacity;
13         end
14     end
15     for each path in paths_capacity do
16         if meets_delay(path, T_max) then
17             add path to paths_delay;
18         end
19     end
20     min_path = lowest_delay(paths_delay);
21     add edges in min_path to topo;
22     add min_path to shortest_paths;
23 end
```

5.4 OptiCUT Algorithm

The CUT algorithm in Procedure CUTAlgorithm improves upon the brute force algorithm in Procedure BruteForceAlgorithm. The execution time of *_all_paths_constraint* function has been greatly reduced from by over 75% from 2.152 to 0.527 seconds. The *constrained_paths* function still makes up over 50% of the total execution time of the algorithm. This was the next target of optimisation to improve execution time.

5.4.1 OptiCUT Algorithm Details

From the cProfile, it was observed that the CUT algorithm executes the capacity checking step in the function *meets_capacity* shown on lines 10-13 in Procedure CUTAlgorithm.The pruning step on line 5 already removes the edges that cannot be used that do not do not meet the capacity requirements defined in Eq. 3.2 and 3.3. The capacity checking stage of the CUT algorithm was removed in this optimised algorithm to further improve execution times as there is no need to iterate over the paths found to check for capacity.

The pseudocode for the OptiCUT algorithm is shown in Procedure OptiCUTAlgorithm.

5.4.2 Analysis of OptiCUT Algorithm

The execution time of the *constrained_paths* function is further reduced in the OptiCUT algorithm from 1.165 to 0.757 seconds when compared to the CUT algorithm. This is a further improvement of 35% over the CUT algorithm and a 62% improvement over the brute force algorithm.

Procedure OptiCUTAlgorithm

input : G, C_1, C_2, f, γ, T_{max} (variables shown above in Sec. 5.1)

output: $G' = topo$, $paths$ = $shortest\_paths$

```
C = C_1 * C_2
R = C − f
topo = new_graph();
shortest_paths = []
for each i, j pair ∈ N do
    pruned_graph = graph_capacity(N, E, R, i, j)
    paths = compute_all_simple_paths(pruned_graph, i, j);
    paths_delay = [];
    for each path in paths do
        if meets_delay(path, T_max) then
            add path to paths_delay;
        end
    end
    min_path = lowest_delay(paths_delay);
    add edges in min_path to topo;
    add min_path to shortest_paths;
end
```

```
4539597 function calls (4539561 primitive calls) in 1.682 seconds

   Ordered by: cumulative time

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
       36    0.757    0.021    1.679    0.047 problemSetupSpanningTreeAlgorithm.py:687(constrained_paths_optimised)
   119665    0.539    0.000    0.767    0.000 problemSetupSpanningTreeAlgorithm.py:807(_all_paths_constraint)
  1642582    0.094    0.000    0.094    0.000 {ord}
   789596    0.073    0.000    0.073    0.000 {next}
817599/817563    0.054    0.000    0.054    0.000 {len}
   507656    0.039    0.000    0.039    0.000 {method 'append' of 'list' objects}
   268470    0.038    0.000    0.038    0.000 {method 'pop' of 'list' objects}
   120035    0.035    0.000    0.035    0.000 {range}
   134235    0.034    0.000    0.034    0.000 {iter}
   134235    0.018    0.000    0.018    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:380(__getitem__)
       36    0.001    0.000    0.003    0.000 problemSetupSpanningTreeAlgorithm.py:456(create_pruned_graph)
     1005    0.001    0.000    0.001    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:731(add_edge)
     1968    0.000    0.000    0.000    0.000 {chr}
     1041    0.000    0.000    0.000    0.000 {method 'update' of 'dict' objects}
      324    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:409(add_node)
       36    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:260(__init__)
     1005    0.000    0.000    0.000    0.000 {method 'get' of 'dict' objects}
       36    0.000    0.000    0.000    0.000 problemSetupSpanningTreeAlgorithm.py:801(all_paths_constraint)
       36    0.000    0.000    0.000    0.000 /Library/Python/2.7/site-packages/networkx-2.0.dev_20160621121357-py2.7.egg/networkx/classes/graph.py:363(__len__)
        1    0.000    0.000    0.000    0.000 {method 'disable' of '_lsprof.Profiler' objects}
```

Figure 5.3: Using Python's cProfile tool to determine execution times (shown in the tottime column) of the OptiCUT algorithm shown in Procedure OptiCUTAlgorithm.

5.5 Execution Times of Algorithms

The total execution times of the algorithms were measured using cProfile. Figure 5.4 compares the measured execution times of the three algorithms. The brute force algorithm has the highest execution time of the three at 5.65 seconds. The OptiCUT algorithm has the lowest execution time with an improvement of over 70% compared to the brute force algorithm.

Decreasing the search space of the *constrained_paths* and *_all_paths_constraint* functions have contributed to this improvement in execution time. The search space increases exponentially as links are added as that would increase the total number of paths. Removing redundant links that already meet capacity decreases this search space which decreases the execution time.

5.6 Optimality of CUT and OptiCUT Algorithms

The performance of both the CUT and OptiCUT algorithms have greatly reduced the computational time of the paths as described in Section 5.5. The chosen method of validating the algorithms produced is to compare the paths found in both the algorithms against the brute force algorithm.

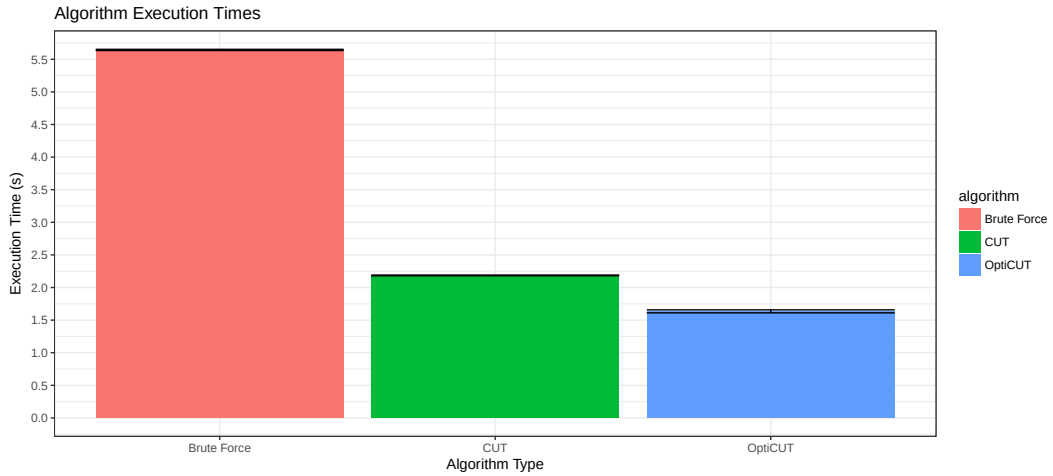

Figure 5.4: The executions times of the three algorithms when using cProfile to evaluate time taken over 100 executions.

The brute force algorithm is guaranteed to find the optimal solution as it iterates through all possible paths and discards paths that do not meet capacity, and then furthermore, selects the path with the lowest delay if the total delay along the path is lower than the maximum tolerable delay.

To determine whether the CUT and OptiCUT algorithms produce similar results to the original brute force algorithm, we compare the paths found for each algorithm. The paths found for each source destination pair along with the associated delay are logged into a file for each simulation of the algorithm. Note that the same set of inputs are used for each comparison of the algorithm.

5.7 Summary

In this chapter, the brute force algorithm which is the basis for the CUT and OptiCUT algorithm is introduced. The brute force algorithm yields the optimum topology and paths but it is slow when the number of nodes increases. The CUT and OptiCUT algorithms improve upon the bottleneck

functions in the brute force algorithm. Execution time of the OptiCUT algorithm has shown to have an overall 70% improvement in execution time when compared to the brute force algorithm.

In the next chapter, the CUT and OptiCUT algorithms (both yielding same results) are evaluated using a set of metrics.

Chapter 6

Evaluation

This chapter presents the evaluation performed on the algorithm(s) proposed in Chapter 5. The algorithm is implemented in the network model described in Chapter 4 and we simulate network traffic to evaluate the model calculated. A list of metrics is used to evaluate the algorithm and determine trends in delay and power consumption as traffic levels change. The list of metrics is used to evaluate two use case topologies of differing size and connectivity.

6.1 Evaluation Metrics

This section details the metrics used to evaluate the algorithm that is implemented inside the network model described in Chapter 4. Each metric below is the dependent variable within the results. Average utilisation is used as the independent variable that is perturbed to determine changes to these metrics.

Results and data is generated from each simulation of the network. This data is recorded into files and used as points for the graphs.

- **Path Delay** - This is the average delay for the paths computed for each pair of routers in the network. The paths found for each pair

of routers are chosen for the delay value. These values are averaged. This is calculated using the formula denoted in Eq. 3.1.

- **Path Diversity** - Every possible path given capacity constraints is computed for each pair of nodes. The path chosen for the pairs of nodes are chosen from these sets of paths. This metric evaluates the pool of paths that the algorithm computes from. Mathematically, this is denoted in Eq. 3.4.

- **Power Consumption** - This is the average number of active pairs of ODUs in the topology. The original number of pairs of ODUs in the input topology is used as the baseline to compare to the output topology. With information on the power consumption of the ODUs, this can be translated into monetary costs. The formula for calculating the total number of links in the network is denoted in Eq. 3.5

- **Monetary Cost** - This is the cost of transmitting the traffic demand through the network. Associated with each link in the topology, there is a monetary cost per Mb of traffic that is transmitted through the network. The cost of transmitting the traffic is determined by averaging the costs for each simulation of the network.

An additional metric used to evaluate the algorithm produced is execution time. Execution time in the context of the algorithm is defined as the time it takes to compute the output topology and the paths for each pair of routers given the input variables.

6.2 Steps Followed to Obtain Results

This section describes the steps taken to obtain the results from the simulations. The general steps of the simulations is listed below and a detailed explanation to some of the steps are also detailed below. The input into the algorithm is denoted in Step 1 and the output in Step 7.

- **Step 1**: Input topology file
- **Step 2**: Calculate matrices
- **Step 3**: Generate network representation
- **Step 4**: Compute output topology
- **Step 5**: Run packet level simulator
- **Step 6**: Write results to file
- **Step 7**: Repeat until number of samples met

A flow chart of the evaluation process can be seen in Fig. 6.1.

Step 2 generates the matrices to use as inputs into the algorithm. The topology input file contains coordinates that are used to determine node placement. The coordinates of the nodes are on the cartesian coordinate system. Having the nodes on the cartesian coordinate system allows the euclidian distance between nodes to be calculated. Connectivity is determined by the equations detailed in Section 4.1.2. Taking the euclidian distance and wireless channel characteristics (transmit power, antenna gain shown in Section 4.1.2) determines whether the nodes can reach each other.

Step 3 takes the matrices calculated in step 2, and initialises a model of the network that contains the routers and links in the topology. This network model provides the base on which the algorithm is run.

Step 4 runs the CUT or OptiCUT algorithm proposed in Chapter 5 within the network model to obtain the minimum cost topology and the paths for each pair of nodes with the lowest delays. This minimum cost topology is validated in step 6 using a discrete event packet level simulator which is explained in detail in Section 6.5.

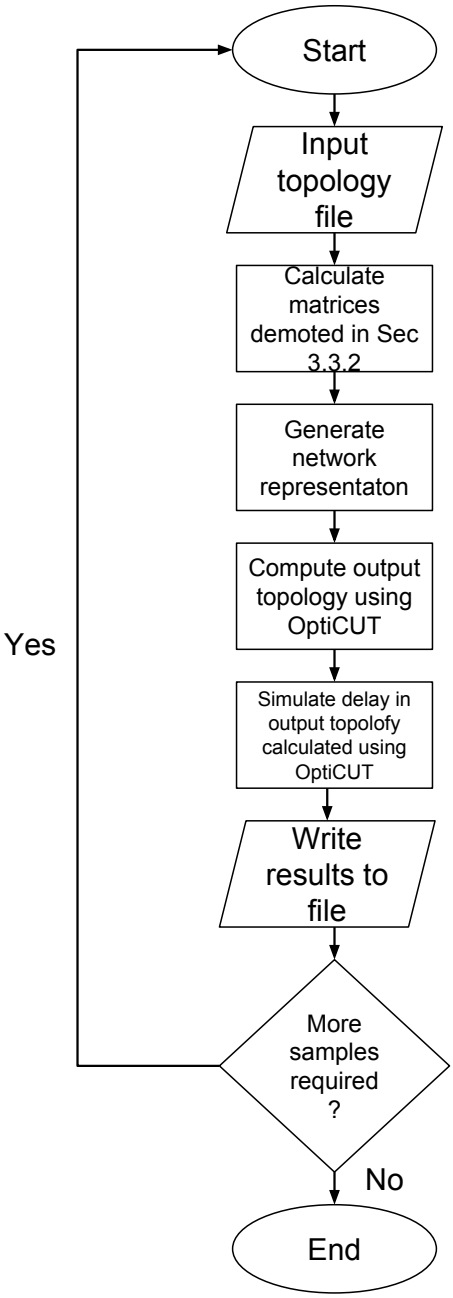

Figure 6.1: Flow chart showing process of obtaining results for evaluation of the algorithms.

6.3 Use Case Topologies

Two use case topologies are used as the base topologies in the network model described in Chapter 4. The topologies provided by Aviat Networks are defined in Extensible Markup Language (XML). A parser takes these XML files in and outputs a file in the format the network model expects.

The topologies used to evaluate the algorithm introduced in Chapter 5 are shown in Fig. 6.2 and 6.3. The first use case topology shown in Fig. 6.2 is a mesh topology with a high level of connectivity, with routers on average having a degree of connectivity of 6. The second use case topology shown in Fig. 6.3 is a ring topology with a larger number of nodes, but the average degree of connectivity is lower at 3. The degree of connectivity refers to the number of other routers each router is connected to.

The topology diagrams shown in Fig. 6.2 and 6.3 display the routers and links in the topologies. The number of pairs of ODUs are not represented. The edges of the graphs show that there is a physical link between the pairs of nodes that they connect together.

These topologies represent live production networks which contain a degree of redundancy as they are not already in a tree topology. Having loops in the network allows for it to be simplified by the algorithm, detailed in Chapter 5 to minimise delay and reduce energy consumption.

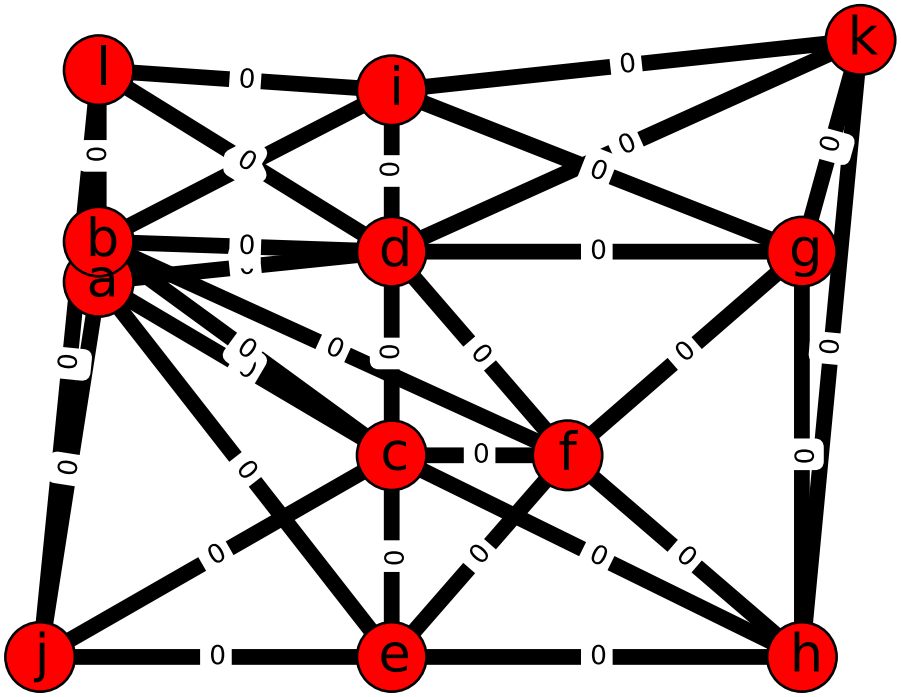

Figure 6.2: First use case of a mesh topology with a high degree of connectivity from the routers in the network.

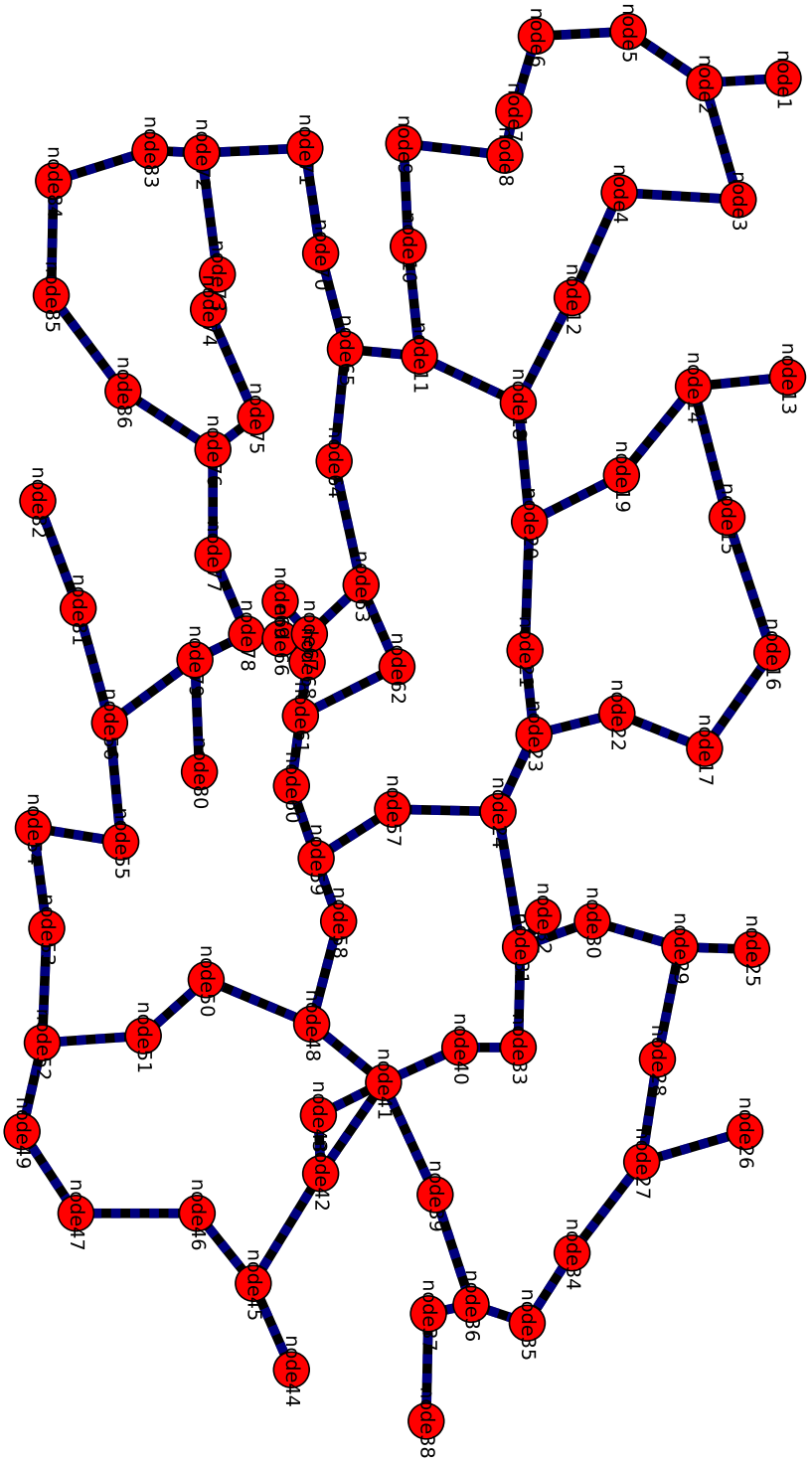

Figure 6.3: Second use case of a production network with a large number of routers but a lower degree of connectivity from the routers in the network.

6.4 Model Results

The results of the evaluation of the algorithm proposed in Chapter 5 using the metrics described in Section 6.1 are presented in this section.

6.4.1 Delay

Minimising delay is the second objective of this thesis. Using delay as an evaluation metric determines whether this objective has been met. The equation used to calculate delay is $T_{i,j} = \frac{\frac{f_{i,j}}{C_{i,j}-f_{i,j}}}{\gamma_{i,j}}$. To minimise average delay in the network, the minimal delay path for each source-destination pair needs to be found in order to calculate the minimum cost topology.

The average delays in the minimum cost topologies is found from using the two use case topologies (shown in Fig. 6.2 and 6.3) in simulations. Figure 6.4 and 6.5 shows the average delay across the paths in the output pruned topologies when mean utilisation is perturbed. Mean utilisation in Fig. 6.4 and 6.5 follows a normal distribution with every $f_{i,j}$ and $\gamma_{i,j}$ calculated with a truncated normal distribution i.e. $\sim N(\mu, \sigma^2)$. In Fig. 6.4 and 6.5, the boxplots shows the trend of delay increasing as mean utilisation increases. There is little spread (range) in the data up until mean utilisation is at 0.7. Once utilisation increases above that point, the variability in the data (between the upper and lower quartiles) increases with a higher spread in the values above the upper quartile and below the lower quartile. The insights gained from these observations is that delay is less predictable for high utilisation values. This means the average utilisation of the links should be kept lower by having more links enabled if predictability is desired.

When using mean to calculate the average of all the results, the trend shows that there is an exponential increases as utilisation increases.

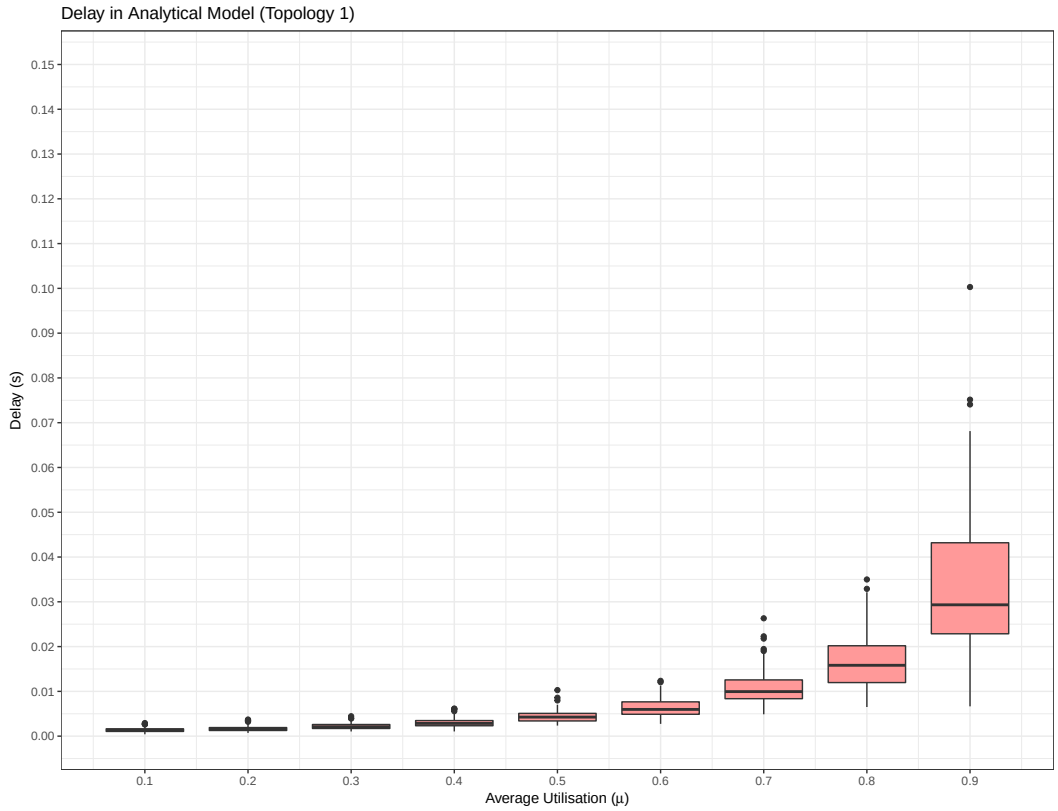

Figure 6.4: Delay in the topology calculated by the algorithm when using the first use case topology.

Perturbing Input Variables

The implementation of the network model described in Chapter 5 performs a topology recomputation when an input variable changes. This could be demand along a link increasing or decreasing, the MCS index decreasing due to interference or losing a pair of ODUs on a link which decreases capacity.

The input variables are represented as adjacency matrices as described in Section 3.4. The values within these adjacency matrices are changed to represent a perturbation of the input variables. This process was automated in the simulations performed for the evaluation of the algorithm.

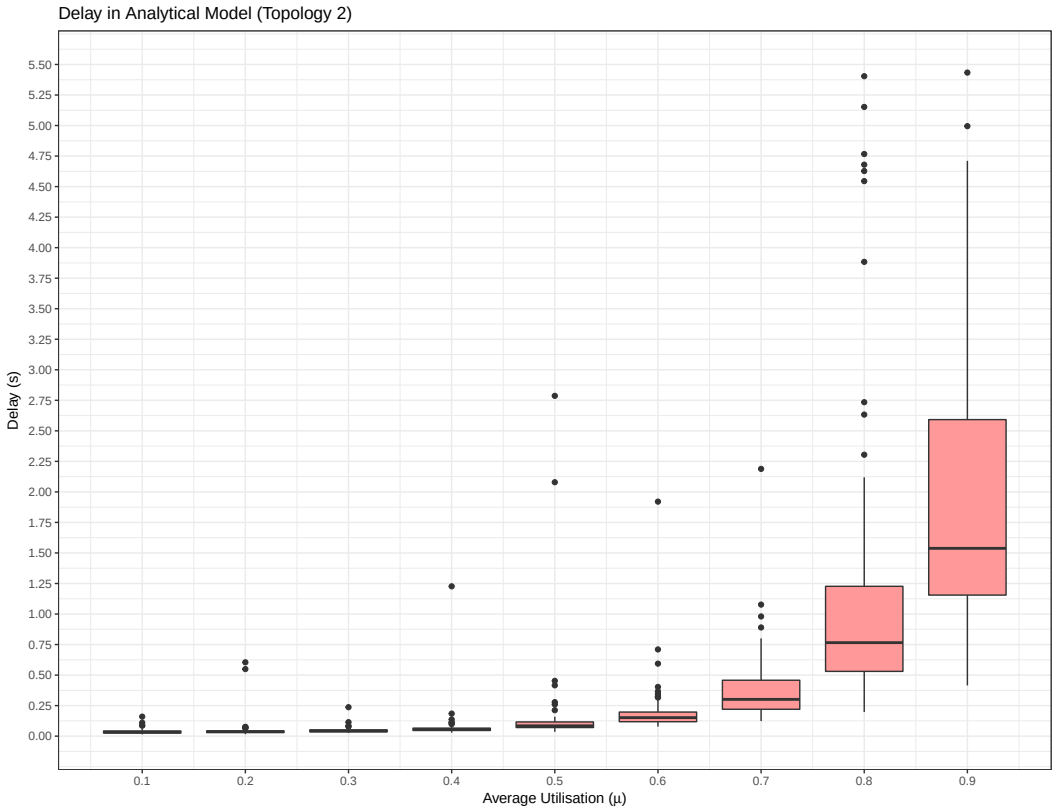

Figure 6.5: Delay in the topology calculated by the algorithm when using the second use case topology.

The use case perturbation used in this evaluation is rain fade which caused a microwave link to go down. In Section 4.1.4, the connectivity aspect of the network model is used to determine whether two directly connected nodes are able to communicate via the common link. Interference in the form of noise is introduced along the link to represent the rain fade along the wireless channel that the ODUs are transmitting on. The extra noise causes the MCS index of the link to decrease to 0, which results in the link having 0 Mbps of capacity, effectively removing that link. At this point the algorithm is rerun and performs a recomputation of the topology and the minimum cost paths.

By removing a link and performing a topology recalculation, the average delay in the link removed topology is higher than the pruned topology. This confirms the delay values calculated by the algorithm shown in Section 6.4.1 are minimal.

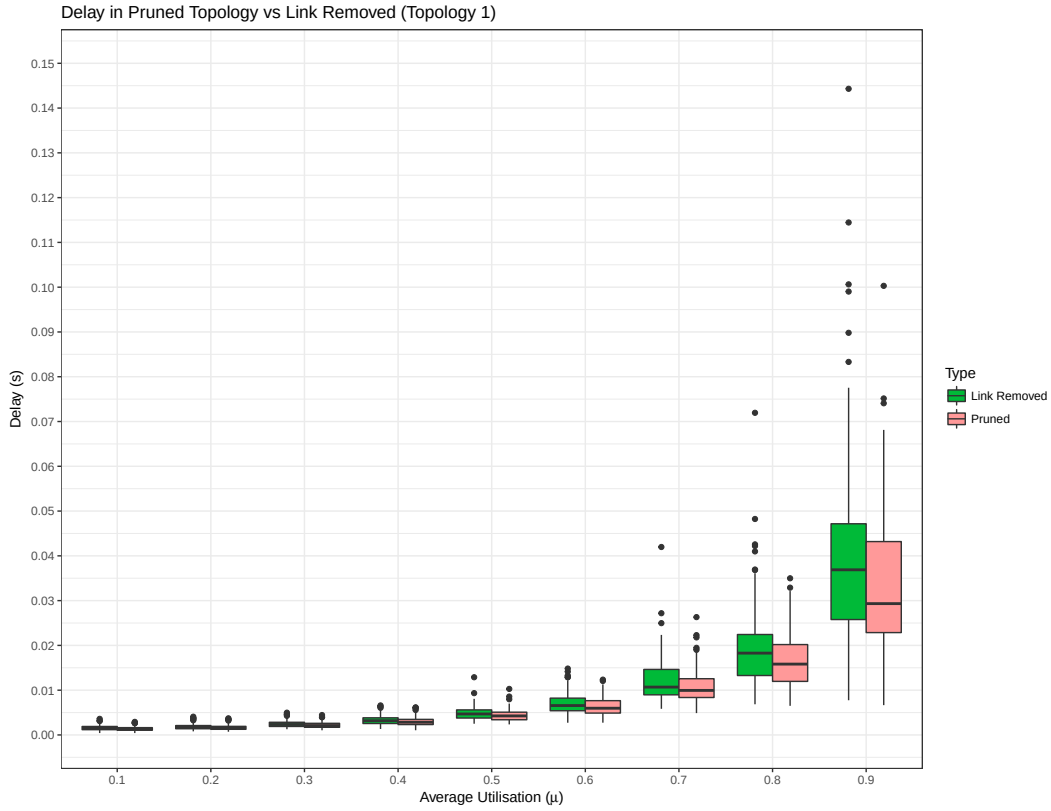

Figure 6.6: Comparing average path delay in the output topology after algorithm is run when a link is removed from the network as a result of rain fade in the first use case topology.

The result of rain fade causing a link to be effectively removed from the topology is shown in Fig. 6.6 and 6.7. The output topologies calculated by the OptiCUT algorithm shown in pink (in the figures) has a lower minimum delay and a lower maximum delay than the link removed topologies. The link removed topologies have more outliers than the out-

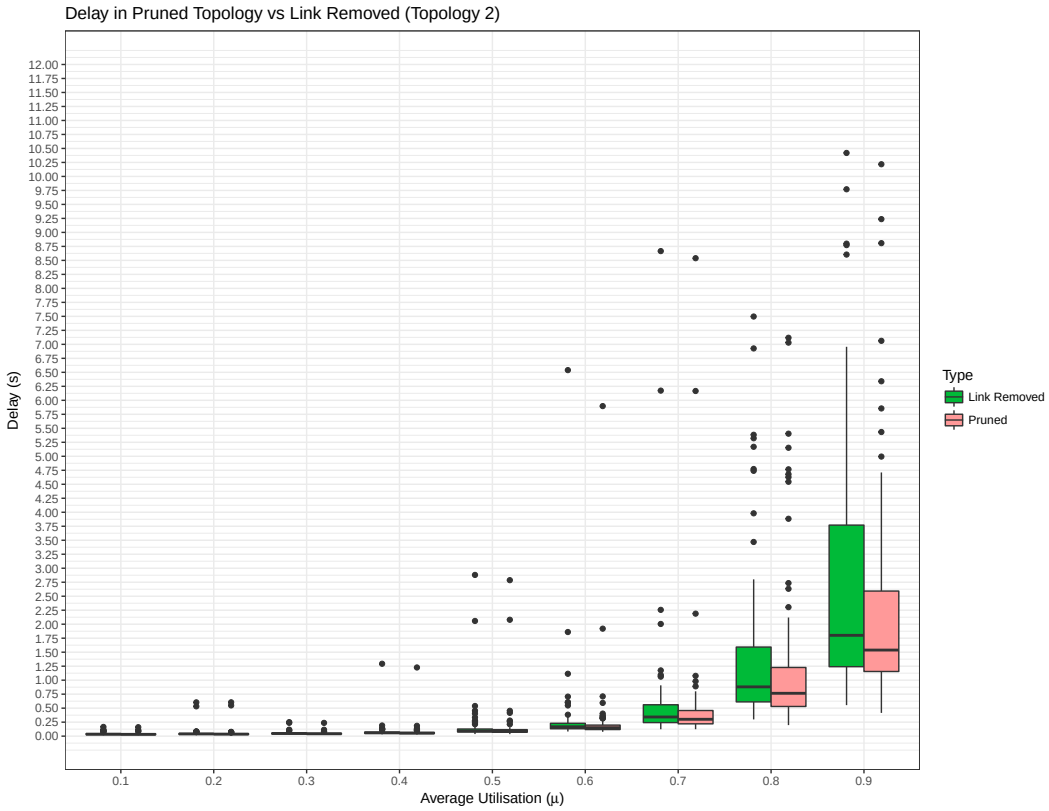

Figure 6.7: Comparing average path delay in the output topology after algorithm is run when a link is removed from the network as a result of rain fade in the second use case topology.

put topologies. In the figures for both use case topologies (shown in Fig. 6.6), the upper quartile, median and lower quartile values for the delays of the utilisations increases when the topology is recomputed. At mean utilisation values above 0.7, the spread of the data increases (more values below the lower quartile and more above the upper quartile). This follows the same trend of delay becoming unpredictable as utilisation increases beyond 0.7. The observed trend in comparing the output and link removed topologies confirms that the output topology is minimal and any subsequent recomputations that occur as a result of rain fade will yield

higher delay. Across each average utilisation value, the topology output when the algorithm is run to recompute the topology has a higher median delay than the original topology calculated. This confirms that the original topology output by the algorithm is minimal in terms of delay. The recomputed topology is minimal in regards to average delay across the paths given the links that can be used now.

The traffic that was sent through the link that has gone down has now been redirected through another link in the topology. In turn, average delay increases as there is less overall capacity in the topology to carry the same amount of traffic.

6.4.2 Path Diversity

Path diversity is a measure of how many possible paths exist in a network to get to a destination node from a given source node. It is commonly used as a measure of redundancy in a network. This was used to evaluate the algorithm, by determining how many possible paths to select from for each source-destination router to build the minimum cost topology.

Every possible path for a source-destination pair of routers is computed during the *compute_all_simple_paths* step of the algorithms detailed in Chapter 5. The path used for a particular source-destination pair of routers is chosen from this set of paths and subsequently the links that make up the paths make up the output topology.

The path selected for each source-destination pair of routers must meet capacity constraints as required by the constraints denoted by Eq. 3.2 and 3.3. In both algorithms (both end up with the same optimal topology) detailed in Chapter 5, the algorithm picks the path with minimal delay if and only if it also meets capacity constraints. As mean utilisation along the links increase, traffic demand is higher meaning there will be less residual capacity on the links to meet new traffic demand.

There was a difference in the simulations of obtaining a minimum cost

topology with the two use case topologies. With the first use case topology (shown in Fig. 6.8, there is a decrease in path diversity as utilisation increases as observed by the decreasing upper and lower quartile, and median values in both the output and link removed topology (after recomputation). As utilisation increases, not all the paths found during the *compute_all_simple_paths* step will be able to meet the capacity constraints. In the output topology, at utilisation values below 0.7, there is a larger spread in the number of paths found. The spread in the link removed topology stays approximately constant through all the utilisation values. This expected trend of the number of paths found decreasing with mean utilisation was observed in the first use case topology but not the second. In the second use case topology, the path diversity stayed constant when calculating the output topology. The same median number for path diversity was observed in the second use case topology in Fig 6.9 despite mean utilisation changing. This is due to a lower proportion of links that can be pruned off in the second use case topology (shown as the active number of ODUs in Fig. 6.11 and 6.12) as the average degree of connectivity is lower. In the link removed topology of the second use case topology, there is a much larger spread in the path diversity values but the median value stays constant.

The results show that path diversity for the link removed topology (after a recomputation has occurred) will always be lower than the output topology. Path diversity is based on the average of total number of paths that can be used in the whole topology. If a link can no longer be used, path diversity will decrease as there will be a lower number of total permutations of paths.

Comparing path diversity to the maximum tolerable delay is another way of evaluating the algorithm using this metric. Maximum tolerable delay is another input variable that is used to represent QoS requirements set by the operator. In the first use case topology with a mean utilisation value of 0.5, the average path diversity given maximum tolerable delay

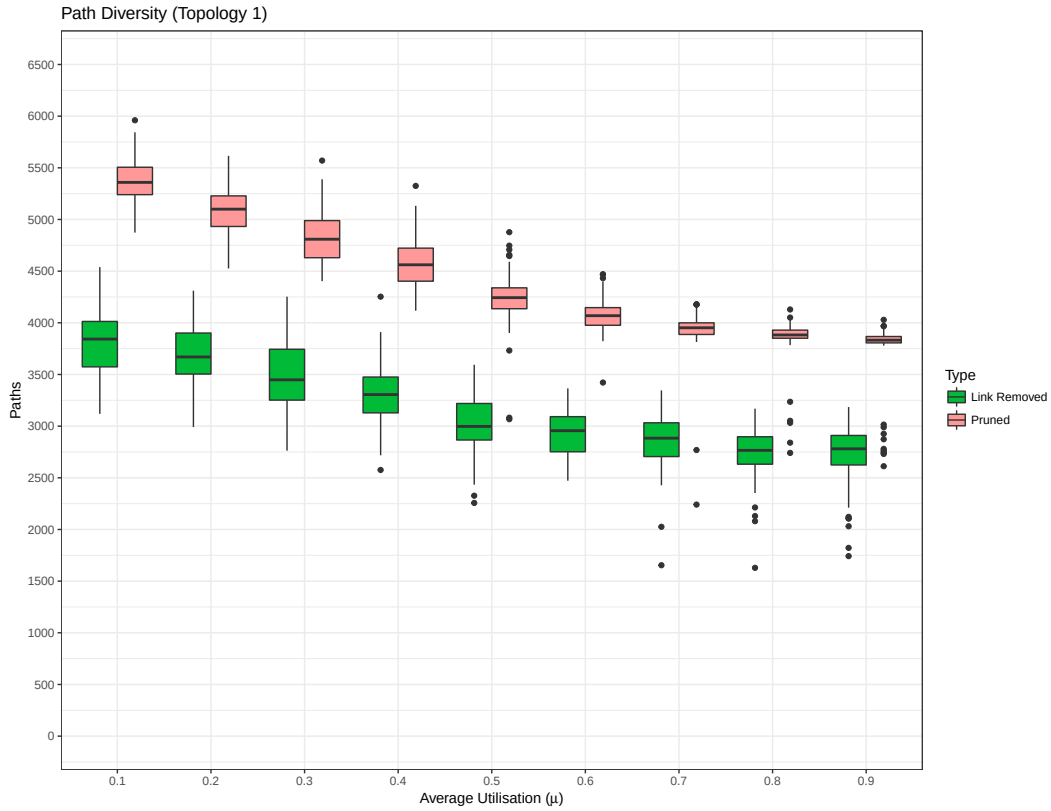

Figure 6.8: The average number of paths that meet capacity constraints in the first use case topology vs. mean utilisation.

values are seen in Fig. 6.10. Path diversity decreases as the maximum tolerable delay decreases. There are no possible paths once maximum tolerable delay drops below 0.0125 seconds. The simulations start with maximum tolerable delay at the highest value of 0.100 seconds as there is no observed change in path diversity beyond that point given the mean utilisation value of 0.5.

6.4.3 Power Consumption

Minimising power consumption is the third objective of this thesis. The current convention in operating microwave backhaul networks is to leave

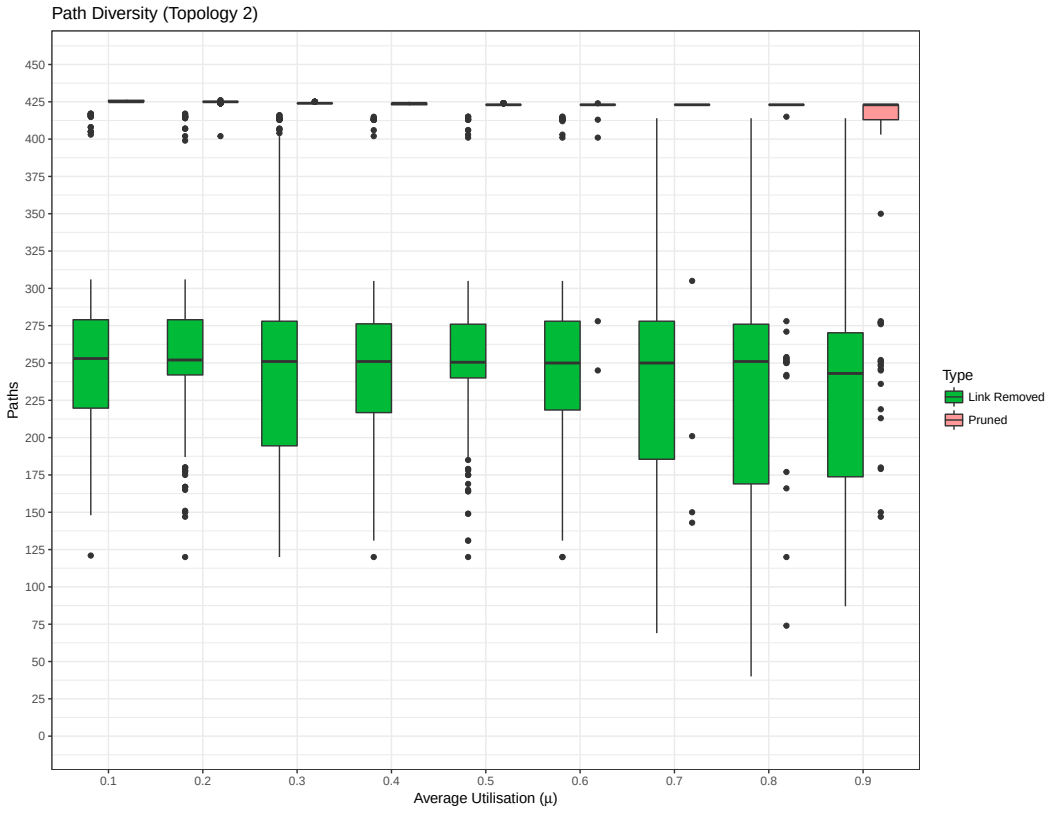

Figure 6.9: The average number of paths that meet capacity constraints in the second use case topology vs. mean utilisation.

all links enabled regardless of traffic levels as described in Chapter 2. The algorithm detailed in Chapter 5 simplifies a microwave backhaul topology by pruning off the links that do not need to be used in order to minimise delay. Unused links are pruned off in the resulting topology, so power can be saved if the ODUs of that link are powered off. This was not part of the thesis but was in previous work done. The algorithm determines which links can be shut off to reduce power consumption.

Figures 6.11 and 6.12 shows the power consumption of the two use case topologies. The brown boxes represent the number of active ODUs in the input topology across the simulations. These topologies have the highest

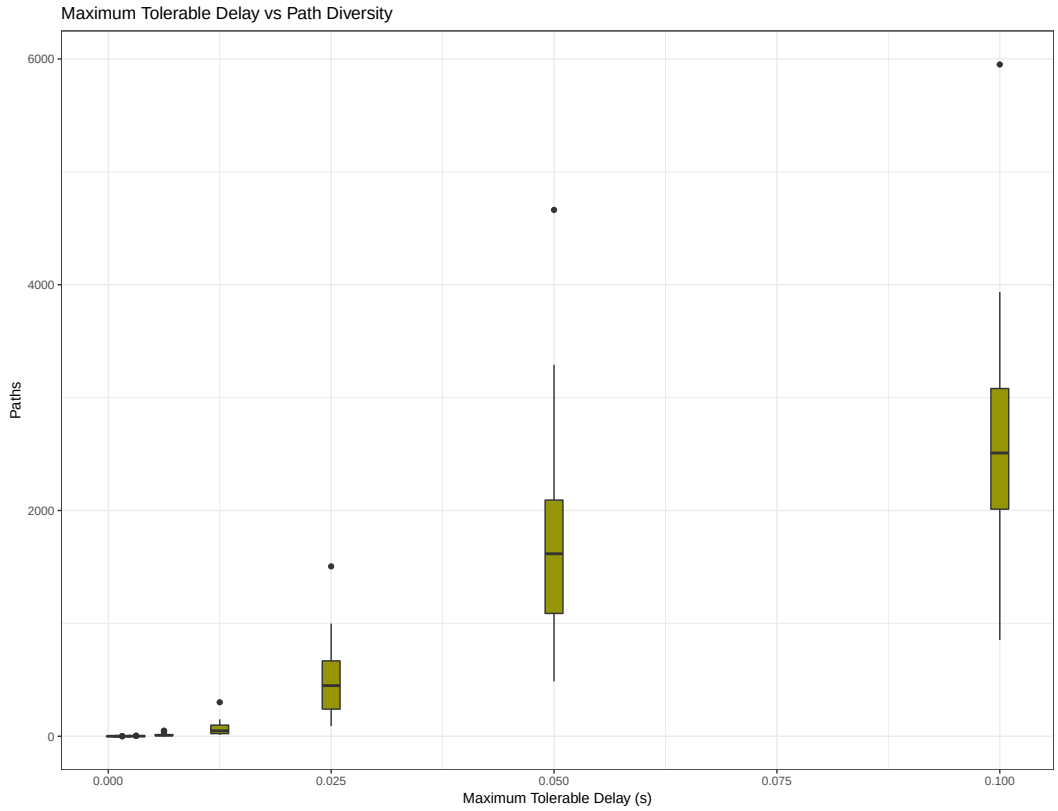

Figure 6.10: The average number of paths that meet capacity constraints vs. maximum tolerable delay.

number of active ODUs and subsequently the highest power consumption. This is the maximum possible power consumption of the topology if everything is left on. The pink boxes are for the initial output topology after the algorithm is run, and the topology is pruned and simplified to yield a topology and paths with lowest delay. The green boxes represent the topology that is recomputed when the algorithm is rerun as a result of an input variable changing. The initial output topology has the lowest number of active ODUs out of the three in Figures 6.11 and 6.12 when using the maximum, minimum, upper and lower quartiles and median. Also, the overall spread in the number of active ODUs stays the constant

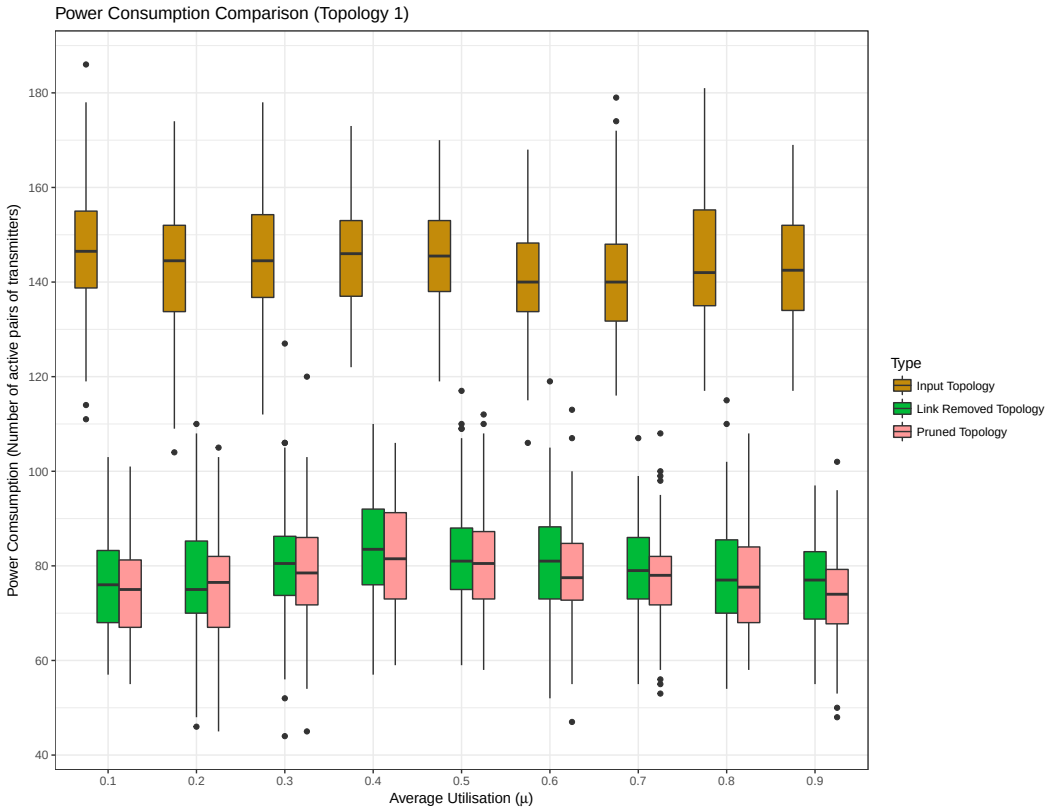

Figure 6.11: The number of active ODUs in the input topologies, the output topology and when a link has been removed when the first use case topology is used as input into the algorithm.

throughout all utilisation values. When a link goes down, the topology obtained from rerunning the algorithm produces a topology with a higher number of active ODUs which in turn increases power consumption.

In a particular simulation with the first use case topology (Fig. 6.2), the pattern of links decreases when the topology is pruned and increases when a link in the pruned topology is removed. When links are pruned off to minimise delay in Fig. 6.13, the number of ODUs decreases. The average number of active ODUs has decreased from 132 to 79. When the link between routers i and l is removed, the number of active ODUs then

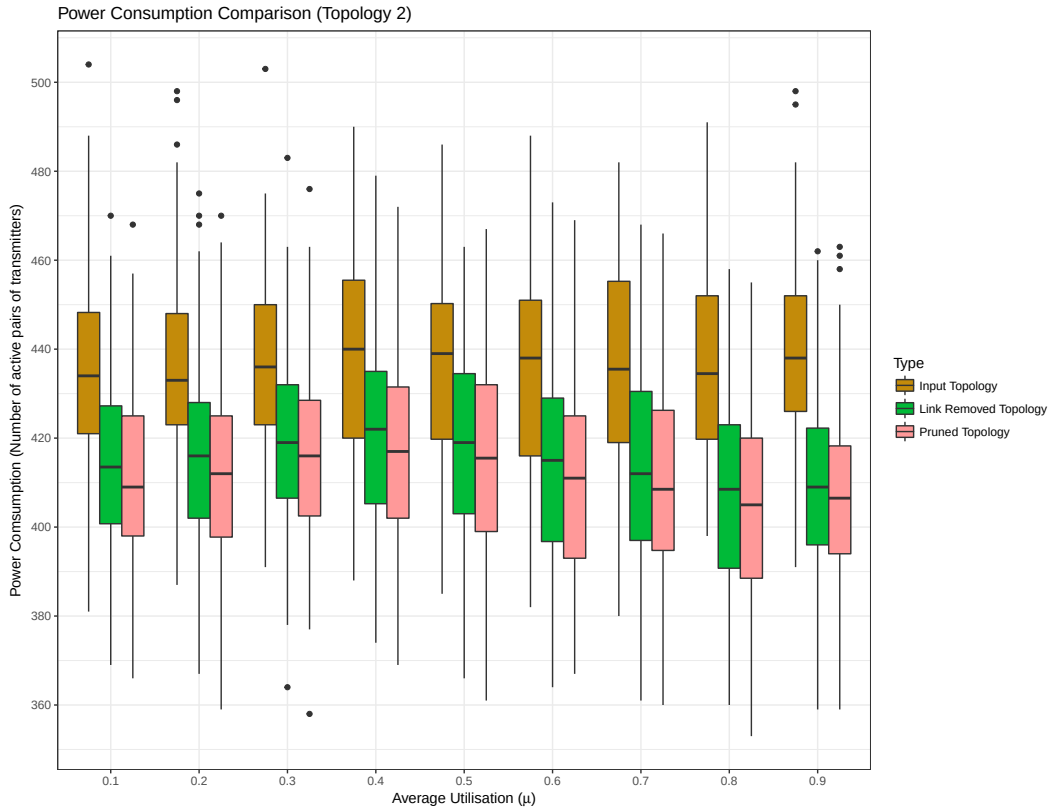

Figure 6.12: The number of active ODUs in the input topologies, the output topology and when a link has been removed when the second use case topology is used as input into the algorithm.

increases to 85. This is due to the link added between routers b and i (shown in Fig. 6.14), having a higher number of active ODUs than the link removed between routers i and l. On average across the simulations, there was an observed 39% decrease in ODUs when the input topology is pruned using the OptiCUT algorithm detailed in Chapter 5.

Unlike delay which changes with mean utilisation, there is no notable change in the number of active links in the topology as utilisation increases in Figures 6.11 and 6.12. There is a slight increase in maximum, minimum, upper/lower quartile and median values in the link removed topology but

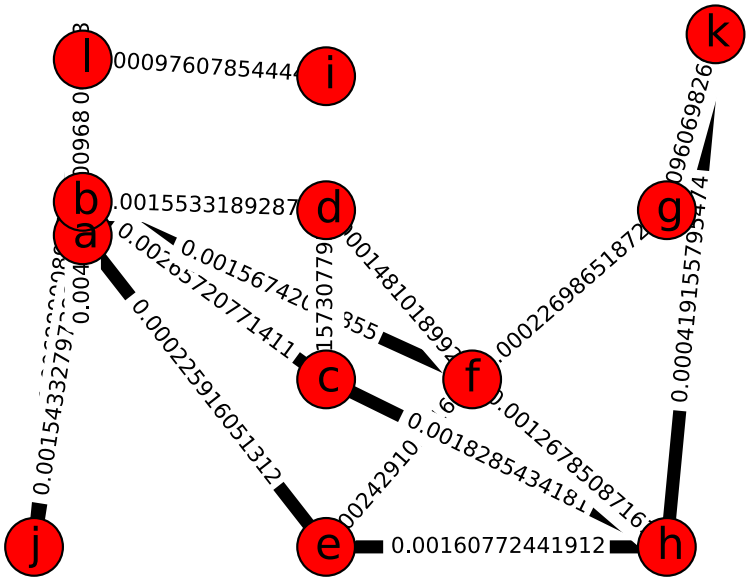

Figure 6.13: A pruned topology after running the algorithm on the first use case topology shown in Fig. 6.2.

the difference is insignificant. The individual ODUs that make up the capacity on a link are not powered off in this thesis. Instead, Figures 6.11 and 6.12 show that links can be turned off in the topology when the algorithm is run on the input topology.

6.4.4 Monetary Cost

Monetary cost is the final metric used to evaluate the algorithm used to simplify the input topologies in order to minimise delay. As described in Section 6.1, a monetary cost is assigned to the links in the topology to transport per Mb of traffic.

The minimum delay topology shown in Fig. 6.13 does not necessarily have the lowest cost. The average cost of the paths in the resulting topology is $10.95. The costs of routing traffic through the topology changes when a topology recalculation is performed. When the link between routers

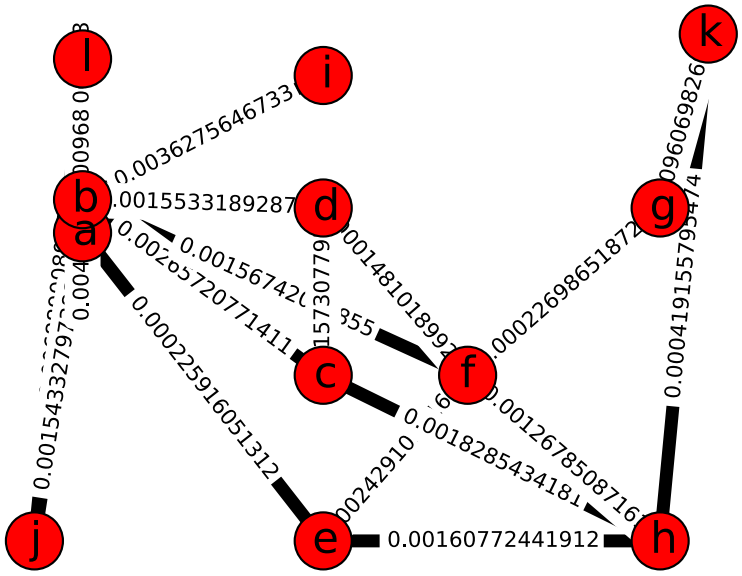

Figure 6.14: The resulting topology after a topology recomputation has occurred after link *b*-*i* is removed from the topology shown in Fig. 6.13.

i and *l* is removed, the average cost decreases to $10.49.

Figures 6.15 and 6.16 show the average costs across the simulations in both use case topologies. There is an increase in cost when μ is between 0.3 and 0.6, with the overall curve appearing as a normal distribution. As traffic is generated using a truncated normal distribution $\sim N(\mu, \sigma^2)$, the observed pattern is expected. The spread in the data is about the same across all utilisation values with the only visible observations being the output topology having a lower median, and quartile values than the link removed topology at all values below 0.8. At utilisation values of 0.8 and above, the link removed topology has lower median, and upper and lower quartile values. This shows that there isn't a significant monetary cost to the operator when using the OptiCUT algorithm. Less emphasis can be placed on the cost of the topologies generated by OptiCUT.

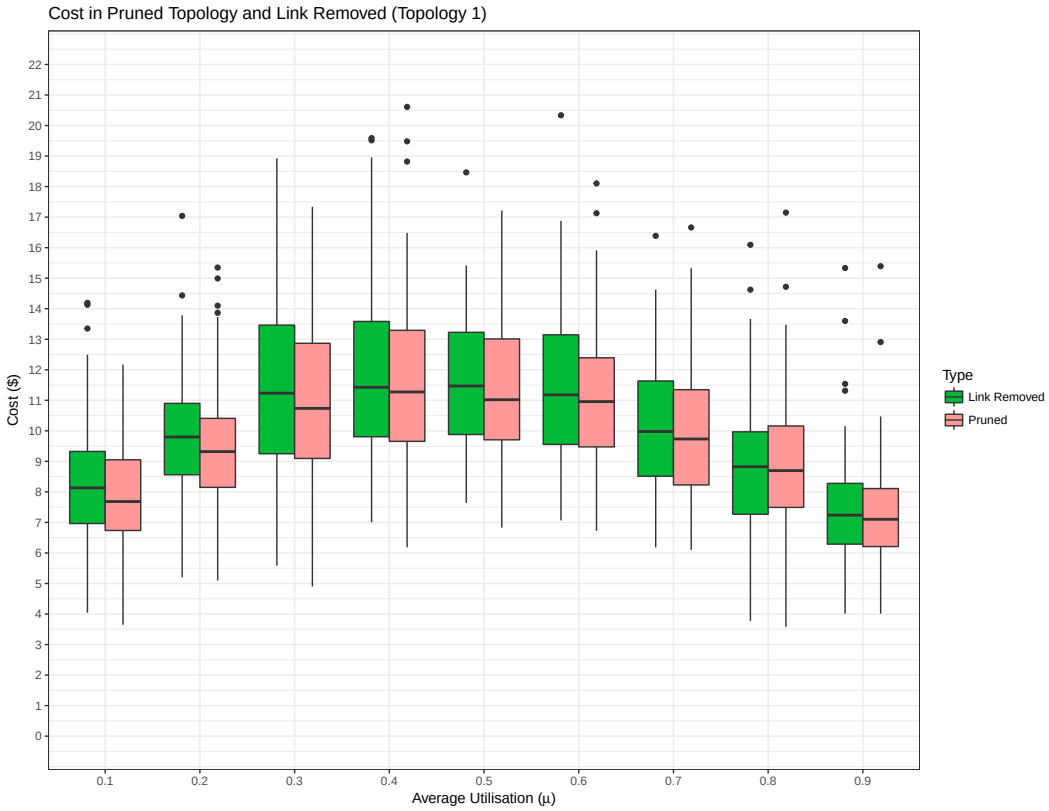

Figure 6.15: Comparing costs vs. mean utilisation across the links in the topology.

.

6.5 Packet Level Simulations

The discrete event packet level simulator described in Section 4.2 was used to validate the topology. This section discuses the results obtained when simulating the pruned topologies obtained from the algorithm.

6.5.1 Simulation Method

The resulting topology calculated by the algorithm in Chapter 5 is used as an input along with the input matrices (of constraints and link utilisations

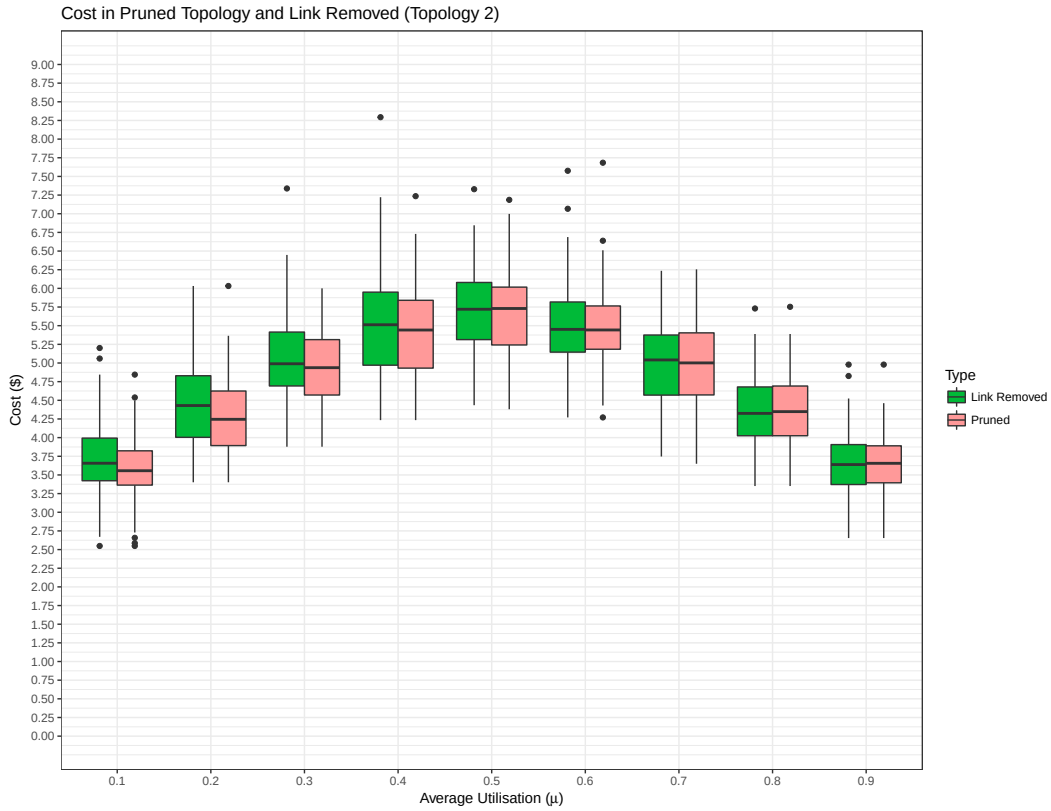

Figure 6.16: Comparing costs vs. mean utilisation across the links in the topology.

.

and etc.) into the discrete event packet level simulator. Packets are then sent through the topology using the simulator. This simulator is used to determine whether the calculated topology is minimal by either removing a link or increasing noise along that link. This requires the existing traffic to be rerouted on an alternative path if capacity is no longer sufficient or if the link is removed. Delay is expected to increase when the topology needs to be recalculated as a result of changes to the link (capacity or link removed) as the topology calculated is minimal with respect to delay. The results from the simulator can be seen in Section 6.5.2.

6.5.2 Simulation Results

Given an input topology as seen in Fig. 6.2 and 6.3, the algorithm is executed to simplify the topology such that capacity and maximum tolerable delay requirements are satisfied and delay for each pair of routers in the network is minimised. The output of the algorithm is a minimum cost topology and contains the shortest path in regards to delay for each source-destination pair of nodes in the network.

The results from the simulator can be seen in Fig. 6.17 and 6.18. The results for the average delay in the simulator follows the same exponential curve as shown in the analytical results (shown in Fig. 6.4 and 6.5) when mean utilisation is perturbed.

The analytical results of the first use case topology (shown in Fig. 6.17), shows that there is little spread in average delay in the topology until utilisation increases above 0.6. Each of the maximum, upper quartile, lower quartile, median and minimum values increase with utilisation though. In the simulations for the first use case topology, the median values increase with utilisation but there is not a clear increase in the upper/lower quartile when comparing sequential utilisation values. The results obtained from the simulator contain more outliers than the results from the model.

The results for the second use case topology (shown in Fig. 6.18) follows the same exponential trend but the ranges are different to the first use case topology. In the analytical results of the second use case topology, the maximum, minimum, lower/upper quartile and median values increase with utilisation. The range also increases with utilisation meaning there is a higher spread in possible delay values as utilisation increases. The same spread is not observed in the simulation results with the spread staying constant and maximum, minimum, lower/upper quartile, and median values as utilisation increases. The exponential increase in delay with utilisation as observed in the simulator confirms that results calculated by the analytical model are correct.

In the first use case topology, the average delay incurred during the

simulations is higher than what is calculated from the model as queuing is applied on the model to simulate a live network. This trend wasn't observed in the second use case topology.

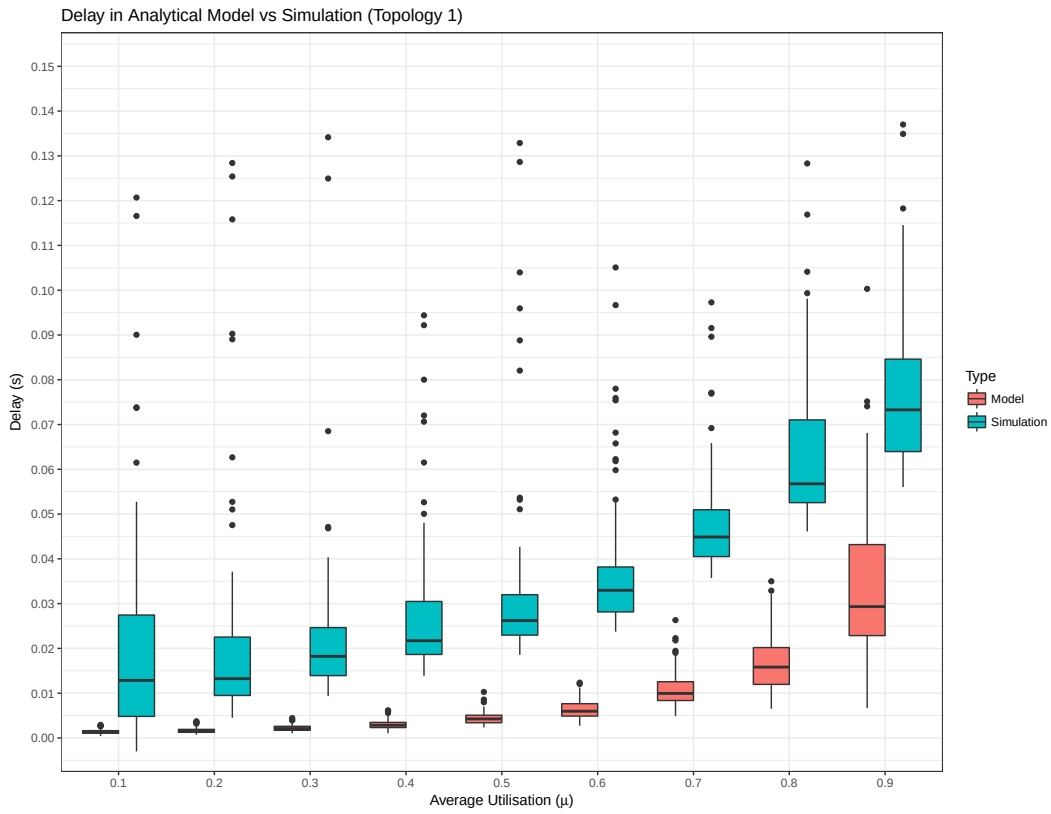

Figure 6.17: Validating delay trend in the pruned topology calculated from the first use case topology using a packet level simulator.

6.6 Summary

In this chapter, the topologies calculated by the OptiCUT algorithm was evaluated using *delay, path diversity, power consumption and monetary cost* as the metrics. Introducing the loss of links and a packet level simulator validated that the topology and paths calculated by OptiCUT was optimal

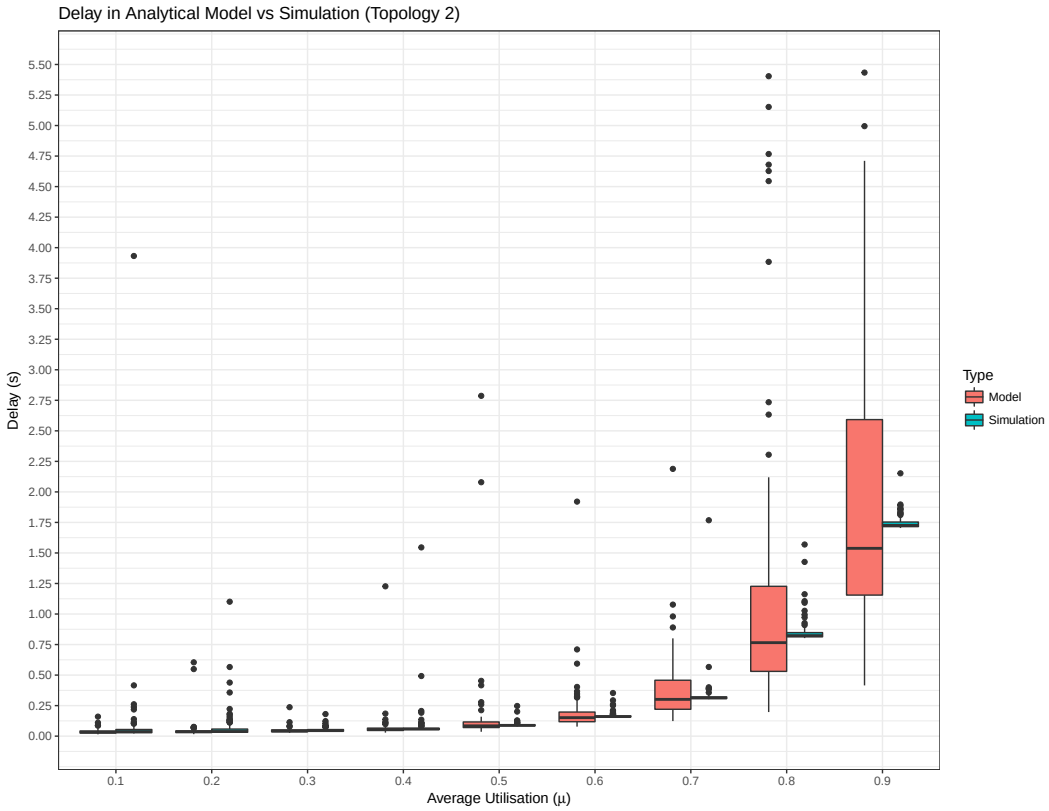

Figure 6.18: Validating delay trend in the pruned topology calculated from the first use case topology using a packet level simulator.

in regards to delay. There was no clear trend in monetary cost when perturbing mean utilisation from the evaluation tests but it was shown that OptiCUT improves energy consumption by pruning off unnecessary links in the topology.

Chapter 7

Conclusions

The backhaul is an important part of a network that link the access back to the core network. Microwave radio make up over 50% of all backhaul networks in the world. They are widely used due to their low cost and flexibility of deployment where fixed wired backhaul is not available. Unfortunately, the current convention in all backhaul networks is to statically provision capacity to paths, which leads to QoS issues and high energy consumption.

A network model to represent a microwave backhaul network was created to accommodate algorithms that simplify, or prune a topology. As part of the network model, physical coordinates of the routers were used inside of a wireless model to simulate the conditions of the channel to calculate connectivity. The model allows metrics (such as delay, power consumption and etc.) to be calculated.

A brute force algorithm was written to prune off links in a topology to reduce energy consumption, ensure capacity constraints and minimise delay within the topology. The execution time of the algorithm was high and improvements were made in the CUT and OptiCUT algorithms that obtain the same optimal topology and paths.

The OptiCUT algorithm was evaluated using delay, path diversity, power consumption and monetary cost as metrics. The output topology and

paths have been proven to be optimal by the introduction of noise to the network. A link in the resulting pruned topology is removed and a topology recalculation occurs. From this, the results show that delay and power consumption increases. A discrete packet level simulator was used to validate the results. The output of the simulations proved the delay trend.

7.1 Future Work

For future work, the algorithms produced in this thesis are to be deployed and run on a live microwave backhaul network. Deploying the work completed for this thesis was not within scope but the centralised attribute of SDN is key to the algorithms working. The previous work with Aviat Networks to obtain readings from the switches will need to be extended to gather all of the information needed for the algorithm to run. Values such as interference in the wireless model and traffic volume is currently generated inside the model. Actual readings from the switches will determine real-world performance of the algorithms providing insights into any improvements needed.

Bibliography

[1] ALCATEL-LUCENT. Microwave Backhaul for LTE and Beyond with the Alcatel-Lucent 9500 Microwave Packet Radio.

[2] ANDERSON, H. R. *Fixed broadband wireless system design*. Wiley, West Sussex, England, New York, 2003.

[3] BANNISTER, J. A., FRATTA, L., AND GERLA, M. Topological design of the wavelength-division optical network. In *INFOCOM '90, Ninth Annual Joint Conference of the IEEE Computer and Communication Societies. The Multiple Facets of Integration. Proceedings, IEEE* (Jun 1990), pp. 1005–1013 vol.3.

[4] BEN-OTHMAN, J., BESSAOUD, K., BUI, A., AND PILARD, L. Self-stabilizing algorithm for efficient topology control in wireless sensor networks. *Journal of Computational Science 4*, 4 (2013), 199 – 208. {PEDISWESA} 2011 and Sc. computing for Cog. Sciences.

[5] BERCOVICH, D., CONTRERAS, L. M., HADDAD, Y., ADAM, A., AND BERNARDOS, C. J. Software-defined wireless transport networks for flexible mobile backhaul in 5g systems. *Mob. Netw. Appl. 20*, 6 (Dec. 2015), 793–801.

[6] BOJIC, D., SASAKI, E., CVIJETIC, N., WANG, T., KUNO, J., LESSMANN, J., SCHMID, S., ISHII, H., AND NAKAMURA, S. Advanced wireless and optical technologies for small-cell mobile backhaul with

dynamic software-defined management. *IEEE Communications Magazine 51*, 9 (September 2013), 86–93.

[7] CHIA, S., GASPARRONI, M., AND BRICK, P. The next challenge for cellular networks: backhaul. *IEEE Microwave Magazine 10*, 5 (August 2009), 54–66.

[8] CHIARAVIGLIO, L., MELLIA, M., AND NERI, F. Reducing power consumption in backbone networks. In *2009 IEEE International Conference on Communications* (June 2009), pp. 1–6.

[9] CHOI, J. S. A hybrid topology discovery protocol for mobile backhaul. In *Proceedings of the 16th Communications & Networking Symposium* (San Diego, CA, USA, 2013), CNS '13, Society for Computer Simulation International, pp. 15:1–15:4.

[10] CHUNDURY, R. Mobile broadband backhaul: Addressing the challenge, mar 2008.

[11] CISCO. Cisco visual networking index: Global mobile data traffic forecast update, 2014 - 2019. *Cisco Public.* (2015).

[12] CISCO. Cisco visual networking index: Global mobile data traffic forecast update, 2015 - 2020 white paper. *Cisco Public.* (2016).

[13] COLBOURN, C. J. *The Combinatorics of Network Reliability*. Oxford University Press, Inc., New York, NY, USA, 1987.

[14] COLDREY, M., ENGSTROM, U., HELMERSSON, K. W., HASHEMI, M., MANHOLM, L., AND WALLENTI, P. Wireless backhaul in future heterogeneous networks, October 2014.

[15] CROVELLA, M., AND KRISHNAMURTHY, B. *Internet Measurement: Infrastructure, Traffic and Applications*. John Wiley & Sons, Inc., New York, NY, USA, 2006.

[16] CUOMO, F., CIANFRANI, A., POLVERINI, M., AND MANGIONE, D. Network pruning for energy saving in the internet. *Computer Networks 56*, 10 (2012), 2355 – 2367.

[17] DOROGOVTSEV, S. N., AND MENDES, J. F. F. *Evolution of Networks: From Biological Nets to the Internet and WWW (Physics)*. Oxford University Press, Inc., New York, NY, USA, 2003.

[18] ERICSSON. Microwave towards 2020, September 2015.

[19] FENCL, T., BURGET, P., AND BILEK, J.

[20] FENCL, T., BURGET, P., AND BILEK, J. Network topology design. *Control Engineering Practice 19*, 11 (2011), 1287 – 1296.

[21] FERNNDEZ-FERNNDEZ, A., CERVELL-PASTOR, C., AND OCHOA-ADAY, L. Improved energy-aware routing algorithm in software-defined networks. In *2016 IEEE 41st Conference on Local Computer Networks (LCN)* (Nov 2016), pp. 196–199.

[22] FU, Y., BI, J., GAO, K., CHEN, Z., WU, J., AND HAO, B. Orion: A hybrid hierarchical control plane of software-defined networking for large-scale networks. In *2014 IEEE 22nd International Conference on Network Protocols* (Oct 2014), pp. 569–576.

[23] GAVISH, B. Topological design of computer communication networks the overall design problem. *European Journal of Operational Research 58*, 2 (1992), 149 – 172.

[24] GE, X., CHENG, H., GUIZANI, M., AND HAN, T. 5g wireless backhaul networks: challenges and research advances. *IEEE Network 28*, 6 (Nov 2014), 6–11.

[25] GERLA, M., AND KLEINROCK, L. On the topological design of distributed computer networks. *IEEE Transactions on Communications 25*, 1 (Jan 1977), 48–60.

[26] HSU, C.-Y., WU, J.-L. C., WANG, S.-T., AND HONG, C.-Y. Survivable and delay-guaranteed backbone wireless mesh network design. *Journal of Parallel and Distributed Computing 68*, 3 (2008), 306 – 320.

[27] JAFARI, A. H., LÓPEZ-PÉREZ, D., SONG, H., CLAUSSEN, H., HO, L., AND ZHANG, J. Small cell backhaul: challenges and prospective solutions. *EURASIP Journal on Wireless Communications and Networking 2015*, 1 (2015), 1–18.

[28] KAMIYAMA, N. Efficiently constructing candidate set for network topology design. In *2009 IEEE International Conference on Communications* (June 2009), pp. 1–6.

[29] KREUTZ, D., RAMOS, F. M. V., VERSSIMO, P. E., ROTHENBERG, C. E., AZODOLMOLKY, S., AND UHLIG, S. Software-defined networking: A comprehensive survey. *Proceedings of the IEEE 103*, 1 (Jan 2015), 14–76.

[30] KUMAR, R., AND BANERJEE, N. Multiobjective network topology design. *Applied Soft Computing 11*, 8 (2011), 5120 – 5128.

[31] KUO, F. C., ZDARSKY, F. A., LESSMANN, J., AND SCHMID, S. Cost-efficient wireless mobile backhaul topologies: An analytical study. In *2010 IEEE Global Telecommunications Conference GLOBECOM 2010* (Dec 2010), pp. 1–5.

[32] LEHPAMER, H. *Microwave Transmission Networks: Planning, Design, and Deployment*. McGraw-Hill Education (India) Pvt Limited, 2010.

[33] LI, L. E., MAO, Z. M., AND REXFORD, J. Toward software-defined cellular networks. In *2012 European Workshop on Software Defined Networking* (Oct 2012), pp. 7–12.

[34] LI, Y., CAI, A., QIAO, G., SHI, L., BOSE, S. K., AND SHEN, G. Multiobjective topology planning for microwave-based wireless backhaul networks. *IEEE Access 4* (2016), 5742–5754.

[35] LI, Y., QIAO, G., CAI, A., SHI, L., ZHAO, H., AND SHEN, G. Microwave backhaul topology planning for wireless access networks. In *2014 16th International Conference on Transparent Optical Networks (ICTON)* (July 2014), pp. 1–4.

[36] LIN, G., SOH, S., CHIN, K.-W., AND LAZARESCU, M. Efficient heuristics for energy-aware routing in networks with bundled links. *Computer Networks 57*, 8 (2013), 1774 – 1788.

[37] LIN, G., SOH, S., CHIN, K.-W., AND LAZARESCU, M. Energy aware two disjoint paths routing. *Journal of Network and Computer Applications 43* (2014), 27 – 41.

[38] LIN, G., SOH, S., LAZARESCU, M., AND CHIN, K. W. Power-aware routing in networks with delay and link utilization constraints. In *37th Annual IEEE Conference on Local Computer Networks* (Oct 2012), pp. 272–275.

[39] LITTLE, S. Is microwave backhaul up to the 4g task? *IEEE Microwave Magazine 10*, 5 (August 2009), 67–74.

[40] LU, H., ARORA, N., ZHANG, H., LUMEZANU, C., RHEE, J., AND JIANG, G. Hybnet: Network manager for a hybrid network infrastructure. In *Proceedings of the Industrial Track of the 13th ACM/IFIP/USENIX International Middleware Conference* (New York, NY, USA, 2013), Middleware Industry '13, ACM, pp. 6:1–6:6.

[41] MAHLOO, M., MONTI, P., CHEN, J., AND WOSINSKA, L. Cost modeling of backhaul for mobile networks. In *2014 IEEE International Conference on Communications Workshops (ICC)* (June 2014), pp. 397–402.

[42] MCKEOWN, N., ANDERSON, T., BALAKRISHNAN, H., PARULKAR, G., PETERSON, L., REXFORD, J., SHENKER, S., AND TURNER, J. Openflow: Enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev. 38*, 2 (Mar. 2008), 69–74.

[43] MEHBOOB, U., QADIR, J., ALI, S., AND VASILAKOS, A. Genetic algorithms in wireless networking: techniques, applications, and issues. *Soft Computing 20*, 6 (2016), 2467–2501.

[44] NADIV, R., AND NAVEH, T. Wireless backhaul topologies: Analyzing backhaul topology strategies. *Ceragon White Paper* (Aug 2010), 1–15.

[45] NARLIKAR, G., WILFONG, G., AND ZHANG, L. Designing multihop wireless backhaul networks with delay guarantees. *Wireless Networks 16*, 1 (2010), 237–254.

[46] NEPOMUCENO, N. *Network Optimization for Wireless Microwave Backhaul*. 2010.

[47] NETWORKS, A. Optimum network visibility with aviat networks services & provision, 1 2013.

[48] NETWORKS, A. *CSR1000 L1LA Efficiency and Thoughput for ETSI RAC Applications in S/W 2.4*. 2015.

[49] NEWMAN, M. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[50] (ONF), O. N. F. Wireless Transport SDN Proof of Concept White Paper.

[51] PAKZAD, F., PORTMANN, M., TAN, W. L., AND INDULSKA, J. Efficient topology discovery in software defined networks. In *2014 8th International Conference on Signal Processing and Communication Systems (ICSPCS)* (Dec 2014), pp. 1–8.

[52] PENTIKOUSIS, K., WANG, Y., AND HU, W. Mobileflow: Toward software-defined mobile networks. *IEEE Communications Magazine 51*, 7 (July 2013), 44–53.

[53] PIERRE, S., HYPPOLITE, M.-A., BOURJOLLY, J.-M., AND DIOUME, O. Topological design of computer communication networks using simulated annealing. *Engineering Applications of Artificial Intelligence 8*, 1 (1995), 61 – 69.

[54] RAMANATHAN, R., AND ROSALES-HAIN, R. Topology control of multihop wireless networks using transmit power adjustment. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE* (2000), vol. 2, pp. 404–413 vol.2.

[55] ROSEN, K. H. *Discrete Mathematics and Its Applications*, 5th ed. McGraw-Hill Higher Education, 2002.

[56] SANTOS, R., AND KASSLER, A. A sdn controller architecture for small cell wireless backhaul using a lte control channel. In *2016 IEEE 17th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)* (June 2016), pp. 1–3.

[57] SCHLINKER, B., MYSORE, R. N., SMITH, S., MOGUL, J. C., VAHDAT, A., YU, M., KATZ-BASSETT, E., AND RUBIN, M. Condor: Better topologies through declarative design. *SIGCOMM Comput. Commun. Rev. 45*, 4 (Aug. 2015), 449–463.

[58] SHEN, Y., CAI, Y., AND XU, X. A shortest-path-based topology control algorithm in wireless multihop networks. *SIGCOMM Comput. Commun. Rev. 37*, 5 (Oct. 2007), 29–38.

[59] SHERWOOD, R., GIBB, G., YAP, K.-K., APPENZELLER, G., CASADO, M., MCKEOWN, N., AND PARULKAR, G. Can the production network be the testbed? In *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation* (Berkeley, CA, USA, 2010), OSDI'10, USENIX Association, pp. 365–378.

[60] SRIVATSA, S., AND SESHAIAH, P. On the topological design of a computer network. *Computer Networks and ISDN Systems 27*, 4 (1995), 567 – 569.

[61] SZLACHCIC, E. Fault tolerant topological design for computer networks. In *2006 International Conference on Dependability of Computer Systems* (May 2006), pp. 150–159.

[62] SZLACHCIC, E., AND MLYNEK, J. Efficiency analysis in communication networks topology design. In *2009 Fourth International Conference on Dependability of Computer Systems* (June 2009), pp. 184–191.

[63] TALEB, T. Toward carrier cloud: Potential, challenges, and solutions. *IEEE Wireless Communications 21*, 3 (June 2014), 80–91.

[64] TIPMONGKOLSILP, O., ZAGHLOUL, S., AND JUKAN, A. The evolution of cellular backhaul technologies: Current issues and future trends. *IEEE Communications Surveys Tutorials 13*, 1 (First 2011), 97–113.

[65] TJELTA, T., JENSEN, T., KARASEN, A. G., MILLSTEIN, G., NGUYEN, C. V., ZOUGANELI, E., AARSTAD, E., ASIF, S. Z., AND SUKUR, H. An evaluation of future mobile networks backhaul options. In *Wireless and Mobile Communications, 2009. ICWMC '09. Fifth International Conference on* (Aug 2009), pp. 146–151.

[66] VAN STEEN, M. *Graph Theory and Complex Networks: An Introduction*. On Demand Publishing, LLC-Create Space, 2010.

[67] WETTE, P., AND KARL, H. On the quality of selfish virtual topology reconfiguration in ip-over-wdm networks. In *2013 19th IEEE Workshop on Local Metropolitan Area Networks (LANMAN)* (April 2013), pp. 1–6.

[68] WIGHTMAN, P. M., AND LABRADOR, M. A. A3: A topology construction algorithm for wireless sensor networks. In *IEEE GLOBECOM 2008 - 2008 IEEE Global Telecommunications Conference* (Nov 2008), pp. 1–6.

[69] YONEZU, H., KIKUTA, K., ISHII, D., OKAMOTO, S., OKI, E., AND YAMANAKA, N. Qos aware energy optimal network topology design and dynamic link power management. In *36th European Conference and Exhibition on Optical Communication* (Sept 2010), pp. 1–3.