Design and Implementation of a Deterministic Multipath Solution for Optimizing Backhaul Performance in Geographically Distributed 5G Campus Networks

Nicolas Zunker

n.zunker@campus.tu-berlin.de

January 9, 2024

MASTER'S THESIS

Telecommunication Systems Institute

Technische Universität Berlin

Examiner 1: Prof. Dr. Thomas Magedanz

Examiner 2: Prof. Dr. Axel Küpper

Advisor: Dr. Ing. Marius Corici

Declaration

I hereby declare that I have created the present work independently and by my own without illicit assistance and only utilizing the listed sources and tools.

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschliesslich unter Verwendung der aufgefährten Quellen und Hilfsmittel angefertigt habe.

Die selbständige und eigenständige Anfertigung versichert an Eides statt:

Berlin, January 9, 2024

Nicolas Zunker

Acknowledgment

I would like to thank my unofficial advisor Hauke Buhr for his help, advice and support. I also thank my advisor Marius for his insights. Both they and Thomas provided excellent constructive feedback to me, for which I am grateful. I am also indebted to the rest team at Fraunhofer FOKUS for all of their help, especially Hemant Zope, Christian Scheich, and Aleksandar Yonchev. Lastly I would like to thank Prof. Dr.-Ing. habil. Falko Dressler and Prof. Dr. Zubow from whom I learned a lot, and whose courses I enjoyed immensely during my studies.

NZ

Abstract

The dawn of 5G brings with it an increased focus on bespoke quality of service provisions for different applications. Backhaul that respects the quality of service requirements of the applications running on-site is a difficult task, especially for remote or emergency deployments. The new space race in Lower Earth Orbit (LEO) has created a new possibility for backhaul over LEO satellite constellations. For industrial 5G sites, often located outside of cities and without access to fibre, this transforms the backhaul challenge into a multipath question.

This thesis presents an investigation into the use of multiple paths to provide deterministic backhaul in a geographically distributed 5G Campus deployment. In the course of this document a design and an implementation are presented that take a 5G flow's requirements and use knowledge collected about the available backhaul paths to calculate which path(s) to use, via a binary optimization equation. The proposed solution comprises both a control plane and a data plane component, as well as a traffic shaper. In addition, a deployment configuration designed specifically for use in geographically distributed Campus 5G Networks is described.

This thesis' solution is investigated with respect to its ability to deterministically provide various network characteristics to traffic flows which require them. This investigation is performed on a custom testbed which emulates multiple links. The results do not present conclusive evidence for the solution's ability to provide determinism, but do show promise. The latency-based path switching is 80% optimal, and the implementation is able to reduce the experienced delay. It fails to reduce jitter for jitter-sensitive flows, and needs to compromise throughput in order to provide reduced packet loss. However it is able to provide lower packet loss than even the Optimal Single Path Oracle, via its use of path switching and packet replication.

Zusammenfassung

Mit der Verbreitung von 5G rückt die von speziellen Applikationen benötigte Servicequalität immer mehr in den Fokus. Insbesondere abgelegene oder für einen Notfall errichtete 5G Netze haben Probleme, einen geeigneten Backhaul bereitzustellen, der die benötigten Netzwerkeigenschaften der kritischen on-site Applikationen unterstützt. Durch die neue, stetig wachsende Verfügbarkeit der Verbindungen über LEO (Low Earth Orbit) Satellitenkonstellationen wird jetzt das traditionelle Problem vom Backhaul eine Multipath Frage.

Diese Arbeit implementiert einen Ansatz der verschiedene ausgehende Pfade benutzt und schlägt eine Lösung für die 5G Campus Umgebung vor, die für 5G Flows einen deterministischen Backhaul bereitstellt. Hierzu wird eine Architektur entworfen, die über ausgehende Pfade Daten sammelt, und anhand einer binären Optimierungsgleichung Entscheidungen trifft. Zusätzlich bietet dieser Lösungsansatz geteilte Steuerungs und Datenebenen, sowie einen Traffic Shaper, und stellt eine Konfiguration spezifisch für 5G Campus Netze vor.

In einem Testbed, dass Multipath Szenarien emuliert, wird diese Lösung, der sogenannte "WAN Connector", auf ihre deterministischen Eigenschaften untersucht. Die Ergebnisse weisen leider nach dass die Lösung keinen kompletten Determinismus anbieten kann. In der Frage der Latenz agiert die hier untersuchte und vorgeschlagene Methodik 80% optimal, und schafft es die Latenz insgesamt zu verringern. Den Jitter zu reduzieren gelingt aber nicht, und auch der Durchsatz von Flows, die für Paketverluste und Zuverlässigkeit optimiert werden, wird extrem reduziert. Dennoch war es möglich, durch die Nutzung von Pfad-wechsel und Replikation, die Paketverluste stärker zu verringern als es mit einen Orakel der Einzelne-Pfade auswählen kann möglich gewesen wäre.

Table of Contents

List of Figures

List of Listings

Chapter 1

Introduction

1.1 Motivation and Problem Description

One of the aims for the fifth generation of mobile networks (5G) and it's successors is a greater diversification of the classes of service they offer. As the use cases for these networks evolve, there is a greater need for quality of service (QoS) tailored to each use case. For example, Industrial Internet of Things (IIoT) applications have stringent requirements on latency, jitter, and reliability. Supporting these kinds of classes of service can be a challenge for mobile network operators (MNOs) and will require novel approaches to familiar problems, such as backhaul.

As there are more heterogeneous edge deployments and more campus networks, backhaul becomes more challenging, since many sites may not have access to optical fibre, and may be forced into using other solutions such as satellite links, mmWave backhaul, or the connections of their Internet Service Provider (ISP). Providing the kind of deterministic quality of service that these sites may require can be a difficult challenge.

Particularly with the rise of satellite backhaul options, fuelled by the new space race, remote

deployments may choose to integrate satellite backhaul because fibre is infeasible in their location or its roll-out is too slow. When these connections are added to the site of a deployment it is usually in addition to a pre-existing one. Thus network operators could choose to utilize both the new and the old backhaul connection at the same time, in order to utilize the different qualities of the backhaul links. This bears the question whether multipathing could be used to provide deterministic backhaul by intelligently selecting on which links to forward packets. This approach bears similarity to multihoming as well as to multi-path routing in Wireless Sensor Networks (WSNs), and can take inspiration from the substantial body of research in these fields which already exists. That research has demonstrated that using multiple links or paths simultaneously can improve QoS [6, 7, 17, 19, 21, 36], and prompts the question whether something similar could be done to address backhaul in 5G campus networks.

1.2 Target

The goal of this thesis is to design and provide an implementation of a Wide Area Network connector (WAN connector), that can be placed at the ingress and egress point of two locations. It should then utilize multipathing in order to provide deterministic backhaul between the two sites. The performance of this approach will then be quantitatively analyzed in experiments.

Looking at Figure 1.1 we can see how this is envisioned to work. WAN connectors are deployed both in the geographically distributed 5G campus network, which has more than one egress link, and in the core. Then, using the multiple outgoing links, they backhaul traffic to the other site, while respecting the traffic's QoS requirements. This can be especially beneficial for critical applications (e.g. at industrial sites), which are in locations that do not have access to optical fibre for backhaul.

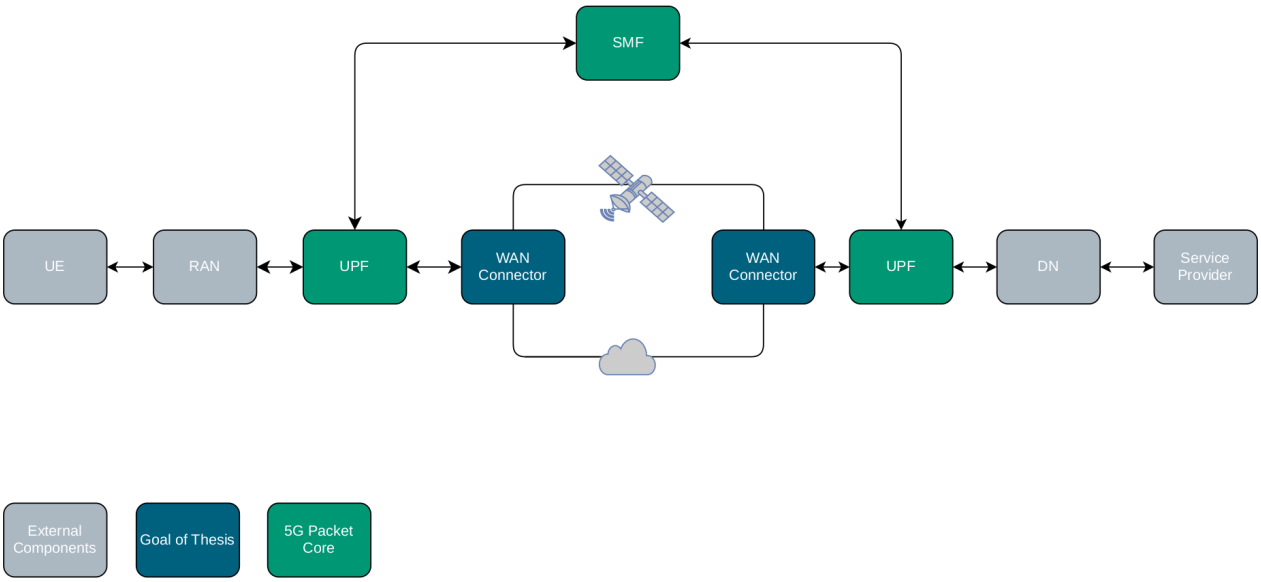


Figure 1.1: 5G Deployment with 2 UPFs

For a 5G deployment the proposed WAN connector could be deployed in between two User Plane Functions (UPF's) , in order to provide deterministic backhaul. The architecture for such a deployment is shown in Figure 1.1. Further, the on-site UPF is not a strict requirement; it is also feasible to connect the RAN directly to the WAN Connector.

1.3 Scope

While a high level design will be presented, and a first implementation will be tested, this implementation will not be industry grade and should not be judged next to commercial packet processing products. Furthermore, an explicit specification for the format of communication with the 5G Core's control plane is largely beyond the scope of this thesis. Although an interface to the 5G Core's Session Management Function (SMF) will be assumed, it is not specified nor implemented. By some means the flows and their requirements must be communicated to the WAN Connector over the control plane, however an exact description of the data formats used in this communication will not occur. The user plane interface to

the WAN Connector is not described either. However, the WAN Connector may be able to act in a fashion which is opaque to the user plane component with which it is interacting (RAN or UPF). The WAN Connector can bridge both the N3 or N9 reference points. The user plane traffic is simply forwarded to the WAN Connector and backhauled to the second WAN Connector, which reintroduces it into the other user plane component. No part of the user plane needs to be aware that the WAN Connector is sitting in the middle.

Lastly, in the process of investigating multipathing for deterministic backhaul, this thesis will not actually test the WAN Connector in a real deployment at a remote site. Additionally, the tests will not instantiate a full 5G Core network for the tests, and will not feature UPF's, a RAN, nor UE's (like in graphic number 1.1).

1.4 Structure

This thesis will follow a 7 chapter structure. This section concludes the introduction chapter, what follows will be one chapter to provide both basic background information as well as to highlight the existing literature which is of relevance to the problem statement. Next will be the requirements, design, and implementation chapters. Next, the evaluation chapter will analyze the WAN Connector's performance. The conclusion chapter will review the relevant findings, the successes and failures of the approach, how to improve on it, and lastly the possibilities for future research in this area.

Chapter 2

Background

To place this thesis in its proper academic and practical context, this chapter will provide background information on relevant topics as well as reviewing some of the literature which bears similarity or served as an inspiration. First, mobile networking and 5G will be covered, then multipath solutions, and finally deterministic networking.

2.1 Mobile Networks and Backhaul

For the sake of brevity, the full history of the development of mobile networks will be hugely condensed and simplified, and only the 5G architecture will be explained in depth.

The defining characteristic of a mobile network is wireless connectivity. This enables the users to move around while remaining connected to the network, which stands in direct opposition to historical computer networks which were stationary and usually featured fixed connections between computers. A user in a mobile network must first detect and then connect to a Base Station, this process includes authentication and authorization. A user may be switched from one Base Station to another depending on the signal strength (this is referred to as a

handover), and this is what enables the user to maintain a connection whilst moving across wide areas. In mobile networks which support IP traffic, the user will normally appear to remain behind a single IP address despite their mobility.

All of the user's data sent to the network goes through the Base Station to which it is currently connected. At first this data was exclusively analog voice data, however over time newer versions of the mobile networking specifications have defined different types of data, and eventually defined support for Internet Protocol (IP) messages as well. Starting with 4G, mobile networks transitioned to an all IP architecture [1]. To support and manage user mobility and provide internet connectivity, today's mobile networks have a complex core network architecture where different Network Functions (NF's) are responsible for the different aspects of user management. This core network continues to evolve with each new specification.

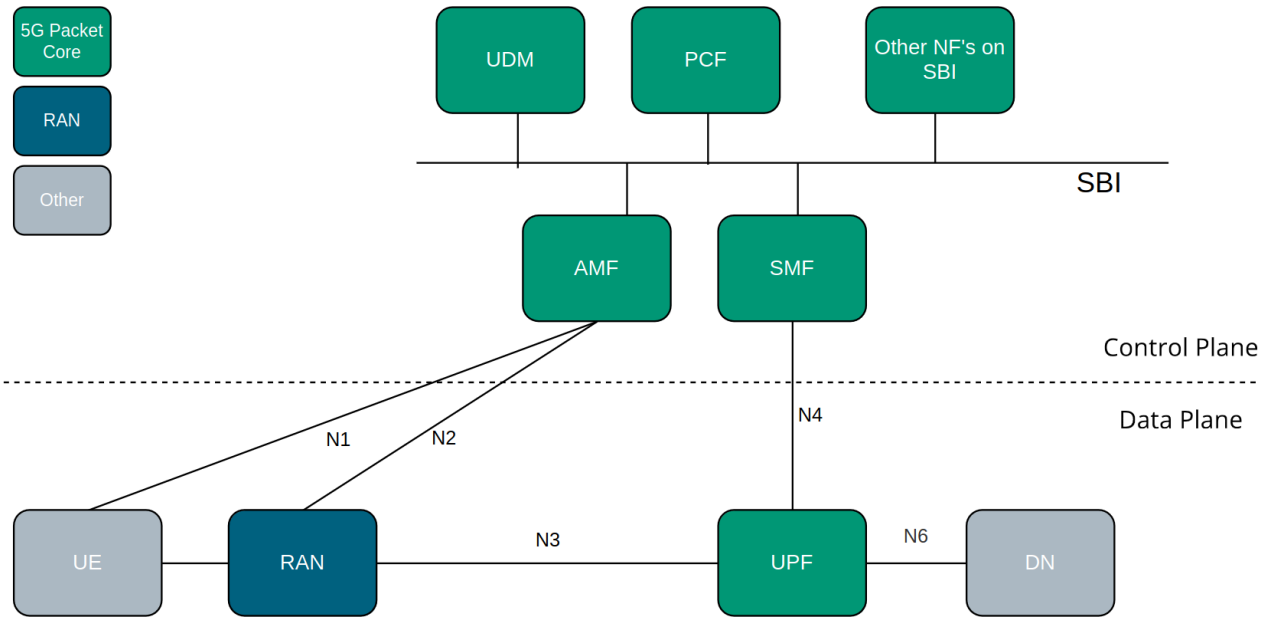


Figure 2.1: Simplified 5G Architecture

Figure 2.1 shows a simplified version of the architecture of a 5G core network and how it is connects to the RAN, the user, and the internet. The figure does not include all of the Network Functions defined in the 5G specifications but shows a select few which will be

briefly explained in following.

The Access and Mobility Management Function (**AMF**) handles, among other things, the user registration (access authentication and authorization), connectivity management, and the control plane interface to the RAN. The reference point for interactions between the AMF and the RAN is called the N2 reference point. The reference point for interactions with the UE is N1. This is a purely logical interface, messages between the UE and the AMF actually have to physically go through the RAN.

The User Plane Function (**UPF**) is responsible for forwarding packets between the Data Network and the RAN. The UPF must also perform QoS enforcement, for example making sure users do not exceed the bandwidth limits of their subscription. The UPF communicates with the RAN over the N3 reference point. For this they utilize the GPRS Tunneling Protocol (GTP). GTP is a tunneling protocol designed for mobile networks. Packets originating in the RAN are contained in a GTP tunnel and must be de-capsulated by the UPF before being forwarded to the Data Network (DN). Packets in the opposite direction must have a GTP header added to them before they are sent to the RAN.

The Session Management Function (**SMF**) is responsible for session establishment and management, and it configures traffic steering in the UPF. The SMF and the UPF communicate over the Packet Forwarding Control Protocol (PFCP) on the N4 reference point. Their interactions are a lot like those between a switch and a controller in a Software Defined Network (SDN), with the SMF acting as the controller.

The Policy Control Function (**PCF**) provides policy rules to different Control Plane functions in order to enforce them. The SMF consults the PCF before informing the UPF what Quality of Service a given user should experience. The interactions between the SMF and the PCF go over the Service Based Interface (SBI). This is a logical interface which represents all of the different interactions between all of the Control Plane Network Functions. All communication on the SBI uses HTTP2.

In the figure it can also be seen how the 5G Core (like the 4G Packet Core before it) splits the control plane and the user plane (or data plane). The network functions in the control plane co-ordinate to make decisions over how to orchestrate the user plane, while the components in the data plane are the ones forwarding the actual user traffic.

2.1.1 5G and Campus Networks

In order to enable new industrial use cases, 5G campus networks may be deployed exclusively for users within a campus organization, and covering only a prescribed geographical area [30]. The deployment of such a network can be crucial for time-sensitive wireless communication, e.g. between robots, and the 5G specifications contain traffic classes and QoS requirements for delay-critical use cases [4]. This is done in order to make 5G a competitive, or even superior, option to WiFi in campus deployments [39].

There is lots of room for different deployment architectures for 5G campus and non-public networks (NPN's). In [29] many of these architectures are discussed and investigated. In order to operate a 5G campus network, spectrum may need to be acquired, as well as 4G or 5G base stations, depending on the mode of operation- NSA (non-standalone: 4G Base Stations (eNodeB's) but a 5G Core Network) or SA (standalone: 5G Base Stations (gNB's) and a 5G Core Network). Furthermore, a 5G Core Network is required in order to manage the network. Since 5G places an increased focus on virtualization and flexibility of deployment, there are a litany of options for which network functions from the core to deploy, and where to deploy them.

This thesis addresses specifically those campus network deployments which require backhaul to a core located off premises. Networks deployed in this fashion will typically feature a subset of the core's network functions deployed on premise. In the simplest case the only component on premise will be the RAN. A different deployment model may feature one UPF on site with a second UPF located in the core, and the backhaul takes place between the

two UPF's. This thesis' WAN Connector will be designed to be able to backhaul IP traffic from the remote campus to the core, either between the RAN and the UPF, or between two UPF's.

2.1.2 Backhaul in 5G

The first hop from the user equipment (UE) into the mobile network occurs over the air, via wireless signals. This communication is between the UE and the Base Station. The network of different Base Stations is called the Radio Access Network (RAN). Once the user has transmitted its data to the RAN, the RAN is responsible for passing on the user's messages to the core network. The transport network between the RAN and the Core is the backhaul network, or "backbone" [22]. However the term backhaul can also be used to refer to the process of transporting data from the RAN back to the core.

In light of the increased focus in 5G on enabling time sensitive industrial use cases, the authors in [28] proposed a method for integrating asynchronous Time Sensitive Networking (TSN) into 5G backhauling. To achieve this their architecture uses an Asynchronous Traffic Shaper, which is crucial for integrating the time sensitive traffic with the regular traffic. Their solution also utilized frame replication for the delay critical flows defined in the 5G specifications (5G QoS Identifiers (5QIs) 82 through 85).

The authors of [22] highlight the need for innovation in 5G backhaul provisioning. At present optic fibre backhaul is not available nationwide in Europe, and Fibre to the Home (FTTH) is scarce worldwide. Currently most backhaul networks are built with microwave links or fibre/copper based links. There has also been research [31, 32] into using multipath routing within the context of millimeter Wave (mmWave) backhaul in order to increase both reliability and throughput.

Another option for mobile backhaul is satellite constellation networks. Geostationary Orbit satellites (GEO satellites) appear motionless to observers from the ground, and thus provide

consistent latency and jitter. However this comes at a cost, they are a large distance away from the earth and round trip times are routinely on the order of 400 to 500 milliseconds. Regardless, GEO satellites are still a crucial means for connectivity for remote regions and areas where it is not possible or very difficult to provide other means of backhaul. Recently, Low-Earth Orbit (LEO) satellite constellations have begun to emerge and represent a more appealing option due to their greatly reduced latencies when compared with GEO satellites [12]. The 3G Public Partnership Project (3GPP), who are behind the 5G specifications, have commissioned a technical report, TR 23.737 [3], on possible architectures and integration scenarios for satellite backhaul in 5G. Furthermore, independent studies have also highlighted the usefulness of satellite backhaul for 5G, and identified it as a key enabler of some of the goals set for 5G [10].

In light of the increased focus on using satellite backhaul, the technical report mentioned above identifies one scenario which pertains exactly to this thesis. The "key issue #9" in the report: "multi connectivity with hybrid satellite/terrestrial backhaul". The report constructs a specific use case for hybrid backhaul. Consider Additive Layer Manufacturing (ALM) factories, wherein the command and monitoring systems for the automation of the factory require reactivity, and machines in the factory must download the ALM electronic files. This download process will saturate the network resources and requires high bandwidth, conversely the automation processes will need low latency and low bandwidth. In order to meet the different requirements of the two applications, the file transfer could be sent over the satellite backhaul, while the remote command and monitoring could take place over the terrestrial link. Precisely this kind of situation highlights the need for a deterministic multipath backhaul solution for 5G campus networks. Here, this thesis addresses the issue more generally, aiming to develop a system that can handle more backhaul options than just one terrestrial link and one satellite link.

2.2 Multipathing and Multihoming

At this point it becomes important to establish the definitions of two similar terms, and how they will be used in this thesis. Multihoming refers to a host, organization, or network, having more than one external link- either to the same Internet Service Provider (ISP), or to more than one [6]. A multihomed host can choose to use different links for different traffic. Multipath routing, on the other hand, is a routing approach which selects separate paths when routing packets. In order to perform multipath routing there must be more than one path from the source to the destination, and the source must be able to choose which path to take. If the source or destination or both are multihomed then it is possible to choose from more than one path between the two of them. In this thesis the term multipathing will be used interchangeably with multipath routing.

This discussion ties into the difference between links (or interfaces) and paths. When two multihomed hosts are communicating, using the same outgoing link does not mean taking the same path. The same interface could be used to send packets with different destination IP addresses and then the packets would take different paths. It is only in the context of a multihomed host communicating with a host that only has one IP address, that selecting a different interface is the same as selecting a different path. In these cases alone, multihoming and multipathing are the same.

With these definitions in hand, we will now review some of the existing literature to see what kinds of approaches have been taken to address problems similar to those of this thesis.

2.2.1 Collecting and/or Predicting Performance Metrics in a Multihomed Environment

In [7] the authors collected both passive metrics (looking at response times for outgoing packets), and active measurements (sending ICMP ping, or TCP SYN messages and measuring

the response time). Using the passive measurements enabled their multihomed approach to perform well, but when using the active measurements the performance was even better. Crucially, the passive measurements worked better over larger sampling periods, because it took longer to get a full overview of all the possible routes, whereas the active sampling approach acquired it's measurements faster and was thus more effective over smaller sampling intervals.

Considering this result, the solution developed in this work will therefore utilize forms of both active and passive measurements. All three metrics- packet loss, latency, and jitter- will be periodically measured in an active manner. The period over which to perform these measurements is also an important design decision for the WAN connector, and it will be a fixed value, which can be configured by the operator. The use of a fixed value is important in the context of a solution which aims to provide determinism, since traffic may often have hard requirements for the maximum tolerable latency, packet loss, or jitter, which a flow is allowed to experience within a given window. For example in the 5G QoS specifications the averaging window for non delay critical traffic is 2 seconds [4].

For classical wired links in multihomed scenarios, [35] have observed that one link will generally dominate with regards to latency, but with brief periods where other links' performance is superior. These same authors also note that with regards to packet loss there is far less prevalence of a "dominant" link, and generally the links will perform comparably. Packet loss is also a particularly difficult metric to measure, since most links are highly reliable and when they do experience packet loss it is in bursts [35]. Wireless connections are usually less reliable and may experience more consistent rates of packet loss at the data link layer, however often this is opaque from the perspective of the higher layers and hard to perceive.

Ultimately, many of the best performing approaches for predicting packet loss, e.g. Hidden Markov Models [11, 35], are still somewhat imprecise and inaccurate. These models assume the link is in one of two states, good or bad, and each state has a different probability for packet loss, and there is a transition matrix which represents the probabilities of switching

from one state to another.

2.2.2 Path and/or Interface Selection

Multipath routing approaches, as well as interface selection in multi-homed environments are mature and well researched topics. Here, a selection of such approaches is covered.

Wireless Sensor Networks (WSN's) are often deployed in challenging environments and must construct their routing tables dynamically. Research into multipath routing solutions for providing QoS guarantees in WSN's can serve as inspiration for this thesis' approach. In [21], the authors address QoS provisioning in a WSN, using multipath routing. Their approach is to formulate the problem as a probabilistic programming problem and then convert it to a linear programming problem using an approximation technique. When solved, the equation they use to represent the problem yields which paths packets should be sent on (or duplicated on) in order to achieve the desired reliability. Only those paths which would guarantee the maximum latency is still below the QoS requirements are considered. This approach bears similarity to how [28] (discussed earlier) uses frame replication for certain flows.

Another article which deserves mentioning is the work in [7]. Here, the authors evaluate a large set of route-control mechanisms and evaluate their trade-offs for multihomed enterprises, in order to derive "best common practices" for multihoming. Their most important results shows that there is no benefit to using historical samples; route selection should always be based on current samples. They even find that employing historical data to make decisions about which path to forward on can prove detrimental at times.

We may also look at the work in [19], where the authors investigate the use of residential multihoming to improve QoS. Their work suggests that even the use of a rudimentary switching algorithm is enough to improve QoS by switching the outgoing ISP based on *any* metric. The authors observed that most of the time, the decision to switch ISP's based on one metric is consistent with the decision that would be made by considering the other

metrics.

In the context of improving QoS we can see that [36] also used a simple path switching algorithm and were able to get positive results. They dynamically adapt the time scale over which these path switching decisions are made by tracking the improvement that would have been possible for each possible value t among a set of predefined options $\{t_n; n = 1, 2, ...N\}$, and changing t if there is a t_n that gives enough of an improvement. By maintaining an exponentially weighted moving average, they give higher weight to the most recent measurements and the corresponding theoretical improvements that would have been possible for different values of t.

Lastly, we can look to the work in [17], which also addresses the issue of cost in it's approach. It presents several different algorithms, including one which minimizes cost by ensuring no ISP serves more traffic than its charging volume is proportional to (e.g. only serve 95th percentile charging volume 5% of the time). It also presents both an offline approach which solves a Mixed Integer Programming problem, and an online approach which performs greedy assignment by always switching to the ISP with the best predicted performance, out of all the ISP's able to provide the required capacity.

2.3 Determinism in Computer Networking

2.3.1 Deterministic Networking Specification

Whilst TSN is able to provide the desired reliability and timing that critical applications require, there is a need for this sort of support in networks which are too large to be classified as Local Area Networks (LANs) [13, 18]. To this extent, Deterministic Networking aims to enable the migration of applications suited for TSN's to larger packet switched networks, while maintaining support for existing packet network applications over the same physical

network [13].

In this thesis, "determinism" does not refer to the IETF DetNet Working Group's definition. However it is still important to consider their specifications and their definition of a Deterministic Network (DetNet), from which our own definition of deterministic backhaul can be built.

The Deterministic Networking Problem Statement contains certain requirements of the network as a whole. These are not possible for this thesis, which addresses backhaul over resources which are not controlled by the operator. The WAN Connector, once implemented, will only be able to select from the available backhaul connections, it has no control over their network. Beyond this however, the problem statement's definition of determinism can be used. This entails absolute guarantees of minimum and maximum latency as well as packet loss and jitter, and secondly the use of more than half of the network's bandwidth. The latter is a crucial aspect of the definition since it prohibits massive over-provisioning of resources, which is an inefficient solution.

2.3.2 Traffic Shaping, Fair Queuing, and Bufferbloat

The DetNet Architecture specification [14] explicitly mentions the use of a traffic shaper to avoid contention loss, but also warns against causing bufferbloat by using oversized buffers while trying to prevent contention loss. This subsection explores what these are and what they entail.

First, to bufferbloat; this phenomenon refers to the use of disproportionately sized buffers, and their effect on the latencies experienced when these buffers are full [8, 16]. The TCP congestion avoidance algorithm depends on dropped packets in order to detect when the bottleneck link along the path is saturated, and reduce its window size in response. When congested links finally do drop packets, if these packets are the last ones in the queue it means the whole rest of the queue must be sent first before the dropped packets are detected and the window size is reduced. This means that there is an unnatural delay before congestion is

detected, and during that period all packets passing through the node experience unnecessary latency.

The first solution to this problem lies in properly sized buffers, another solution is Early Congestion Notification (ECN), wherein TCP is explicitly warned of congestion through flags in the IP headers of messages going back to the sender, instead of through dropped packets. The other solution comes in the form of traffic shaping. Specifically, by using active queue management and fair queuing, respectively, it is possible to selectively drop packets from a buffer (e.g. only those for a greedy TCP flow) and dequeue other packets earlier (e.g. a different flow which is not congesting the link). This overcomes the bufferbloat problem by causing the greedy TCP flow to reduce its throughput and letting through those flows that are not using more than their share of the bandwidth.

Beyond mitigating bufferbloat, traffic shaping can still be crucial for latency sensitive applications. Contention loss and queuing delays can be avoided by smoothing out any large bursts of packets. Furthermore, which traffic shaper to use is a fundamental design decision that must be made. If a link is kept busy at all times, so that maximum throughput is achieved, it means that latency critical packets may not always be forwarded on time. Conversely, by reserving some portion of the bandwidth in case latency sensitive traffic arrives, the full bandwidth cannot be used at all times and thus the throughput is reduced. This is a fundamental trade-off which cannot be avoided.

A traffic shaper is also important for reducing packet loss. By releasing packets at a slower rate or dropping those that can afford to be dropped, bursts of packets may be smoothed out before they overflow a bottleneck link. It should be noted that a similar effect can also be achieved through resource reservation and/or through a scheduler which makes sure to never over-schedule the network's resources. However this requires exact knowledge of both the bandwidth requirements of all the packet flows in the network, as well as the maximum bandwidth of all the links in the network.

Chapter 3

Requirements

To discuss the solution implemented for this thesis requires a review of, firstly, the requirements contained in the problem statement. Concretely, a multipath backhaul solution, as was described in the introduction, will need to be able to transport the 5G traffic from the edge deployment to the core, while respecting the traffic's QoS requirements. On the condition that this must occur in the presence of more than one backhaul option, this breaks down the problem into a short list of hard requirements.

1. Flow Identification

   This is a simple requirement but implementing it efficiently can be a challenge [37]. Flow identification is a basic and unavoidable requirement for this system. All IP packets the WAN Connector receives must be identified and mapped to their appropriate Flow Descriptions. A flow description may match one or more flows, but any single flow should only ever map back to one flow description. Though it is possible for a flow to match more than one flow description, in these instances a tiebreaker can be used: either whichever flow description was seen first or whichever has the higher priority. Packets from flows which do not match any description must be dropped. Packets for flows matching a flow description will need the QoS requirements of the flow description

to be upheld.

2. Management of Multiple Outgoing Paths

   Both the WAN Connector in the core and the one in the remote 5G Campus must be able to co-ordinate about their available interfaces and/or IP addresses. By doing so they should be able to determine how many paths there are between them. Furthermore it is desirable to perform path management and observe the paths' characteristics. These characteristics are the bandwidth, jitter, latency, packet loss, as well as status (i.e. if the path is completely broken). A path's characteristics are also not guaranteed to be symmetric so path's will have to be managed based on their direction as well. This requires a complex degree of co-ordination between the remote WAN Connector instances.

3. Path Selection in Pursuit of Determinism

   It is not enough just to map incoming packets to their QoS requirements, these must also be enforced to as high a degree as possible. Determinism means that a flow can expect it's QoS to be upheld, provided the network is physically able to do so. As long as some path exists which has a latency less than that required by the flow, then the flow's latency requirements should be met. The same applies to packet loss, jitter, and bandwidth. This also means if there are two paths with sufficiently low jitter and delay, and one of them is down, then the flow should be switched to the other path, or it should be replicated across both paths so that one path going down is not even noticeable.

   This is the core building block of this work. Although the previous requirements are all unavoidable, they are all in service of this requirement. It should also be clarified at this point that a flow with QoS requirements which the WAN Connector deems unattainable may be rejected. This does not constitute failure, provided the WAN Connector has correctly deduced that the backhaul paths are all truly insufficient in their current state.

4. Packet Tunneling to Remote Destination

   With the flow identification, path management, and path selection all in place, the incoming packets must still be tunneled to the terminating WAN Connector, which forwards them on to their destination. This is the final requirement of such a system. After incoming packets are mapped to a flow, and the path on which that flow should be forwarded has been chosen (based on the data collected about those paths), the packets must still be sent across one of the backhaul paths. Crucially, the flow's source and destination should not have to be aware of the WAN Connector, this means packets must arrive unchanged.

Chapter 4

Design

This chapter reviews the ways in which the requirements of the previous chapter can be met. It will discuss the components needed in order to address the problem statement. It will try to discuss not only what decisions were made, but also, crucially, what decisions were *not* made, and why. The first section will begin with the fundamental structure of the solution's architecture, thereafter an overview of the functionality required will be given. The Deterministic Networking specification will heavily inform these choices, and is referenced repeatedly throughout this chapter.

4.1 Skeleton Structure

From the outset it was decided to use a control-user plane split. The packet processing and forwarding is performed in the user plane, and the control plane makes the high level decisions about which packets to send where, and in this case, on which outgoing interface to send them. The data plane is simple and only performs the time critical packet processing, and blindly follows the instructions of the control plane. This type of architecture is common in modern software-based networks, for example OpenFlow, the Evolved Packet Core (from

LTE), and the 5G core all implement this kind of control-data plane split.

However, this decision immediately locks one into certain positions. For example, this requires a method of communication between the data plane and the control plane, it also means that the state machine that represents the WAN Connector becomes more complex, and complexity always brings with it a greater danger for hidden mistakes and fallacies. The benefit is that the user plane function does not need any complex decision making logic, thus simplifying its internal architecture. Further, these separate components can then be deployed at different locations, and scaled up or out with greater ease.

4.2 Components Required for Deterministic Multipath Backhaul in a 5G Campus Environment

Determinism, in a networking context, refers to the ability of a network to supply specific QoS requirements with extremely high degrees of certainty. IP flows and applications running in these networks are assured bounded latency, jitter, and packet loss. To that extent, while the Deterministic Networking (DetNet) specification comes with several recommendations for how to achieve determinism it does not define determinism. A network which does not fully comply to these specifications may still call itself determinstic (but it may not say it is a DetNet). Since this thesis aims to provide determinstic backhaul, it is not required to follow the DetNet specifications in any form. Though, it does make sense to consider what solutions they have proposed for determinism, since ultimately the goals are the same. In this section, therefore, the DetNet specifications will be used as a source of inspiration for ideas, but not as an absolute source of truth.

Deterministic networks have been discussed in the background chapter. In order to guarantee determinism the IETF DetNet working group has proposed an architecture for determinism over IP networks [14]. Their specification identifies four key mechanisms for guaranteeing

the determinism of a flow: 1) elimination of contention loss, 2) jitter reduction, 3) service protection, and 4) explicit routes. Since the 5G campus environment may need to use the infrastructure of other operators this rules out the ability to use explicit routes. A key difference between deterministic networks and 5G campus backhaul networks is that the operator may not control all the links between nodes, and that there are only two nodes in the network that are definitely under the administrator's control- the WAN Connectors at the core and the edge. In such a scenario the problem is less like a network and more like a client-server application with multiple paths between the client and the server.

4.2.1 Elimination of Contention Loss and Bufferbloat

Contention between flows must be carefully managed in order to prevent packet loss due to buffer overflows, and, just as importantly, to prevent high latencies due to bufferbloat or excessive queuing. A greedy flow, which uses too much bandwidth, or the presence of too many flows, whose cumulative bandwidth demands exceed the link's capabilities, will have negative effects on all the flows running over the link.

Elimination of contention loss can be achieved by using a traffic shaper and/or rate limiter, and the ingress to any DetNet domain **must** apply such a function. Therefore a traffic shaper should be applied to the ingress traffic entering the WAN Connector from the RAN, before it is sent across the backhaul links. Since the implementation of a traffic shaper is beyond the scope of this thesis, an existing solution must be used. The traffic control subsystem in the linux kernel (tc) provides several implementations of different algorithms for both rate limiting and traffic shaping. For the purposes of this solution several algorithms were considered: Hierarchical Token Bucket (HTB), Hierarchical Fair Service Curve (HFSC) [34], Time Aware Priority Queuing (taprio), Common Applications Kept Enhanced (CAKE) [20], and Fair Queuing with Controlled Delay (FQ_CoDel).

For HTB and HFSC, due to the implementation of these algorithms in the linux tc subsystem,

integrating them within the context of this solution would require filters for each flow. This is not yet necessarily a hindrance, however it would introduce additional overhead upon the creation and/or addition of each new flow. Perhaps more importantly, the implementation of the HTB algorithm provides no means for fair queuing. This means that while it is not possible for greedy and/or malicious flows to use up the bandwidth of other flows because HTB guarantees bandwidth, the greedy flows can cause other flows to experience higher latency because it does not guarantee delay. The HFSC algorithm improves on this by offering both bandwidth and delay guarantees. However this makes it far more difficult to configure, and would require precise calculation of the service curve for each new flow.

Another feasible option available in linux tc would be time aware priority queuing (taprio). This queuing discipline provides scheduled gates for specific traffic classes. It could be used to schedule flows according to their requirements and/or priorities, and thus provide a form of prioritization, and, to an extent, traffic shaping. Flows can be assigned different windows, and congestion due to a greedy flow is avoided if the greedy flow is scheduled for a different window. However one issue is that taprio fails to provide fair queuing. Thus it would need to be configured, and possibly adapted, as new flows are added and removed, depending on their priority. Flows of the same priority would have to compete with each other to use the bandwidth available in their assigned window.

Lastly there are FQ_CoDel and CAKE. Both provide fair queuing as well as active queue management, which means flows are less likely to experience loss due to congestion, particularly congestion caused by one particularly greedy flow. Since the CAKE algorithm was originally designed for routers in home networks, and because WLAN networks are loosely analogous to a 5G campus network (consider the UE's as the Stations, and the Base Station as the Access Point), it makes it an attractive choice. CAKE also allows the use of 8 different priority tins, which fits well with the nature of 5G traffic, where different flows have different priorities. Finally, on a practical level, the implementation of CAKE in the linux kernel uses the kernel's

"skb_flow_dissector" which exposes a hook point for eBPF programs [1]. Specifically within a 5G context, one could attach an eBPF program that allows CAKE to peek inside of GTP tunnels, and differentiate the flows within them. For all the other algorithms mentioned so far, all of the GTP packets would appear to belong to the same IP flow, and thus none of these algorithms would work at all if the traffic is arriving in a GTP tunnel. This is ultimately the strongest case that can be made for any of the options listed so far. TC does have one other algorithm which hooks into the "skb_flow_dissector", namely the Choose and Keep Scheduler (CHOKe) [27], which does perform active queue management, however does not provide fair queuing. This means that while it manages buffers in order to prevent greedy flows from filling them up, it does not guarantee different flows a fair share of the bandwidth, which can still lead to flows experiencing increased delay due to other greedy flows. Appendix B includes a patch which can be used to create an eBPF program which hooks into the "skb_flow_dissector" and removes GPT headers. By loading this eBPF program, the CAKE traffic shaper will also be able to function on packets obscured by GTP.

4.2.2 Jitter Reduction and Latency Guarantees

In the DetNet Architecture specification it is recommended to adopt time synchronization as well as sending "Time of Execution" fields in the application packets, in order to achieve jitter reduction. For this approach, the Precision Time Protocol (PTP) protocol should serve to synchronize time between the two WAN Connectors. Especially since there is an existing implementation- the linuxptp project [2] - which extends PTP to work over IP networks. As for the time of execution fields, it is possible to place these in a GTP header, or to combine them with timestamping.

Methods to guarantee latency bounds are not mentioned within the specification but they are important for the multi-path setting. Specifically when backhaul options may include satellite

[1] https://www.kernel.org/doc/html/v5.1/networking/bpf_flow_dissector.html#overview
[2] https://linuxptp.sourceforge.net/

links it becomes important to consider latencies. Additionally, in a multi-path environment, latency can be a useful criteria for selecting between links, and guaranteeing delay becomes easier to do when it is possible to switch away from a link that starts to exhibit increased latency.

4.2.3 Service Protection

In order to protect against equipment failure, it may be recommendable to perform packet replication and/or encoding. To address this the DetNet specification speaks of both duplication of flows, and coding schemes (e.g. Network Coding [5], or Forward Error Correction (FEC)). Network Coding could have been a clever solution, however it would require controlling, or at the very least coordinating, all the nodes in the backhaul network (like other parts of the Deterministic Networking specifications), which makes it infeasible for this situation. However, to guard against equipment failure and/or packet loss, duplication does provide an option.

Another option would be FEC. The first problem, though, is that most FEC schemes work on a continuous stream of bytes, providing correction for bit errors, as opposed to correcting the loss of entire packets. Of course packet loss can be viewed as a large continuous sequence of bit errors, but this still places strenuous demands on the encoding scheme. There are schemes which address this issue; fountain codes, such as Raptor [24] encode the data as multiple discrete symbols. These symbols can be mapped directly to packets, thus protecting against packet loss as opposed to just bit errors.

Unfortunately the open source ecosystem surrounding fountain codes is not as developed as that around other FEC schemes, and, because of the complexity involved in implementing them efficiently, there are few freely available projects or libraries which can be used to encode packets using such a scheme. For reference, [3] maintains a list of C/C++ FEC libraries and

[3] https://aff3ct.github.io/fec_libraries.html

projects, and none of them support the Raptor family of fountain codes (all other fountain codes incur high overhead).

Even if it were more feasible to integrate FEC into the backhaul solution it bears questioning how much benefit this would bring. FEC is very powerful in situations with consistently lossy links, however paths over the internet tend to experience loss in bursts, not at a consistent rate. To this extent, it may make more sense to duplicate those flows which require a high degree of reliability. This avoids the overhead of forward error encoding, and the duplication ensures that when there are bursts of packet loss they are smoothed over.

If one does choose to duplicate packets in order to achieve service protection, that means an elimination function is also required, as well as a packet ordering function. This is because multiple packets may arrive on one link before they arrive on the other one. If a packet got lost on the faster path, it may still show up on the slower one and thus the packet forwarding needs to pause until the missing packet arrives. The DetNet specifications provide both a basic and an advanced Packet Ordering Function (POF) algorithm [38]. This algorithm can be easily adjusted to also perform the elimination of duplicate packets. The DetNet specification also clarifies how long one should wait to see if a missing packet arrives. This value "cannot be smaller than the delay difference of the paths used by the flow" and is called the $POFMaxDelay$.

```
   /* initialization                                                    */
1  foreach flow f do POFLastSent_f ← 0;
   /* Start POF logic                                                  */
2  while true do
3      receive packet p, with sequence number seq_num for flow f;
4      if seq_num <= POFLastSent_f + 1 then
           /* eliminate duplicates                                     */
5          if seq_num == POFLastSent_f + 1 then
6              forward p;
7              POFLastSent_f = seq_num;
8          end
9      else
           /* in a separate thread                                     */
10         buffer p until seq_num = POFLastSent_f + 1 or POFMaxDelay_f elapses;
11         forward p;
12         POFLastSent_f = seq_num;
13     end
14 end
```

Algorithm 1: Basic POF Algorithm Adjusted for De-Duplication

The algorithm shown in 1 is the Basic POF algorithm, with a single addition in line 5. The addition was made to make sure that an incoming packet's sequence number is not lower than the next one we were expecting, in order to properly eliminate duplicates. The algorithm's **if/else** logic in this representation could be unified in a cleaner way, but it was done this way specifically because it thus requires just one extra *if* condition (in line 5) to be added to the DetNet specification's POF algorithm, and that one change suffices to make it a Packet Ordering *and Elimination* Function (POEF).

4.2.4 Multipath Considerations

While the previous subsections have all considered requirements for determinism in an IP network, the issue of multipathing still remains. For a component which is backhauling over multiple links in an IP networks this means link and/or path selection is required. Traditionally, routers select links primarily based on their ability to route the packet to the destination, and secondarily based on various metrics, which can be defined by the administrator. Routing considerations are not made for the WAN Connector's scenario- it's purpose is to backhaul the traffic from the remote Campus Network deployment to the Core, where the network's own internet gateway can then perform the routing.

At this point it is crucial to differentiate, again, between multihoming and multipathing. Multihomed hosts are able to select one of many links for their outgoing traffic's destination. In multipathing the same link may lead to multiple paths to *different* destinations. When two multihomed hosts communicate with each other multiple different paths are opened up. Multipathing over the internet requires complex data collection about all the links in between the source and the destination, see [15] for an example of such an implementation. This is because multipathing over the internet needs to choose paths with maximally edge-disjoint paths to achieve the best reliability, as well as maximally vertex-disjoint paths in order to reduce the load across all nodes on the path.

The goal of this thesis is to use the link and/or path selection to provide determinism for the flows being backhauled. To this extent, the link selection algorithm must attempt to choose paths which can provide lower latency and jitter than the maximum allowed values of a given flow, as well as meeting the flow's minimum reliability requirements. Another consideration for multipath backhaul environments is that it is also possible to duplicate flows across two links (unlike in traditional backhaul scenarios). It is worth noting that while duplication cannot reduce latency it can improve reliability, since a flow which is duplicated across two paths is far less likely to experience packet loss.

4.2.5 Summary of Necessary Features

Looking back on the previous subsections, the following points (in no particular priority) are identified as mandatory for a deterministic backhaul solution: 1) traffic shaping, 2) path selection according to jitter and latency requirements, 3) service protection (i.e. via packet duplication), 4) packet ordering and de-duplication, and 5) time synchronization. Discussion of the ways in which these point are addressed, or how they are implemented, takes place in the following chapter.

4.2.6 Solution Architecture

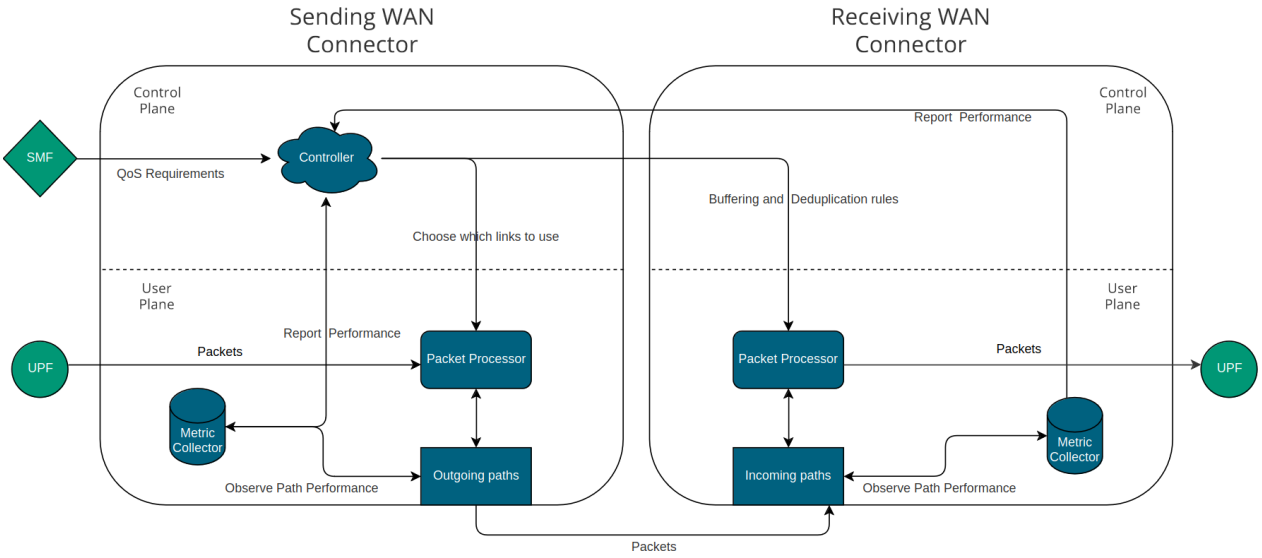


Figure 4.1: Internal Architecture of the WAN Connector

Finally, before the next chapter's discussion of the individual components, the design of the internal architecture of the WAN connector is shown. Figure 4.1 illustrates a scenario where the control plane element of the WAN connector is deployed on the same host that is backhauling traffic to a remote WAN Connector. The diagram is shown for just a single one-way flow. The left side of the diagram is sending the traffic to the right side. Under normal operation there would also be traffic going the other way, and both the "remote"

and the "edge" WAN Connector's data planes would be performing the tasks shown in the "sender" and "receiver" boxes. However this representation hopefully makes it easier to understand the tasks that the WAN Connectors will usually be performing simultaneously, without cluttering the image.

The WAN connector is split into a control plane, which uses the algorithm in 1 to choose which interfaces to forward on, and a data plane, which performs the forwarding, as well as any necessary buffering, re-ordering and/or de-duplication. Statistics are reported from the user plane back to the control plane. The WAN connector performs different tasks depending on whether it is the origin or termination of a flow. For example, the terminating node may be receiving duplicate packets on the other paths, and it must know to drop these. Additionally, the sender, can only store statistics about how many bytes it has sent, and on what paths. The receiver, however, is able to use the information contained in the packets it has received to construct a complete picture about the nature of the path - its jitter, latency and also packet loss. The link selector in the control plane can then use this information to make its decisions. When deploying the WAN Connector there is only one control plane unit (the controller), and two data plane instances. This is because it would be redundant to deploy an additional controller for the remote data plane instance. In the event that each link fails then the remote controller cannot help, even though it can detect it, and if at least one link remains up then the core controller will switch to using that link and can inform the data plane unit to do the same. As has been mentioned before, the graphic depicts a situation in which one of the data plane instances is deployed on the same host as the control plane component. This does not need to be the case, the control plane could also be placed somewhere else entirely, however for the experiments performed in the evaluation chapter, and for the envisioned deployment, this co-location is most likely the easiest to administrate and deploy.

One part which is missing from this diagram is the traffic shaper. That is because it is not connected to the control and the data plane, rather it is meant to be installed alongside

them. Furthermore, it is important that the traffic shaper resides in the part of the network before the packets enter the WAN connector. If the deployment features a UPF in the edge network, it may make sense to place the traffic shaper before this as well. For the core, the traffic shaper will similarly need to be placed at the ingress point where packets first enter the core network.

Chapter 5

Implementation

At this point, one can take the skeleton structure proposed in section 4.1 of the previous chapter, and superimpose the other requirements which were determined in section 4.2. Combining these ideas yields the final implementation, which will be now be presented.

5.1 Overview of the WAN Connector's Features and Components

5.1.1 Path Selection Algorithm

Meeting the various requirements - jitter, latency, and delay - of a flow can be formulated as a multi-constrained QoS problem. Solving such a multi-constrained QoS problem via path selection is a binary optimization problem, and the key to deterministically meeting the flows requirements. The problem can be posed as: "select those paths on which to forward packets while making sure to satisfy the latency, jitter and reliability requirements of the given flow, and minimizing the overall weight of the paths used". The mathematical definition is as

follows:

$$\text{Minimize} \sum_{i=1}^{P} w(x_i) \tag{5.1}$$

$$\text{Where, } d(i) * x_i \leq D \tag{5.2}$$

$$j(i) * x_i \leq J \tag{5.3}$$

$$1 - \prod_{i=1}^{P} (1 - r(i) * x_i) \geq R \tag{5.4}$$

$$\text{for } x_i \in \{0, 1\} \tag{5.5}$$

Here the variables D, J, and R are the flow's delay, jitter, and reliability requirements, while the functions $d(i)$, $j(i)$, $r(i)$, $w(i)$ are the estimated delay, jitter, reliability, and weight of link i. The predicted values will usually just be the latest measurement, as recommended in [7], however there is room here to use more advanced metrics to predict the future link quality and thus perform preemptive path switching in future work. The total number of paths is P. The x_i variable indicates whether or not link i shall be used. If a solution is found, then the flow's packets will be forwarded on each link i where $x_i = 1$, and if no solution can be found which satisfies these conditions then the flow is rejected because its QoS cannot be guaranteed.

It is worth noting that solving such problems is NP-Hard [23]. However this hardness arises primarily because of equation 3.4, the equation for reliability (also the only non-linear equation). Due to this equation one must consider every possible combination of paths on which to forward, and the complexity is $O(2^n)$. This means, even though akella et. al [6] have shown that multihomed approaches experience diminishing returns after more than 4 links, attempting to brute force the solution by limiting the number of outgoing interfaces to 4 still yields a very large problem space - in the worst case both WAN connectors could

have 4 outgoing paths, leading to 16 possible paths and thus 65536 possible combinations to consider.

In order to further avoid the combinatorial explosion the problem needs to be parameterized even more. The first logical parameterization has already taken place by limiting the number of interfaces to 4. This can be expanded on by limiting duplication to only take place over disjoint interfaces. The reasoning behind this is that in a geographically distributed Campus 5G environment, and especially for environments featuring wireless backhaul (e.g. satellite), it is more likely that a link's reliability is most affected by the over the air transmission on the first hop. By performing this parameterization the problem space shrinks considerably, to just 2^4 possible combinations of outgoing paths, but, because certain paths going out on the same interface are ignored, the optimality of the overall solution is gone. This is an acceptable trade off for quick computation, and reasonably strong guarantees on reliability. When choosing which path to include from a set of paths using the same outgoing link, only the path with the greatest reliability is taken into consideration so that the likelihood of rejecting a potentially viable flow is as low as possible.

5.1.2 Packet Ordering and Elimination Function

Since it is possible for flows to be duplicated across multiple paths, it becomes a necessity to have a packet ordering and elimination function. To be able to re-order and de-duplicate packets means that their sequence numbers need to be tracked. This thesis' implementation will track the sequence number on a per flow basis, using the GTP "sequence number" field. This thesis uses it's variation of the DetNet POF specification's Basic POF algorithm, as presented in 1.

5.1.3 Internal Architecture

Control - Data Plane Communication

A protocol is desired which can provide reliable communication over multiple paths in order to communicate between the data plane and the control plane which is where the statistics are collected, flows are added or removed, and the multipath decisions are made. This is crucial, since in a geographically distributed campus 5G deployment it is unlikely that there will be a separate management or control network which uses different underlying network infrastructure than the data plane. Therefore link failure, as well as packet loss, on the link being used by the control plane must either be avoided or protected against. To this extent there are only two feasible options - either multipath TCP or SCTP. SCTP does support multihoming and quick failover when it detects a path is down [26, 33], despite not being explicitly designed for multipathing, and is slightly easier to manage and configure than multipath TCP, which is why it was chosen for the data plane control plane interactions.

GTP Tunneling and Custom Header Extension

The tunneling protocol used between the two data plane instances of the WAN Connector will be GTP version 1 [2]. The specification for GTP allows for the use of sequence numbers, as well as the use of extension headers. For the data plane communication there will always be an additional custom header sent which includes a timestamp taken by the sender. The timestamp is taken in microseconds. For the timestamp there are several options available within linux. The TAI clock in linux (representing the Atomic International Time) is used in this implementation because it does not have leap seconds, and so is a monotonic function, which is an important guarantee for time sensitive applications. An example of a packet capture of the WAN Connector using the custom header is included in the appendix D.

Normally, in 64 bit linux systems, a timestamp takes up 16 bytes and consists of 8 bytes for

the time in seconds since the epoch, and another 8 bytes for the nanoseconds. To reduce the footprint of the timestamp in the header, the seconds are represented with just 8 bytes, thus wrapping around the interval $[0, 255]$. The receiver needs to take this into account, but is programmed to do so, and since the path delays are not realistically expected to ever exceed 3 seconds this is more than enough. The microseconds have a maximum value of 10^6 and can be represented with just 24 bits. For Wide Area Networks and internet connections usually the latencies are on the order of milliseconds and as such microsecond precision is deemed to be sufficient.

Metric Collection

To be able to intelligently switch flows between paths, and to be able to know when this is required, it is necessary to collect the relevant metrics about latency, jitter, bandwidth usage, as well as packet loss. Packet loss can be detected via the sequence numbers, while delay and jitter can be calculated using the timestamps passed along in the GTP headers. These metrics are periodically reported to the control plane so that it can make its decisions on up to date data. Keeping a healthy overview over the state of each path requires periodic probing on these paths. This is especially important for detecting when paths become viable again, since these paths will not have any traffic on them while they are considered down, or if they have experienced high latency and/or jitter recently. The probes are sent once per period of reporting so that they have a minimal impact on the bandwidth usage.

Flow Descriptions, Flows, and Hashing

Incoming packets need to be quickly mapped to their respective flows. It is common in network environments to perform flow hashing, in order to quickly lookup which flow incoming packets belong to. This approach makes sense here too. However, since 5G flow descriptions can apply to various IP and port ranges as well as matching based on the transport layer

protocol it is not possible to hash an incoming flow and obtain the same hash as the flow description. Furthermore it is possible for a flow to match multiple different flow descriptions. In these cases the first matching flow description is taken. This implies some sort of ordered storage of flow descriptions will be required, as well as a method to quickly match packets to flows. The solution used here is to maintain a simple linked list of the known flow descriptions, and match packets to their flow descriptions, if the packet is unknown. For "known" packets, which have been matched to a flow description, these are hashed and stored in a table, alongside their flow descriptor and a list of paths on which the flow is supposed to be forwarded. This allows quick lookup for every subsequent packet, as well as making it simple to change which paths flows are meant to be forwarded on. One alternative to storing the flow descriptions in a linked list would be to store them in a tree, such as in [37], sorted either by priority or ID or even by the hash value of the flow description. This method would provide a faster lookup of new, "unknown" packets, whose hashes don't yet match to a flow, but since new flows are not such a frequent event the additional complexity of implementing such a data structure may not be worth the gain in performance.

It is important to make a good choice when selecting a hash function. Since the hashes used are only for the purpose of lookups it would not be recommendable to pick a hashing algorithm designed for use in cryptography. These algorithms are designed so that it is very difficult to reverse engineer the original value from the hash, and this may often make the computation of the hash more computationally intensive than when computing hash algorithms designed for looking up table entries. Lastly, for hashing algorithms it is desirable that they provide a healthy distribution, to reduce collisions. Collision reduction can also be affected by choice of hash table size- and in general it is usually recommended to use a prime number. In this implementation, the MurmurHash algorithm was chosen due to its strong performance for lookup-based hashing. Specifically, the MurmurHash3 [9] version was chosen, and since it can generate 32 or 128 bit values - the exact sizes of IPv4 and IPv6 addresses - this simplifies the hashing implementation for IP flows. When a packet comes into the WAN connector it is hashed based on it's destination address, source address, destination port, source port,

and finally the transport layer protocol number. If the packet was tunneled with a GTP header, the header is removed and the hash is performed on the tunneled packet, not on the GTP packet. If the hash does not match, the packets is compared to the list of existing flow descriptions. If the flow belongs to one of these descriptions then that decision is stored in the hash table.

Chapter 6

Evaluation

This chapter reviews how the approach from the previous chapter was tested, and discusses the observed results. This chapter will explain the setup of the testbed, as well as the motivation behind the types of tests which were performed. Finally a discussion of the results and their origins will take place.

6.1 Approach to Evaluating Performance

6.1.1 Testbed Setup

In order to evaluate the success of the proposed approach a multipath scenario with four links will be emulated and investigated. Each test will consist of two WAN connectors, and the four emulated links going between them. The characteristics of the links will be changed during the tests, each time with respect to different properties: latency, packet loss, and jitter. The evaluations will be performed on a testbed consisted of four seperate host servers. Two of these servers will act as the originating and terminating WAN Connectors. One server will generate and measure traffic, in between the two WAn Connectors, a server will act as

the link emulator. This architecture is shown in figure 6.1

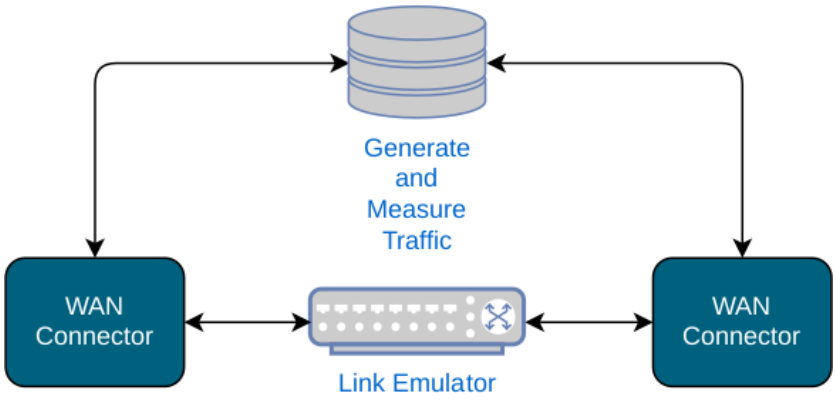


Figure 6.1: Testbed Setup

In figure 6.1, the links between the servers are 10 GB ethernet cables. The hosts are connected as is shown. In the "traffic generator and measurer" two linux namespaces are used to separate the traffic generation and the measurement. The packets generated in one namespace are routed via the ethernet connection to the next host (the terminating or "edge" WAN Connector), because of the linux namespace they are not aware that their destination is technically the same host. The packets are then are backhauled by the WAN Connector over the emulated links in the next host, and via the terminating WAN Connector, back to the original host, but now in the "measurer" namespace.

The link emulation is done using the tc subsystem. First a root Heirarchical Token Bucket qdisc is established on the "downlink" and "uplink" interfaces (the ethernet cables between the WAN Connectors). In this case, the link emulation is performed before the outoing packet is queued on the ethernet cable, so that is also where the qdisc is placed. Secondly, the qdisc is given children classes, one class per link. This allows one to establish bandwidth caps for each emulated interface. Within these HTB classes the root qdisc is a netem qdisc. These allow the installation of link characteristics such as latency, packet loss, and jitter.

In order to make the emulation more realistic, the netem qdiscs will periodically be adjusted, this allows one to degrade or improve links over time, for example by increasing the latency or packet loss in frequent intervals, up to a large value. It also more closely mimics the

real-life behavior of WAN connections, which do experience changes in their characteristics over time.

For the purposes of evaluating the WAN connector, three series of experiments will be performed to isolate and investigate it's link switching capabilities. First, purely latency based link selection will be investigated- a flow will be defined with specific latency requirements and then the emulated links will have their latency repeatedly changed. The same will be done once with packet loss and with jitter. In each of these scenarios the other two parameters will be held constant, so that the WAN Connector can be judged based on its ability to meet one of the latency, jitter, or packet loss requirements, without the other two requirements interfering. In order to simulate realistic scenarios these experiments will also be repeated once with background traffic which aims to saturate the link's capabilities.

The data for the emulation used in all of these scenarios was obtained using the Satellite Constellation Network Emulator (SCNE) [1] from the European Space Agency. The default scenario is 24 hours long and consists of 288 satellites, 10 gateways (GW), and 50 user terminals (UT). The reason this data is used is because LEO satellite links are expected to be the most common type of backhaul link for geographically distributed 5G Campus Network deployments. Furthermore, since LEO satellite connections are exceptionally more volatile than "regular" links [12] [25]. By choosing to emulate more volatile links in the testbed, the WAN Connector experiences a more difficult test, and one can be assured that the real life performance in "normal" scenarios, is highly unlikely to be worse.

Since the WAN Connector is defined to work on at most four outgoing links, each investigated scenario will feature four links. These links will all be based on data from the SCNE emulation.

[1] https://connectivity.esa.int/projects/scne

6.1.2 Accuracy of the Emulation

To make sure that the testbed emulation also matches the values from the SCNE, a brief comparison was done, where the testbed's latencies were compared with those recorded in the SCNE.

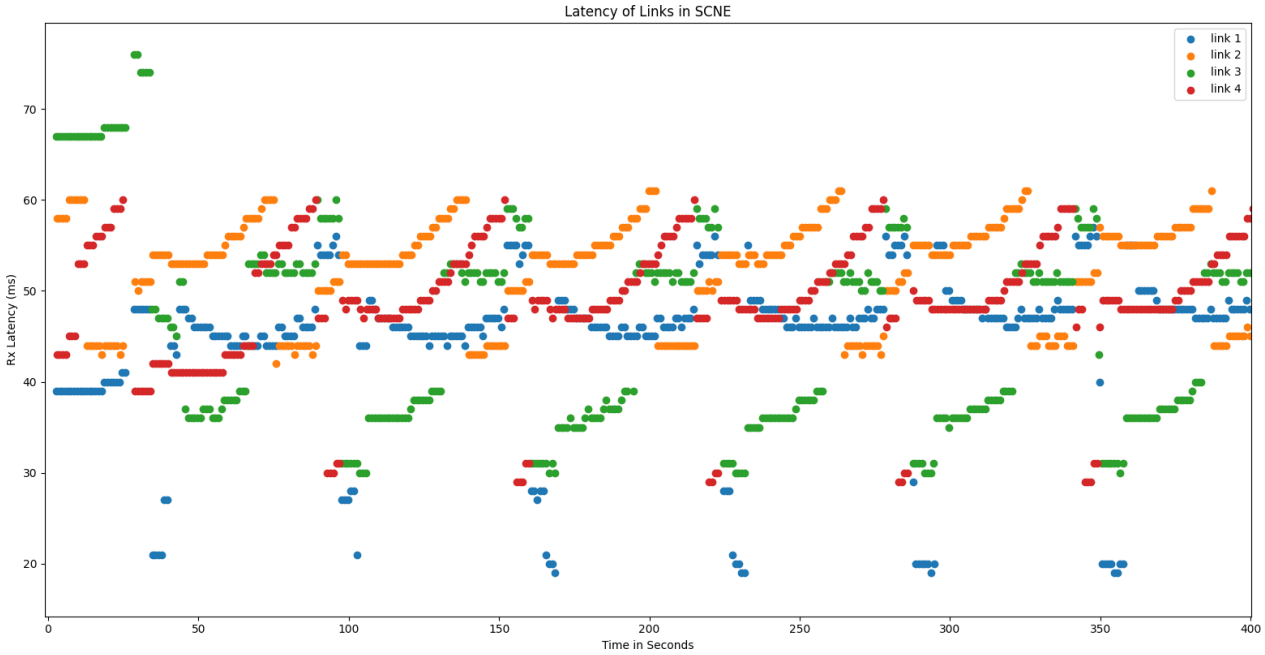
Figure 6.2 shows the comparison between the data observed in the testbed, and the SCNE. The testbed data in the graphic is measured by the packet flows running over the individual links. The emulation data comes from the values recorded during withh the SCNE. Since the testbed data is collected at a rate of 100 packets per second, but runs at 10 times the speed of the SCNE emulation, there are 10 times as many data points. This explains why the testbed data appears blurrier.

As can be seen in the figure 6.2, the latencies recorded on the testbed closely match those from the SCNE, which it was seeking to replicate.

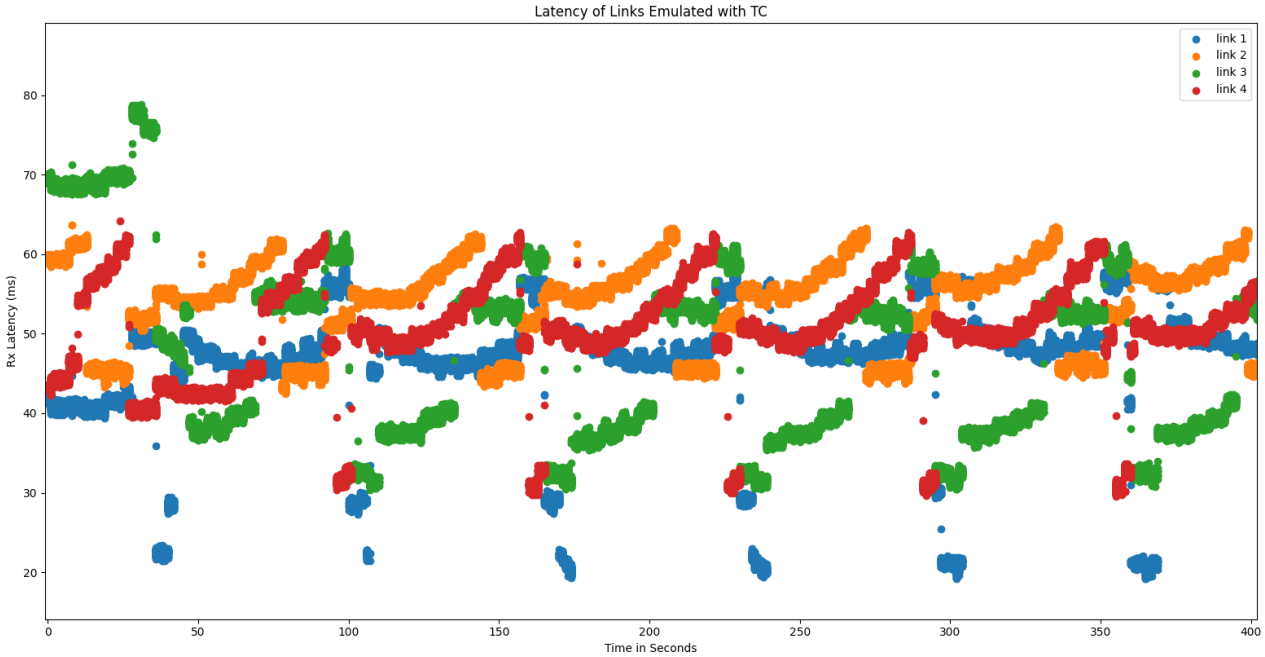
6.2 Latency Based Path Switching

First is the scenario investigating latency. During the emulation, the simulation data is used to adjust the tc netem qdisc on the "link emulator" machine periodically. Because the SCNE simulator provides latency data over a 24 hour period, in 10 second intervals, the simulation data is "sped up". That is the emulation is changed each second, based on the next datapoint from the simulation. This means the 24 hour window can be run over a 2 hours test. Since real links are usually steadier and less volatile than the sped up emulation one can expect results in a real situation to be better.

In the experiment 100 packets are sent per second across each link, as well as 100 packets per second from the traffic generator, through the WAN Connector. This is done to have a comparison between the performance of the WAN Connector and the baseline performance



(a) Latencies Recorded in SCNE



(b) Emulated Latencies Measured in the Testbed

Figure 6.2: Comparison of the Emulators Latencies and the Testbed

level, which would be the best performing link. In any scenario the minimum acceptable performance of a path switching application would be to provide a better performance than the best single link. Otherwise it would be better to just select the best link, and not use the application at all.

6.2.1 Performance Compared to Single Links

For the experiment a flow is defined in the WAN Connector with a minimum round trip time (RTT) of 56 milliseconds. This value is chosen because it is the mean of the mean values of the latencies of the four links from the emulation. For the evaluation of the latency based path switching a single flow matching the definition stored in the WAN Connector is backhauled through the WAN Connector at a rate of 100 packets per second. In order to serve as a comparison, identical 100 packet per second flows are sent across the four other links. Finally their latencies are compared, and the latencies measured by the four links are used to calculate the Oracle approach.

The graphics 6.3 and 6.4 show a cumulative distribution of the different latencies experienced by the packet flows running over the individual links and the WAN Connector, as well as the performance of the Oracle approach. The Oracle results are purely theoretical. For each packet, it is able to select the best link to forward on, based on that links future characteristics. This approach is calculated by comparing each packet sent by the WAN Connector with the latency that packet *would* experience on any of the other links, and if the latency of the WAN Connector's link would be greater than the minimum value (56 ms), selecting a different link (if one exits) with a lower latency. Since it always picks an optimal path, the Oracle approach provides an upper bound on the best possible performance. In practice it is not possible because it has knowledge of the future, but nonetheless it acts as a very valuable comparison.

The CDF shown in figure 6.3 and zoomed in on in figure 6.4 tracks the probability that the

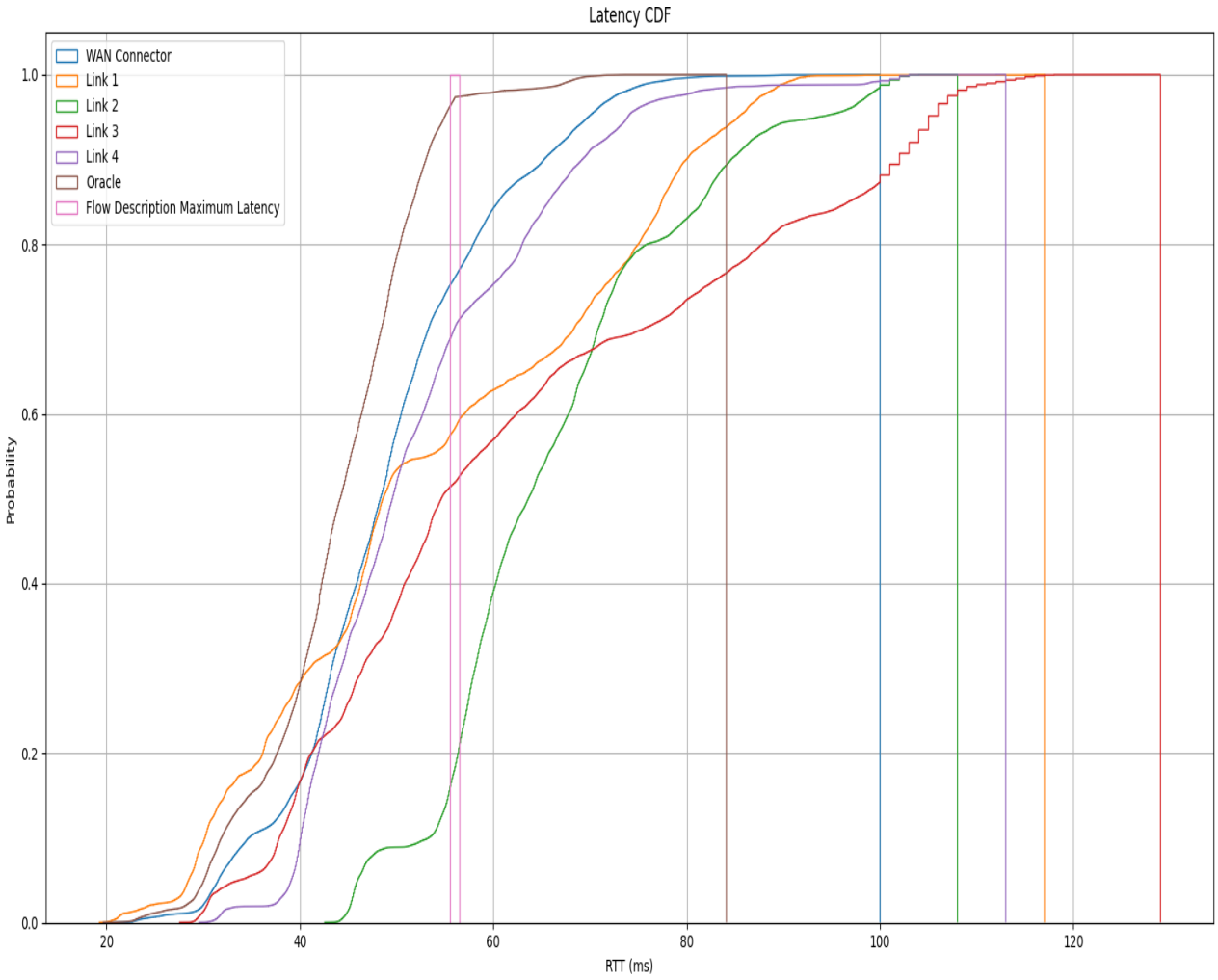


Figure 6.3: Cumulative Distribution of the Latencies

flow experiences a latency less than or equal to the latency value on the y axis. The pink bar labelled "Flow Description Maximum Latency" is set at 56 milliseconds and represents latency bound for the flow being measured, which was communicated to the WAN Connector.

The results in the figures show that the WAN Connector is able to achieve a superior performance to any of the individual links on their own. This was the baseline requirement and it is important that it is able to achieve this. However the performance does not significantly improve on the next best link, which is is able to achieve the maximum allowed latency 72% of the time, while the WAN Connector does so 77.5% of the time. The Oracle

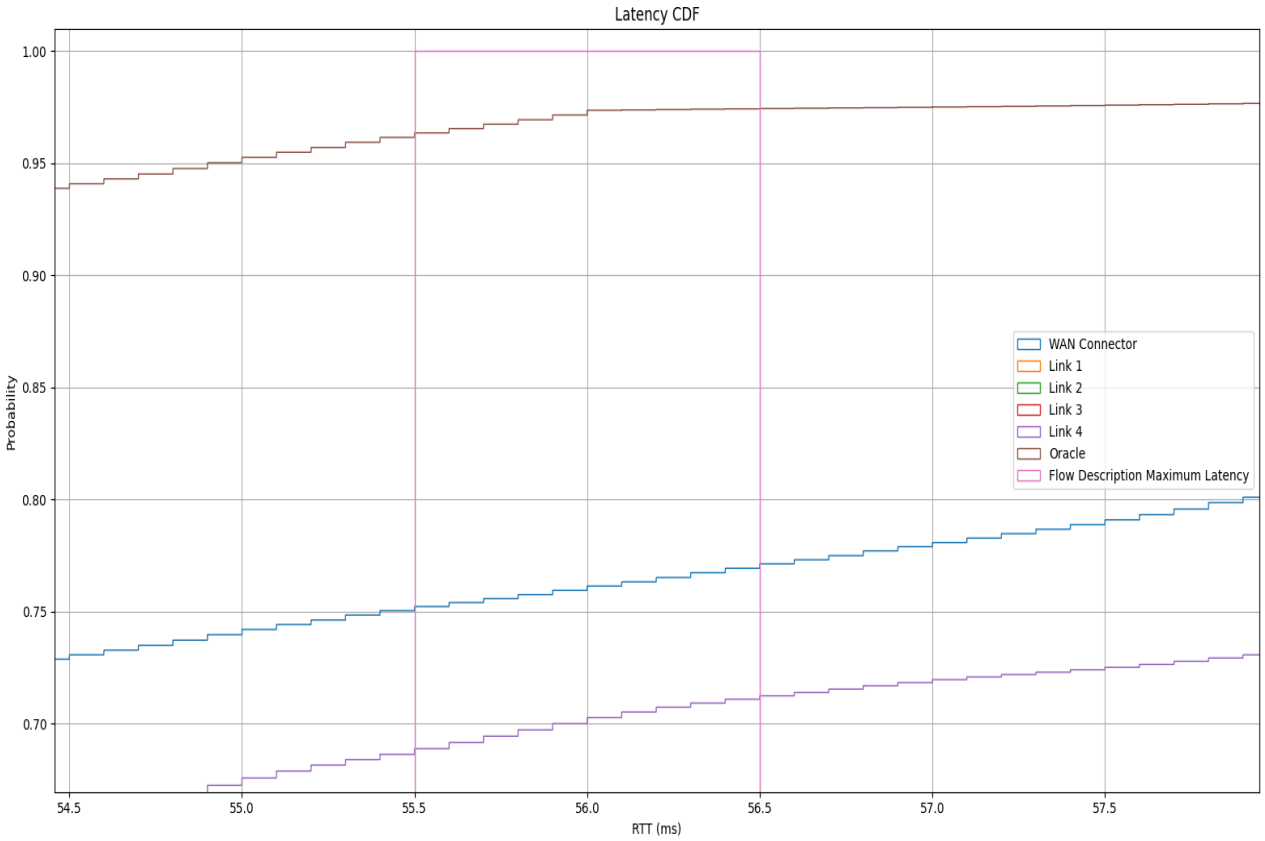


Figure 6.4: Closeup of the Latency CDF

approach shows that in theory one could, at best, have maintained the minimum latency required by the flow in 95% of the cases. This means the WAN Connector's performance is only 80% optimal.

6.2.2 Latency Performance Under Load and the Importance of Traffic Shaping

During this experiment, the same 100 packet per second flow from before is repeated, but background traffic is added. An iperf3 test is run across the WAN Connector at the same time as the latency critical flow, and the background traffic is given the same latency requirements, so that the WAN Connector always schedules both flows to the same link. This ensures that

whatever link is being used will be saturated. Latency performance during congestion and close to congested scenarios is a crucial element of a deterministic backhaul solution. The WAN Connector's control plane only makes decision about which link to forward on- it does not perform load balancing or prioritization. The data plane does not do this either, it implements a purely First In First Out (FIFO) approach to packet queuing. This is done because the burden of implementing an appropriate packet shaper, in addition to the other multipath calculations, is too significant for the purposes of this thesis, and because very powerful traffic shaping implementations, which are easy to integrate into this solution, already exist.

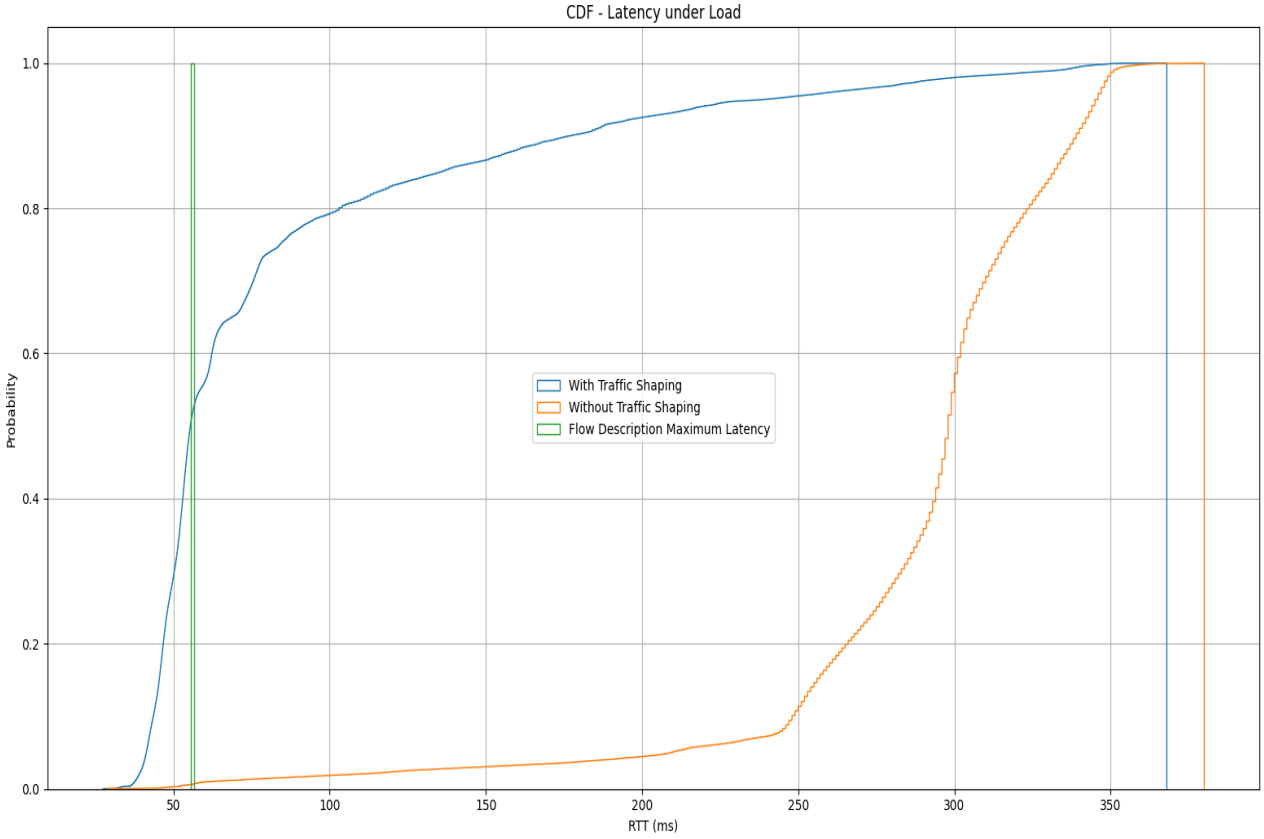


Figure 6.5: Latency with Background Traffic

The figure 6.5 neatly demonstrates why traffic shaping is an essential element of any deterministic network. In the case in which there is no traffic shaper the latency critical flow experiences significant additional latency due to the network being overloaded. The TCP algorithm fills

the link's capacity, and the latency critical packets spend too much time buffering and arrive late. It should be noted that the WAN Connector's performance with the traffic shaper is worse than in the previous scenario, where there was no background traffic, and only 50% of the packets are under the required latency of the critical flow. However the approach without the traffic shaping is at just 1%.

6.3 Reliability Based Path Switching

In the next scenario, to analyze the ability of the WAN Connector to achieve the required reliability of a given flow, the same test flows from before are run over the WAN Connector and the four outgoing links. However this time instead of varying the latency, the packet loss ratio of the link is changed. The SCNE emulation's data is used for the testbed once again. However since the emulation does not provide packet loss on a per second basis, the latencies of all the different links in the simulation are used. For the emulation, these 51 values are looped over in 10 second increments. Each link is given a shuffled set of these 51 values. This means that the cumulative value of the packet loss experienced by all of the links is the same (2%), but the instantaneous values will differ. To appropriately analyze this scenario the flow being backhauled over the WAN Connector is given a reliability threshold of 0.1% this is because in this scenario, where the existing links all provide an average of 2% reliability, it becomes necessary to duplicate at some points, in order to achieve the required reliability. Afterward, the experiment is repeated under load, like in the latency scenario, where the test flows across the links themselves are not run, and a second iperf3 flow is run across the WAN Connector in order to saturate the link. Like before, this second flow is given the exact same requirements as the test flow, in order to ensure that they are always running across the same link.

6.3.1 Packet Loss Measurements

The results of this experiment are shown in figure 6.6. The packet loss recorded by each of the flows over the period of the experiment is plotted in a bar graph, as well as the theoretical packet loss which would have been experienced by the Oracle approach.

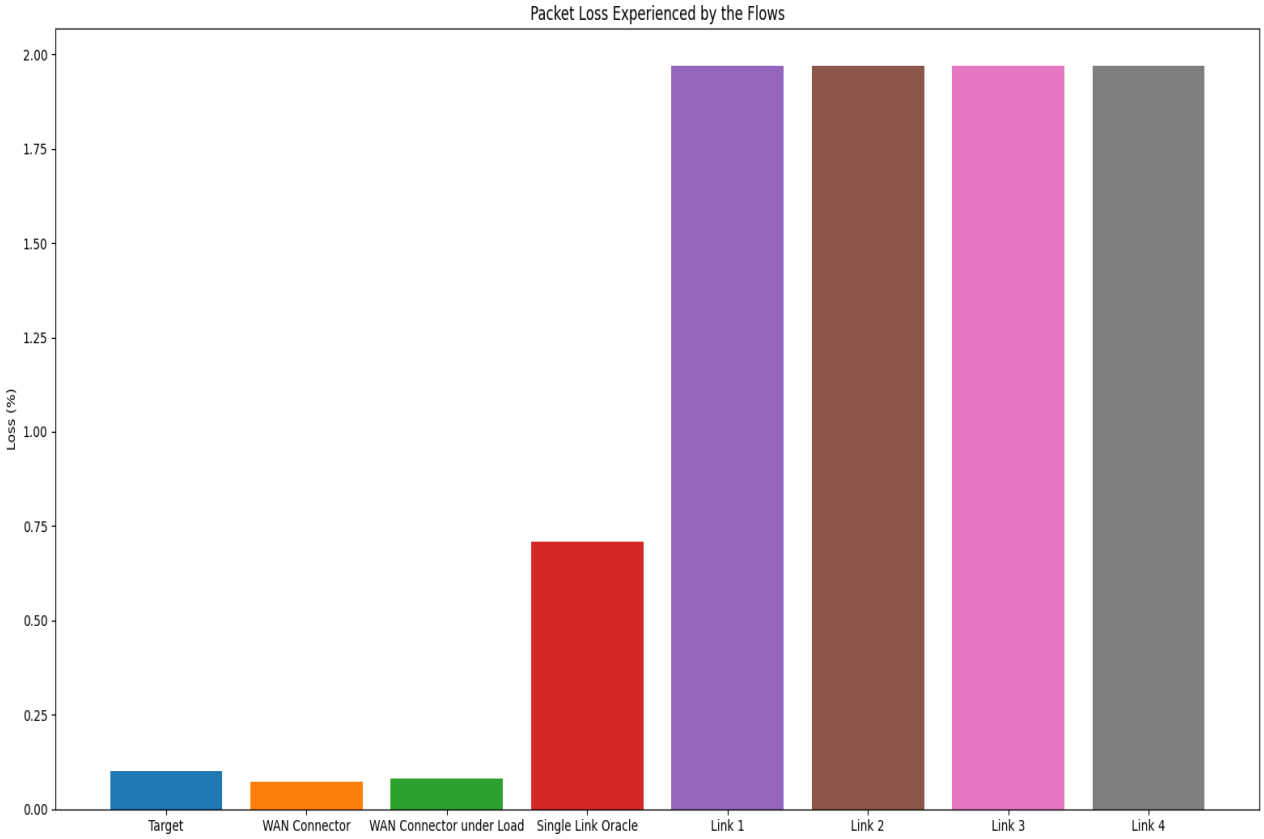


Figure 6.6: Reliability

For the analysis of the performance this time the Oracle approach is defined as choosing, before forwarding any packet, the link with the lowest packet loss ratio. But, crucially, the Oracle is not allowed to duplicate flows across different links. This explains why in figure 6.6 the Oracle is outperformed by the WAN Connector. Indeed, as the graphic shows, it is not possible to achieve the critical flow's required reliability with any of the available links, nor with the Oracle approach. But the WAN Connector is able to achieve it. This speaks to the potency of the flow duplication approach, which is made possible in the optimization

equation used to decided on which flows to forward.

6.3.2 Caveat: Impact on Throughput

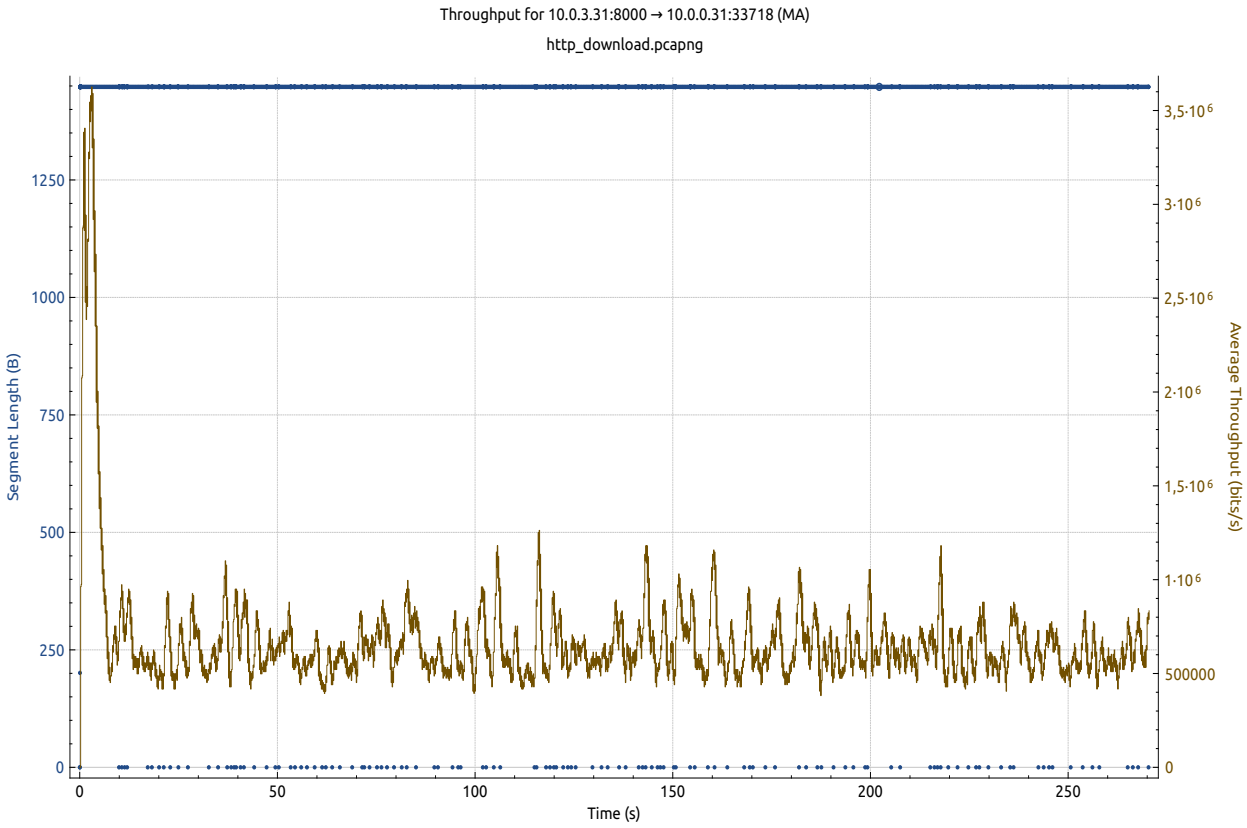


Figure 6.7: Throughput of a Duplicated Flow

One of the unfortunate impacts of packet duplication is that it reduces the available bandwidth. It represents a waste of up to 50% of the available resources, in the worst case, where every flow must be duplicated. Additionally, since flows which are being duplicated need to be re-ordered, this means that the FIFO approach for packets cannot be used and instead the packets must pass through a Packet Ordering and Elimination Function (POEF). When a packet arrives out of order the POEF starts to incur significant additional overhead because it must store all packets until the missing packet arrives, or a point is reached at which the stored packets must be dequeued. This point is either the point at which the packets expire,

or the queue of out of order packets becomes full. Beyond this, the storing and eventual mass de-queuing of out of order packets can also interfere with the TCP algorithm- thus leading to reduced throughput.

These two effects - the POEF overhead and it's effects on TCP- lead to a very heavily reduced throughput, as can be seen in figure **??**. The graph in the figure shows the throughput experienced by a routine TCP download, running over the WAN Connector, and experiencing replication, as well as the segment length of the TCP stream. As can be seen in the graph, the throughput is very low, and is not able to saturate the link's bandwidth despite being the only flow running at the time.

6.4 Jitter Based Path Switching

For the final analysis, the jitter must be investigated. For this purpose the simulation data was once again adapted. This time the standard deviation of the latency experienced was calculated for each of the 102 connections in the simulation. Then these standard deviations were scaled down by diving them by one, two, three, and four, respectively, in order to acquire four different set of jitter characteristics. Finally the scaled values were shuffled. This resulted in 4 links with varying jitter characteristics, which suffices for the emulation. Each different jitter value is emulated for 10 seconds on the respective link. During the emulation scenario a UDP iperf3 stream is ran at a rate of 3/4ths of the link's capability for 1000 seconds. iperf3 records jitter each second, and this data is used to then analyse the performance. Since more than half of the link capacity is being used by the flows being sent on the links (and not through the WAN Connector), the WAN Connector cannot be tested at the same time. Thus, to test the WAN Connector the emulation is repeated but with the flow only being sent over the WAN Connector. Lastly, the Oracle approach is calculated based on the simulation data. The Oracle approach, like before, displays the theoretical optimal performance. It chooses, at each point in time, the link with the lowest jitter. The

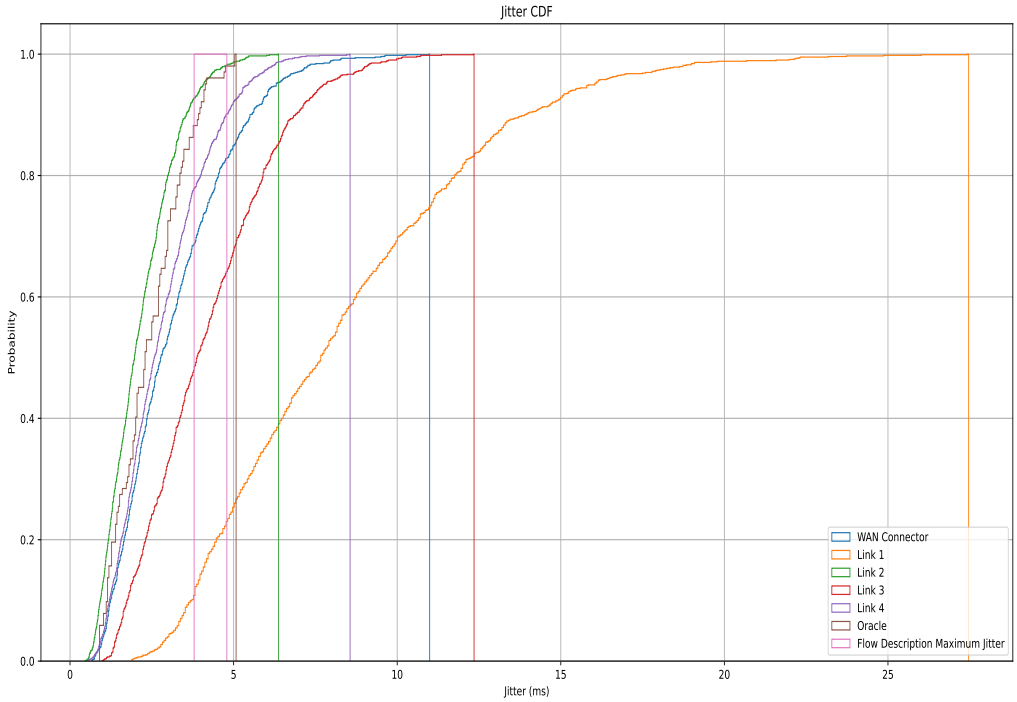


Figure 6.8: Performance of Jitter Based Path Switching

cumulative distribution of the jitter experienced by the flows is plotted in figure 6.8.

As can be seen from the figure, the WAN Connector gave a very disappointing performance. It displays an inferior performance to two of the links. This means the path switching is actually causing a worse performance. This is most likely down to insufficient statistics causing poor path selection, because jitter statistics can't be gathered about other paths in this single flow scenario. The probe packets which are periodically sent out do not provide enough data points to accurately determine jitter. While they can give a sufficient baseline for latency calculations, as well as helping to determine if a link is dead or alive, their jitter measurements are not likely to provide a strong insight into the true nature of the link's jitter.

A final note about the plot in figure 6.8. In the figure the first link actually performs better

than the Oracle for most jitter values, before ultimately proving to have the greater maximum jitter values. This occurs because the Oracle is based on the simulation data and not the measurements. Therefore it is likely that the better performing link experienced, for the most part, very low jitter, and thus because there are more data points available than for the Oracle (and because no doubt there was some small amount of luck with the random values used by tc's netem qdisc), the percentage of packets below a certain level of jitter is greater. However ultimately the maximum values experienced by the flow, during those times when the link experiences it's worst jitter, are worse than the Oracle approach's worst cases, as one would expect.

6.5 Assessment

In light of these results one can conclude that the approach presented in the previous chapter is not without its flaws, however the results have also highlighted some of the functional aspects that were important for good performance. For latency based path switching it was able to outperform the other available links and achieve 80% optimality, and the inclusion of a traffic shaper proved vital for preventing the latency critical flow from experiencing higher buffering times in the presence of a competing flow. For reliability, the WAN Connector showed that it was able to guarantee a critical flow a lower level of packet loss than would have been possible using any single link. Furthermore, it was even able to outperform the single link Oracle, and in a situation where the single link Oracle is unable to provide the required minimum level of packet loss for the desired flow, the WAN Connector was able to deliver the desired reliability. This is a hugely important result because it shows that the WAN Connector can provide greater reliability than even an optimal single link path selection algorithm, via the packet replication. This can enable critical applications to be run in environments where the backhaul options would usually be too unreliable to enable the critical applications.

This section, however, also demonstrates the downside of the packet replication approach, as it greatly reduces the overall throughput. Beyond this, during the jitter experiments, it is seen that the WAN Connector provides a poorer performance than simply using a single link for the entire experiment. This is fundamentally a bad result. The baseline by which any path selection algorithm should be judged is its ability to outperform the individual paths, since it can always select a superior option when one exists. Since the WAN Connector fails in this instance it must be concluded that it cannot provide an overall reduction in jitter for any jitter sensitive applications. The author would suggest, however, that the problem in this respect is not due to the design but rather the implementation. Ultimately there was not enough up to date information about the available links' jitter, so that the path selection algorithm could choose the right one.

Chapter 7

Conclusion and Outlook

7.1 Summary

In review, this thesis has addressed the usage of multiple backhaul paths in a campus 5G setting to achieve determinism. A review of the topic and relevant literature was done in the background chapter, then an approach was developed based on that information and based on the requirements of the problem. Finally this approach was evaluated in a testbed which emulated multiple outgoing links, to see how it performed. The implementation provided was able to achieve some of the goals set for latency and packet loss, but struggled with jitter reduction, as well as utilizing all of the available bandwidth when it is performing packet replication to improve reliability. In the discussion of these results, their potential sources and improvements were addressed.

Placed in a wider context, this thesis presents a foray into an area which will become more and more relevant as 5G campus deployments increase and mature, and as 5G applications which require greater degrees of determinism become more commonplace. On the whole, greater degrees of bespoke packet processing are required for traffic on the internet, as application's evolve and their requirements become more strict. Especially in geographically distributed

settings with multiple backhaul paths, the ability to intelligently select among these paths will be crucial to enabling these types of applications.

7.2 Outlook

7.2.1 Improvements

There are several possible improvements that immediately spring to mind. These could be applied both to the actual implementation as well as to the high level design.

Measuring and Predicting Path Characteristics

Firstly, the path estimation could be adjusted to not just report previous statistics but also attempt to infer what the path might look like in the near future, for example as a path begins to experience increased latency a predictive algorithm/approach might be able to preemptively move flows off of that path, before their latency requirements are violated. This approach could be based on machine learning or AI, or it could use analytical methods.

Furthermore, it would be prudent to use an exponentially weighted averaging function when calculating a path's current characteristics, ba

sed on previous measurements. This would strike a nice balance between re-actively adjusting the predictions based on previous measurements and keeping in mind a path's performance across its entire history.

Also, as the authors in [19] did, this thesis' approach could benefit from adaptive windows of reporting. Instead of collecting statistics at a constant periodic rate, the rate can be adjusted to that period which performs best. This way there is not additional overhead with overly frequent statistical updates, as well as avoiding the reverse situation, where the reporting is

too infrequent for a rapidly changing path.

Lastly, the probe packets which are sent out periodically to measure each path could be adjust to periodically be sent in large bursts, this would serve to provide a more accurate idea of the jitter and the packet loss, since these cannot be measured purely by sending a small number of probe packets. For accurate jitter and reliability measurements more packets are required. This may incur larger overhead on the link due to the bursts of extra packets, however one idea to counteract this would be to adjust the probing so that it is only performed on those links with insufficient traffic currently running through them. The links which are currently backhauling large amounts of traffic will be able to make their own measurements based purely on the backhaul traffic, and there may be no need at all for probe packets. Conversely, the underutilized links will not be able to make good estimations, and sending bursts of probe packets will not interfere with the actual traffic. This would present a solution to one of the biggest issues with the approach presented in this thesis.

Adjustments for Jitter

Another potential improvement would be to pass a "Time of Execution" field in the GTP header of packets of jitter-sensitive applications. This allows the receiving WAN connector to store packets if they have arrived too early, and, conversely, the packet ordering function may use this field to determine that it is more important to forward the current packet now, than to wait for a missing packet. This store and forward approach, with the time of execution field, is one of the solutions mentioned in the Deterministic Networking specification and would be a very sensible addition to this solution.

Additionally, specifically for the Lower Earth Orbit (LEO) satellite case, the path selection could potentially be adjusted to account for the periodic increases in latency as the current satellite leaves the range of the ground station, and the next one comes into range. For example during this phase it might make sense to temporarily forward packets on a different

path, or to enable the store and forward mechanism, before disabling it once the satellite handover has passed.

Forward Error Correction

One possible improvement is the addition of Forward Error Correction (FEC). While this would be difficult to integrate into the equation to select paths, FEC could be used to increase the resilience of consistently lossy links, which is a big benefit for links which commonly exhibit this characteristic, such as wireless links. Especially LEO satellite links tend to present around 1% packet loss [12], which makes the use of FEC much more reasonable since one would only need to add a 2 to 5% overhead, using a fountain code scheme, and thus wind up with a much more reliable link. However as was observed in the evaluation of the WAN Connector's throughput when using the Packet Ordering and Elimination Function, additional computation when receiving packets can lead to reduced overhead. To this extent, the addition of FEC would require efficient decoders and encoders on the sending and receiving side, so that the packet processing speed is not slowed down.

Traffic Prioritization and Rate Enforcement

Perhaps the greatest theoretical flaw in the presented approach is that the WAN Connector does not perform packet prioritization, even though the 5G specifications explicitly mention that flows may have different priorities and should be prioritized accordingly. This is a significant flaw, and very difficult to account for. The most reasonable approach to integrating this is to add it to the traffic shaper. In the experiments performed here, the CAKE traffic shaper was used. It's implementation in the linux kernel does allow for up to 8 different priority "tins" for traffic, however that is not enough for the 5G requirements which allow for more priority levels than just 8. One option, therefore, would be to extend the CAKE implementation in the linux kernel to allow a greater number of priority tins. Alternatively,

the a different traffic shaper could be chosen, or developed, which incorporates prioritization as well as fairness and active queue management at the level of individual IP flows.

A more acute issue is that while CAKE performs fair queuing, this is not the same as what a deterministic solution requires. Fair queuing is a good way to ensure that flows are able to experience a fair share of the bandwidth, but in a deterministic network flows should not be sharing the bandwidth equally, different flows will have different demands. In an ideal world flows would be assigned maximum and minimum rates that they may experience over any given averaging window. These values should then be enforced, thus ensuring that malicious or greedy flows are capped at a certain bandwidth, and flows which have high bandwidth requirements may still receive it. This design could be partially realized using existing solutions, e.g. by integrating the Hierarchical Token Bucket (HTB) queuing discipline (qdisc), which limits bandwidth. However integrating this qdisc requires filter updates for each new flow. This yields a far more complex architecture, and is the reason why it was not utilized in this implementation.

Load Balancing

In instances where more than one path is viable, the algorithm used for path selection does not account for the current load of the paths, nor does it try to perform load balancing when selecting paths. This is because the algorithm attempts to find the minimum weight path. Even in cases where two paths have the same properties and the same weighting, the algorithm does not attempt to shuffle or rotate between them. All flows with similar requirements will wind up on the same path. This can lead to under utilization of other paths and congestion problems on the "primary" path. This could be partially addressed by integrating a degree of random selection for equal paths. However a potentially more interesting approach would be to dynamically update the weights of different paths, based on their current load. This would keep randomness out of the algorithm, for reproducible results, and act as a form of load balancing. However great care would have to be taken with

respect to how the weights are adjusted, so that paths are still appropriately weighted relative to each other. i.e. a path which is deemed twice as expensive as another path should not have more than half as much traffic as the other path, for flows which could go be forwarded on either path.

7.2.2 Implications and Further Areas of Research

The results obtained here imply the presented approach is worthy of further investigation, however only in conjunction with some of the presented improvements. The implementation provided by this thesis cannot achieve determinism without certain adjustments, but nor can it be concluded that it will be definitively unable to do so.

Since the results were only verified in a testbed setup it would make sense to now test the WAN Connector in a real campus 5G deployment where there actually are multiple outgoing paths. Furthermore many of the improvements suggested in the previous section would all be worthy of implementing and evaluating in similar experiments.

Research should also be conducted on evaluating the performance of specific applications performance. For example, the quality of VoIP calls does'nt depend just on latency and jitter [36], but rather how they interact together, and they have their own suites of evaluation criteria. Another specific application to consider has to be interactive video. Mission critical as well as control systems could also be considered in the test suites for evaluation.

It would be interesting to see what benefit AI and machine learning approaches may bring to this problem since they can act more dynamically, and perhaps learn the characteristics of a given link over time. Perhaps they can discover what characteristics the link exhibits right before total failure and thus perform pre-emptive path switching.

Appendix A

Acronyms

3GPP 3rd Generation Public Partnership Project

AMF Access and Mobility Management Function

CAKE Common Applications Kept Enhanced

DN Data Network

eBPF enhanced Berkeley Packet Filter

eNB enhanced NodeB

EPC Evolved Packet Core

FEC Forward Error Correction

FIFO First In First Out

FTTH Fibre to the Home

GEO Geostationary Orbit

gNB gNodeB

GTP GPRS Tunneling Protocol

HTB Hierarchical Token Bucket

HTTP Hypertext Transfer Protocol

IETF Internet Engineering Task Force

IIoT Industrial Internet of Things

ILP Integer Linear Programming

IoT Internet of Things

IP Internet Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISP Internet Service Provider

LAN Local Area Network

LEO Lower Earth Orbit

LTE Long Term Evolution

MNO Mobile Network Operator

NF Network Function

NPN Non Public Network

NP Nondeterministic Polynomial time

NSA non-standalone

NTN Non Terrestrial Network

PCF Policy Control Function

PFCP Packet Forwarding Control Protocol

POEF Packet Ordering and Elimination Function

POF Packet Ordering Function

PTP Precision Time Protocol

QoS Quality of Service

RAN Radio Access Network

RTT Round Trip Time

SA standalone

SBI Service Based Interface

SCNE Satellite Constellation Network Emulator

SCTP Stream Transmission Control Protocol

SDN Software Defined Network

SMF Session Management Function

TCP Transmission Control Protocol

tc Traffic Control

TSN Time Sensitive Networking

UDM Unified Data Mangement

UDP User Datagram Protocol

UE User Equipment

UPF User Plane Function

UPF User Plane Function

VoIP Voice over IP

WAN Wide Area Network

WiFi Wireless Fidelity

WLAN Wireless Local Area Network

WSN Wireless Sensor Network

Appendix B

GTP Patch for the BPF Flow Dissector

As was mentioned in the Design chapter (4), in a real 5G Campus deployment the messages between the UPF (or RAN) and the WAN Connector will be carrying GTP packets. Because GTP encapsulates messages inside a UDP message, the linux kernel will think that all GTP belong to the same flow, when in actuality they may be tunneling multiple different flows. In order for the tc-cake traffic shaper in the linux kernel to be able to detect the packets being tunneled, a custom BPF flow dissector must be used [1].

The linux kernel's source code provides a simple example of a dissector which removes the tunnel headers for various protocols, e.g. Generic Routing Encapsulation (GRE) or IP in IP tunneling (IPIP). Unfortunately a GTP dissector is not included in this set, but it is a straightforward patch to add it.

The listing B.1 shows a patch for the example file included in the linux kernel [2], the patch is for commit 3006adf3be79cde4d14b1800b963b82b6e5572e0 on the linux master branch. This

[1] https://www.kernel.org/doc/html/v5.1/networking/bpf_flow_dissector.html#overview
[2] https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/tree/tools/testing/selftests/bpf/progs/bpf_flow.c

patch constitutes an important part of this thesis' contribution because without it, a realistic deployment of the proposed system would not be possible, because the traffic shaper would not be able to peek inside the GTP tunnel.

After the patch has been applied, the resulting program can be compiled into an eBPF program and loaded into the kernel. Then the tc-cake traffic shaper will be able to detect the flows being tunneled inside of GTP.

```
1
2  diff --git a/bpf_flow.c b/bpf_flow.c
3  index ed437e1..70a08a8 100644
4  --- a/bpf_flow.c
5  +++ b/bpf_flow.c
6  @@ -161,6 +161,8 @@ int _dissect(struct __sk_buff *skb)
7          return parse_eth_proto(skb, keys->n_proto);
8   }
9
10 +#define GTP_ENCAP_OFFSET 24
11 +
12  /* Parses on IPPROTO_* */
13  static __always_inline int parse_ip_proto(struct __sk_buff *skb
       , __u8 proto)
14  {
15 @@ -171,6 +173,10 @@ static __always_inline int parse_ip_proto(
       struct __sk_buff *skb, __u8 proto)
16          struct ethhdr *eth, _eth;
17          struct tcphdr *tcp, _tcp;
18          struct udphdr *udp, _udp;
19 +        struct udphdr *encap_udp, _encap_udp;
20 +        struct iphdr *iph, _iph;
21 +        __u16 srcport;
22 +        __u16 destport;
23
24          switch (proto) {
25          case IPPROTO_ICMP:
26 @@ -242,7 +248,26 @@ static __always_inline int parse_ip_proto(
       struct __sk_buff *skb, __u8 proto)
27                  udp = bpf_flow_dissect_get_header(skb, sizeof(*
                       udp), &_udp);
28                  if (!udp)
29                          return export_flow_keys(keys, BPF_DROP);
30 +                srcport = bpf_ntohs(udp->source);
31 +                destport = bpf_ntohs(udp->dest);
32 +                if (srcport == 2152 && destport == 2152) {
33 +                        keys->thoff += sizeof(*udp) +
       GTP_ENCAP_OFFSET;
34 +                        iph = bpf_flow_dissect_get_header(skb,
       sizeof (*iph), &_iph);
35 +                        if (iph) {
36 +                                keys->thoff += (iph->ihl << 2);
37 +                                encap_udp =
       bpf_flow_dissect_get_header(skb, sizeof(*encap_udp), &
       _encap_udp);
38 +                                if (encap_udp) {
39 +                                        keys->thoff -= (iph->ihl
        << 2);
```

```
40 +                                            keys->is_encap = true;
41 +                                            return parse_eth_proto(
   skb, bpf_htons(ETH_P_IP));
42 +                                    } else {
43 +                                            bpf_printk("Couldnt
   dissect encapsulated udp header");
44 +                                    }
45 +                            } else {
46 +                                            bpf_printk("Couldnt
   dissect encapsulated ip header at GTP offset (%d bytes)",
   keys->thoff);
47 +                            }
48
49 +                    }
50                      keys->sport = udp->source;
51                      keys->dport = udp->dest;
52                      return export_flow_keys(keys, BPF_OK);
```

Listing B.1: Patch to remove GTP Headers for the BPF Flow Dissector Example in the Linux Kernel

Appendix C

JSON Formats Used in Control and Data-Plane Messages

This appendix contains excerpts of the JSON messages passed between the control plane and the data plane during operation. Listing C.1 shows how the control plane sends updates to the data plane nodes. Listing C.2 shows the format of the messages used for reporting statistics.

These are the only two message formats used between the control plane and the data plane. These messages are sent over SCTP, UTF-8 encoded. They are the only data in the SCTP messages' bodies. By using JSON there are three distinct advantages. Firstly, a new protocol doesn't need to be developed for the messaging between control and data plane. Secondly, the messages are both human and machine readable which makes for easier debugging. Lastly, because it is a hugely popular data format there are lots supporting libraries.

The messages which add new flows to the data plane contain the flow description as well as its priority, and the Packet Ordering Function (POF) maximum delay, which was mentioned in the description of the POF algorithm in the design and implementation chapters.

In the reverse direction, when the data plane nodes report to the control plane, the statistics reported include the paths' descriptions (source and destination IP addresses) as well as the number of packets and bytes received on that path. They also include the average jitter and latency experienced, both over the entire reporting period, and since the last message. Finally, the message are also sent with a list of the "sending" statistics. That is, the number of bytes and packets sent out on the WAN Connector's interfaces. Both the path and interface statistics are sent with three timestamps. One timestamp to describe the first packet sent or received during this reporting window, one to describe the last packet sent or received during that same reporting window, and a timestamp taken before the message itself was sent (the heartbeat timestamp).

By including the "sending" statistics, the data plane is able to inform the control plane if an interface is being actively used. On the receiving side, if an interface has not received packets from the respective source in too long a time, then the control plane can infer complete link failure. This is crucial for the ability of the WAN Connector to switch all user traffic off of dead links. Once a link stats working again, the WAN Connector will detect this when it receives the periodically sent probe packets.

In order to save space this message was edited for display so that it only contains messages about two of the available paths instead of all the paths.

```
1
2 {
3   "flowsToAdd": [
4     {
5       "flowID": 1,
6       "priority": 1,
7       "flowDescription": {
8         "src": "10.0.0.31",
9         "srcSubnet": 24,
10        "srcPortStart": 0,
11        "srcPortEnd": 0,
12        "dst": "10.0.3.31",
13        "dstSubnet": 24,
14        "dstPortStart": 0,
15        "dstPortEnd": 0,
16        "protocol": 1,
17        "direction": "in"
18      },
19      "POFmaxDelayMicros": 0,
20      "nForwarding": 1,
21      "decisions": [
22        {
23          "interfaceName": "link3",
24          "remoteEndpoint": {
25            "ip": "10.1.22.21",
26            "port": 2152
27          }
28        }
29      ]
30    },
31    {
32      "flowID": 2,
33      "priority": 1,
34      "flowDescription": {
35        "src": "10.0.0.31",
36        "srcSubnet": 24,
37        "srcPortStart": 0,
38        "srcPortEnd": 0,
39        "dst": "10.0.3.31",
40        "dstSubnet": 24,
41        "dstPortStart": 0,
42        "dstPortEnd": 0,
43        "protocol": 17,
44        "direction": "in"
45      },
```

```
46        "POFmaxDelayMicros": 0,
47        "nForwarding": 1,
48        "decisions": [
49          {
50            "interfaceName": "link3",
51            "remoteEndpoint": {
52              "ip": "10.1.22.21",
53              "port": 2152
54            }
55          }
56        ]
57      }
58    ]
59  }
```

Listing C.1: Format of Flow Forwarding Decisions from Control Plane

```
{
 "pathStats": [{
   "src": "10.1.33.31",
   "dst": "10.0.33.32",
   "nbytes": 6272,
   "npackets": 224,
   "ndropped": 0,
   "avgLatencyMicros": 30655,
   "avgJitterMicros": 1835,
   "bytesSince": 56,
   "packetsSince": 2,
   "droppedSince": 0,
   "latencySinceMicros": 29276,
   "jitterSinceMicros": 1431,
   "earliestRecvdMillis": 1703448057713,
   "lastRecvdMillis": 1703448057716,
   "heartbeatTimestampMillis": 1703448058929
  }, {
   "src": "10.1.44.41",
   "dst": "10.0.44.42",
   "nbytes": 6272,
   "npackets": 224,
   "ndropped": 0,
   "avgLatencyMicros": 28648,
   "avgJitterMicros": 4182,
   "bytesSince": 56,
   "packetsSince": 2,
   "droppedSince": 0,
   "latencySinceMicros": 29540,
   "jitterSinceMicros": 5351,
   "earliestRecvdMillis": 1703448057711,
   "lastRecvdMillis": 1703448057718,
   "heartbeatTimestampMillis": 1703448058929
  }],
 "interfaceStats": [{
   "name": "link2",
   "src": "10.0.33.32",
   "nbytes": 162768024,
   "npackets": 109020,
   "bytesSince": 56,
   "packetsSince": 2,
   "earliestSentMillis": 1703448055928,
   "lastSentMillis": 1703448055928,
   "heartbeatTimestampMillis": 1703448058929
```

```
46    }, {
47      "name": "link4",
48      "src": "10.0.44.42",
49      "nbytes": 174418872,
50      "npackets": 116808,
51      "bytesSince": 56,
52      "packetsSince": 2,
53      "earliestSentMillis": 1703448055928,
54      "lastSentMillis": 1703448055928,
55      "heartbeatTimestampMillis": 1703448058929
56    }]
57  }
```

Listing C.2: Format of Statistics from Data Plane

Appendix D

Example of Custom GTP Header Format

This appendix contains a screenshot of a packet captured during normal operation of the WAN Connector (figure D.1), displayed using wireshark [1]. The packet dissection shows the use of the GTP hedaer, along with the additional custom header, which is 8 bytes long. The custom header consists of 2 bytes of regular overhead from the GTP protocol as well as a 4 byte timestamp and 2 spare bytes. This packet was captured during regular operation of the WAN Connector, in the jitter experiment.

[1] https://www.wireshark.org/

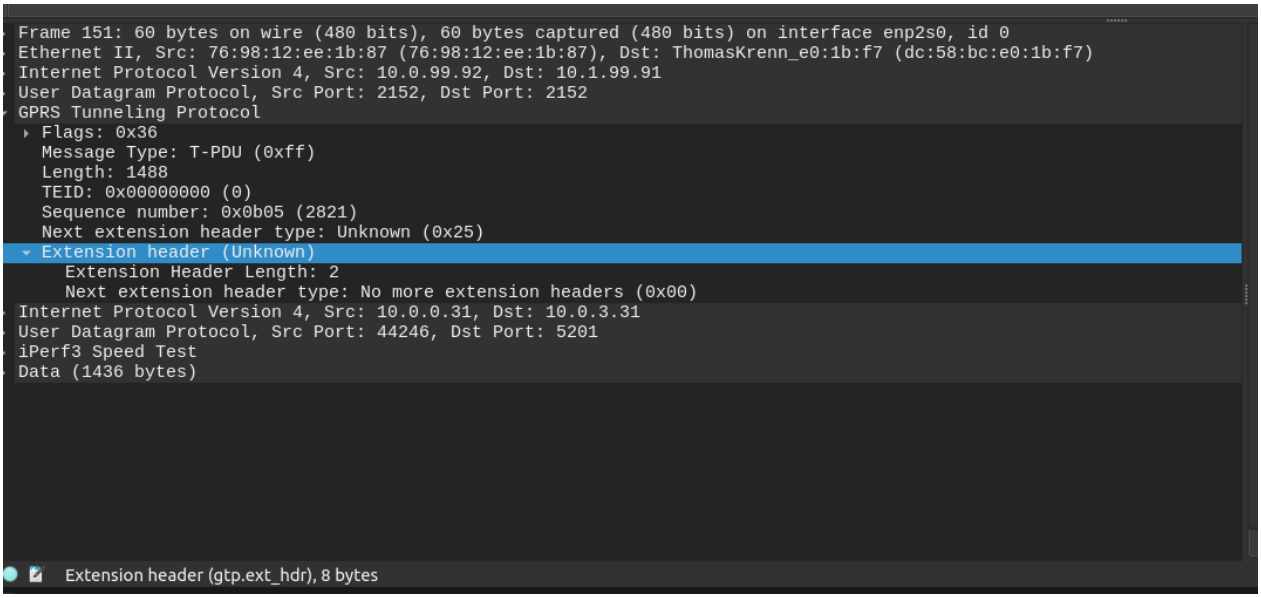


Figure D.1: Screenshot of Wireshark Dissection

Bibliography

[1] 3GPP. *Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E-UTRAN); Overall description; Stage 2*. Technical Specification (TS) 36.300. Version 17.4.0. 3rd Generation Partnership Project (3GPP), Apr. 2023.

[2] 3GPP. *General Packet Radio Service (GPRS); GPRS Tunnelling Protocol (GTP) across the Gn and Gp interface*. Technical Specification (TS) 29.060. Version 17.4.0. 3rd Generation Partnership Project (3GPP), Oct. 2022.

[3] 3GPP. *Study on architecture aspects for using satellite access in 5G*. Technical Report (TR) 23.737. Version 17.2.0. 3rd Generation Partnership Project (3GPP), Mar. 2021.

[4] 3GPP. *System architecture for the 5G System*. Technical Specification (TS) 23.501. Version 17.8.0. 3rd Generation Partnership Project (3GPP), Apr. 2023.

[5] Rudolf Ahlswede et al. "Network information flow". In: *IEEE Transactions on information theory* 46.4 (2000), pp. 1204–1216.

[6] Aditya Akella et al. "A measurement-based analysis of multihoming". In: *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*. 2003, pp. 353–364.

[7] Aditya Akella et al. "On the performance benefits of multihoming route control". In: *IEEE/ACM Transactions on Networking* 16.1 (2008), pp. 91–104.

[8] Mark Allman. "Comments on bufferbloat". In: *ACM SIGCOMM Computer Communication Review* 43.1 (2012), pp. 30–37.

[9] Austin Appleby. "MurmurHash3, 2012". In: *URL: https://github. com/aappleby/smhasher/blob/master/cpp* (2012).

[10] Xavier Artiga et al. "Terrestrial-satellite integration in dynamic 5G backhaul networks". In: *2016 8th advanced satellite multimedia systems conference and the 14th signal processing for space communications workshop (ASMS/SPSC)*. IEEE. 2016, pp. 1–6.

[11] Anat Bremler-Barr et al. "Predicting and bypassing end-to-end Internet service degradations". In: *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*. 2002, pp. 307–320.

[12] Joerg Deutschmann, Kai-Steffen Hielscher, and Reinhard German. "Broadband internet access via satellite: Performance measurements with different operators and applications". In: *Broadband Coverage in Germany; 16th ITG-Symposium*. VDE. 2022, pp. 1–7.

[13] Norman Finn and Pascal Thubert. *Deterministic Networking Problem Statement*. RFC 8557. May 2019. DOI: 10.17487/RFC8557. URL: https://www.rfc-editor.org/info/rfc8557.

[14] Norman Finn et al. *Deterministic Networking Architecture*. RFC 8655. Oct. 2019. DOI: 10.17487/RFC8655. URL: https://www.rfc-editor.org/info/rfc8655.

[15] Igor Ganichev et al. "YAMR: Yet another multipath routing protocol". In: *ACM SIGCOMM Computer Communication Review* 40.5 (2010), pp. 13–19.

[16] Jim Gettys. "Bufferbloat: Dark buffers in the internet". In: *IEEE Internet Computing* 15.3 (2011), pp. 96–96.

[17] David K Goldenberg et al. "Optimizing cost and performance for multihoming". In: *ACM SIGCOMM Computer Communication Review* 34.4 (2004), pp. 79–92.

[18] Ethan Grossman. *Deterministic Networking Use Cases*. RFC 8578. May 2019. DOI: 10.17487/RFC8578. URL: https://www.rfc-editor.org/info/rfc8578.

[19] Ahsan Habib and John Chuang. "Improving application QoS with residential multihoming". In: *Computer Networks* 51.12 (2007), pp. 3323–3337.

[20] Toke Høiland-Jørgensen, Dave Täht, and Jonathan Morton. "Piece of CAKE: a comprehensive queue management solution for home gateways". In: *2018 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. IEEE. 2018, pp. 37–42.

[21] Xiaoxia Huang and Yuguang Fang. "Multiconstrained QoS multipath routing in wireless sensor networks". In: *Wireless Networks* 14.4 (2008), pp. 465–478.

[22] Mona Jaber et al. "5G backhaul challenges and emerging research directions: A survey". In: *IEEE access* 4 (2016), pp. 1743–1766.

[23] Mark W Krentel. "The complexity of optimization problems". In: *Proceedings of the eighteenth annual ACM symposium on Theory of computing*. 1986, pp. 69–76.

[24] Michael Luby et al. *Raptor forward error correction scheme for object delivery*. Tech. rep. 2007.

[25] Sami Ma et al. "Network characteristics of LEO satellite constellations: A Starlink-based measurement from end users". In: *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*. IEEE. 2023, pp. 1–10.

[26] Yoshifumi Nishida et al. *SCTP-PF: A Quick Failover Algorithm for the Stream Control Transmission Protocol*. RFC 7829. Apr. 2016. DOI: 10.17487/RFC7829. URL: https://www.rfc-editor.org/info/rfc7829.

[27] Rong Pan, Balaji Prabhakar, and Konstantinos Psounis. "CHOKe-a stateless active queue management scheme for approximating fair bandwidth allocation". In: *Proceedings IEEE INFOCOM 2000. Conference on Computer Communications. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No. 00CH37064)*. Vol. 2. IEEE. 2000, pp. 942–951.

[28] Jonathan Prados-Garzon and Tarik Taleb. "Asynchronous time-sensitive networking for 5G backhauling". In: *IEEE Network* 35.2 (2021), pp. 144–151.

[29] Jonathan Prados-Garzon et al. “5G non-public networks: Standardization, architectures and challenges”. In: *IEEE Access* 9 (2021), pp. 153893–153908.

[30] Justus Rischke et al. “5G campus networks: A first measurement study”. In: *IEEE Access* 9 (2021), pp. 121786–121803.

[31] Salman Saadat, Da Chen, and Tao Jiang. “Multipath multihop mmWave backhaul in ultra-dense small-cell network”. In: *Digital Communications and Networks* 4.2 (2018), pp. 111–117.

[32] Kari Seppänen et al. “Multipath routing for mmWave WMN backhaul”. In: *2016 IEEE International Conference on Communications Workshops (ICC)*. IEEE. 2016, pp. 246–253.

[33] Randall R. Stewart, Michael Tüxen, and karen Nielsen. *Stream Control Transmission Protocol*. RFC 9260. June 2022. DOI: 10.17487/RFC9260. URL: https://www.rfc-editor.org/info/rfc9260.

[34] Ion Stoica, Hui Zhang, and TS Eugene Ng. “A hierarchical fair service curve algorithm for link-sharing, real-time and priority services”. In: *ACM SIGCOMM Computer Communication Review* 27.4 (1997), pp. 249–262.

[35] Shu Tao et al. “Exploring the performance benefits of end-to-end path switching”. In: *Proceedings of the joint international conference on Measurement and modeling of computer systems*. 2004, pp. 418–419.

[36] Shu Tao et al. “Improving VoIP quality through path switching”. In: *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.* Vol. 4. IEEE. 2005, pp. 2268–2278.

[37] Alok S Tongaonkar. “Fast pattern-matching techniques for packet filtering”. PhD thesis. Citeseer, 2004.

[38] Balazs Varga et al. *Deterministic Networking (DetNet): Packet Ordering Function*. Internet-Draft draft-ietf-detnet-pof-08. Work in Progress. Internet Engineering Task Force, Dec. 2023. 12 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-detnet-pof/08/.

[39] Jaspreet Singh Walia, Heikki Hämmäinen, and Marja Matinmikko. "5G Micro-operators for the future campus: A techno-economic study". In: *2017 internet of things business models, users, and networks*. IEEE. 2017, pp. 1–8.