



# 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

December 7, 2023

MASTER'S THESIS

Telecommunication Systems Institute

Technische Universität Berlin

Examiner 1: Prof. Dr. Thomas Magedanz

Examiner 2: Prof. Dr. Axel Küpper

Advisor: Hauke Buhr

# 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, December 7, 2023

Nicolas Zunker

# Abstract

TODO

# Table of Contents

# List of Figures

# 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 will be a greater diversification of the classes of service. 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, in the Industrial Internet of Things (IIoT) the requirements on latency, jitter, and reliability may be extremely stringent. 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 pre-existing on-site ISP connections. Providing the kind of deterministic quality of service that these sites may require can be a very difficult challenge.

Network operators may choose to utilize more than one backhaul connection at the same

time, in order to increase the available bandwidth or to utilize the different qualities of the backhaul links. This bears the question whether multipathing could then 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 existing body of research in these fields, which has demonstrated that QoS can be improved by using multiple links or paths simultaneously [1, 2, 5–7, 9].

## 1.2 Goal

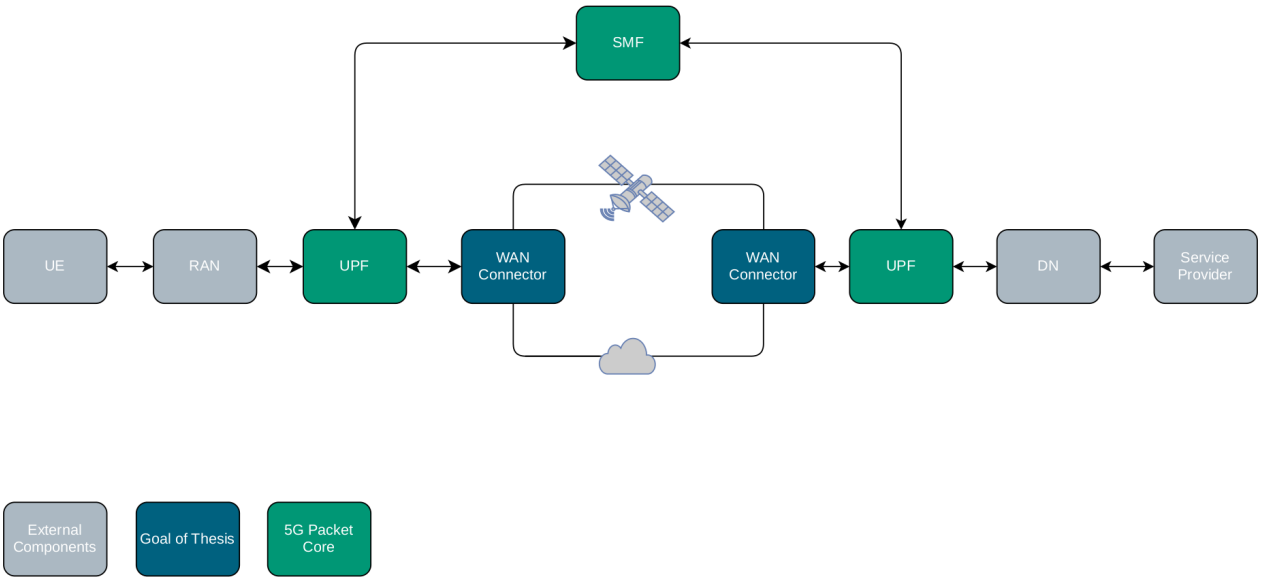


Figure 1.1: 5G Deployment with 2 UPFs

The goal of this thesis is to design a WAN connector, that can be placed at the ingress and egress point of two or more locations, and utilizes 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 in a geographically distributed 5G campus network, which has more than one egress link, and in the core . Then, using the

multiple outgoing links, the traffic is backhauled to the other sites, while respecting it's QoS requirements. This can be especially beneficial for critical applications (e.g. between industrial sites), which are in locations that do not have access to optical fibre for backhaul.

For a 5G deployment the proposed WAN connector could be deployed between multiple User Plane Functions (UPF) and Data Networks (DNs) which also host WAN connectors, in order to provide deterministic backhaul. The architecture for such a deployment is shown in Figure 1.1. Though the on-site UPF is not a strict requirement, it is also feasible to connect the RAN directly to the WAN Connector.

### Proposed Solution

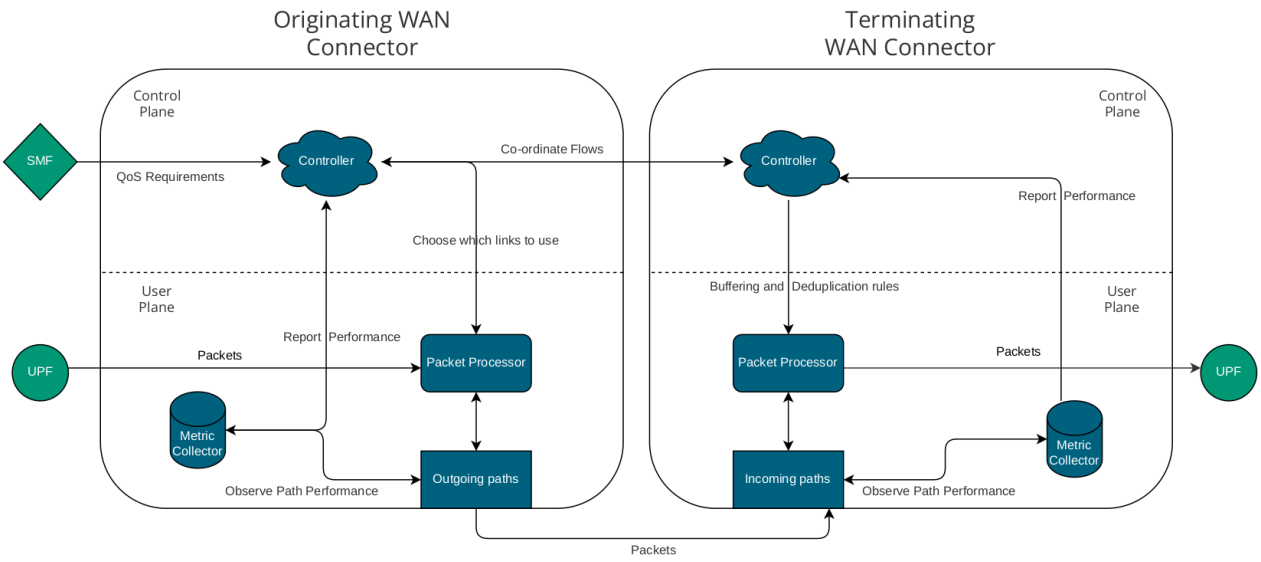


Figure 1.2: Internal Architecture of the WAN Connector

The proposed architecture of the WAN connector is shown in Figure 1.2. The connector is split into a control plane, which intelligently chooses which interfaces to forward on, and a data plane, which performs the forwarding, as well as any necessary buffering or en/decoding for forward error correction. The WAN connector also 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. Or, for

example, it may need to wait and accumulate packets before releasing them at a steady rate, in order to reduce jitter (at the cost of higher latency).

Built into this architecture is a module which collects and stores data about the different links, e.g. their latency, reliability, and jitter. When polled, this module reports the performance, and may also try to predict future performance. The link selector uses this information to make its decisions.

### 1.2.1 Evaluating Performance

In order to evaluate the success of the proposed approach, three scenarios will be set up and investigated. Each setup will consist of two WAN connectors, and some number of emulated links going between them. The first scenario will feature two links emulating regular WAN connections, one link emulating a dedicated line, and one which will emulate a satellite connection. The second scenario will be almost identical to the first but it won't feature the dedicated line, since it is expected that a leased line may be too obvious a choice for any of the traffic with strict QoS requirements. Finally, in order to further investigate these situations where there is an obviously superior candidate, the third scenario will just feature two links: a dedicated line and a WAN link.

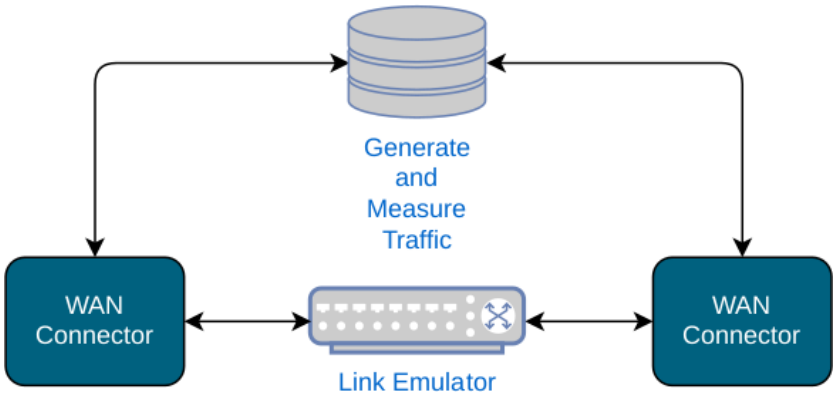


Figure 1.3: Testbed Setup

In all of these scenarios the same traffic flows will be replayed. This traffic will contain

various types of flows, with various QoS requirements. Before a new flow is started, the flow's requirements are sent to the WAN connector and it is either accepted or rejected. During the traffic replay, the delay, jitter, and reliability will be measured.

These measurements will be performed once with the ILP approach proposed earlier and once with a simple round-robin approach for selecting links, and finally compared against an offline approach, where the optimal decision is computed with complete knowledge of the future. This offline approach will serve as the baseline for optimal performance, with which the other two approaches can be compared.

All of these tests can be performed on the same testbed which will be set up as shown in Figure 1.3. The testbed architecture features a traffic generator, two WAN connectors, with a link emulator between them, and a measurement module to analyze performance. In practice the traffic generator and the test bench will be co-located on the same machine.

The first phase of the experimental work will consist of setting up the testbed. This means installing the measurement modules as well as the link emulators. The next phase will require the implementation of the WAN connector, as designed. The WAN connector will require a module for collecting per-metric links, as well as the implementation of an algorithm to select the best link or links for a given flow. In the course of designing and preparing the WAN connector there may be room for adjusting the initial plan based on what we experience during development. Next, the connector must be tested. During this phase traffic replays will be played over the testbed and evaluated. Thereafter the round robin approach and the "god routing" approach will also be evaluated. These two approaches will serve for comparison with the performance of the WAN connector. Finally, this data must be analyzed, and the thesis can reach its conclusions, with 1 or 2 weeks buffer room for final revisions.

# Chapter 2

# Background and Related Work

## 2.1 Mobile Networks and Backhaul

### 2.1.1 Backhaul in 5G

### 2.1.2 5G and Campus Networks

## 2.2 Multipathing and Multihoming

### 2.2.1 Collecting Per-Link Metrics

#### Measurement-based Metrics

In [2] 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 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 these results, it is proposed to utilize 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 an important design decision for the WAN connector, and it will be met later.

Beyond this, these metrics will also be monitored on a passive basis wherever possible. In order to measure either the time needed (for latency and jitter), or to ascertain that a packet has been lost, a response is required for each outgoing message. This may only be possible for TCP's SYN and SYN ACK messages as well as other protocols which are guaranteed to contain request-response handshakes, and thus complicates the process of passive measurement.

For classical wired links in multihomed scenarios, [8] 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 [8]. Wireless connections are usually less reliable and may experience more consistent rates of packet loss at the data link layer, however it is opaque from the perspective of the higher layers, which may only perceive it through the jitter and/or latency.

Ultimately, many of the best performing approaches for predicting packet loss, e.g. Hidden Markov Models [4, 8], 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. This thesis will also use such a model to try to predict packet loss.

### 2.2.2 How to Guarantee QoS

#### Limiting Jitter

The design of an approach which guarantees QoS is also an interesting challenge. One idea to improve jitter when backhauling across multiple links is to duplicate packets and forward them on multiple links, and have the WAN connector on the other end buffer incoming packets and release them at a constant rate. This way, in the event of a packet being lost on one link, the other link is still able to receive it and the delay caused by retransmission is avoided. The downside of this approach is that it guarantees that the latency will always be as slow as the slowest link.

#### Low Delays

For reducing latency it would appear likely that the simplest approach may be a greedy method (as in [5], in the online case) which always selects the lowest latency connection. However there is room for nuance here since the connection must not be overloaded and also because certain traffic may have very relaxed latency requirements but use up more bandwidth. This means monitoring the load on any one link will be important. Finally there are also more intelligent approaches, i.e. integer linear programming (used in [7], and used for the offline case in [5]) which find an optimal solution satisfying the given requirements.

The timescale over which to use a chosen link is also of interest. In [6] the time for which a link should be used is varied based on the predicted qualities of the link. These predictions are made based on past performance.

**Error Rates**

Reliability presents yet another challenge. However in a multihomed scenario it becomes easier to guarantee this via duplication, and/or forward error correction (FEC). For example if a packet flow requires 99% reliability this can be guaranteed by duplicating packets across two links which are both only 90% reliable. Alternatively, in such a situation, an FEC configured for 10% packet loss could be used to pre-code the packets sent across one of the links, and thus increase the reliability to the required level.

Although duplication uses a lot of bandwidth, in order to support 5G's ultra-reliable low latency (URRLC) QoS requirements, which are especially relevant for IIoT applications, it may be the only option for certain traffic flows. Forward Error Correction is an excellent protection against consistently lossy links, however it may fail to be reliable when there are concentrated bursts of dropped packets, which is a more common occurrence in packet switched networks. Either way, the effectivity of FEC in a deterministic backhaul unit is an interesting question which this thesis may also explore.

## 2.3 Determinism in Computer Networking

### 2.3.1 Quality of Service (QoS)

### 2.3.2 Deterministic Networking Specification

### 2.3.3 Traffic Shaping

# Chapter 3

# Approach

To discuss the solution implemented for this thesis requires a review of, firstly, the requirements contained in the problem statement, and, secondly, a description of both the decisions made as well as the process behind making those decisions. To that extent, this chapter 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 nature of the solution's architecture, thereafter a review of the design requirements which arise from the Deterministic Networking specification will follow, finally the precise nature of the solution's internal workings will be presented.

## 3.1 Initial Plan

At the outset of this thesis 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 kept simple and only performs the time critical

packet processing, and blindly follows the instructions of the control plane. This type of architecture is very common in modern software-based networks, e.g. OpenFlow, the Evolved Packet Core (from LTE), and the 5G core all implement this kind of split.

## 3.2 Architectural Components Required for Deterministic Multipath Backhaul in a 5G Campus Environment

Deterministic networks have been discussed in the previous section. In order to guarantee determinism the IETF DetNet working group has proposed an architecture for determinism over IP networks [**detnet-arch**]. Their specification identifies four key mechanisms for guaranteeing the determinism of a flow: 1) elimination of contention loss, 2) jitter reduction, 3) service protection, 4) explicit routes. Since the 5G campus environment may need to use the infrastructure of other operators this rules out the usage of explicit routes. A key difference between deterministic networks and 5G campus backhaul 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 core and the edge UPF's and/or WAN connectors. In such a scenario the network is less like a network and more like a client - server application with multiple paths between the client and the server.

### 3.2.1 Elimination of Contention Loss

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. For the WAN Connector therefore a traffic shaper must also be applied to the traffic on ingress, 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 a choice must be made based on the existing solutions. The traffic control subsystem in the linux kernel (TC) provides several implementations of different

algorithms for both rate limiting and traffic shaping. Some of these algorithms have already been discussed in the previous section. For the purposes of this solution several algorithms were considered feasible, namely HTB, HFSC, taprio, CAKE, and 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 necessarily a hindrance yet, 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 caps bandwidth), the greedy flows can cause other flows to experience higher latency. The HFSC algorithm improves on this by offering both bandwidth and delay guarantees. However this makes it more difficult to configure.

Another feasible option available in linux TC would be the time aware priority queuing (taprio). This queuing discipline provides scheduled gates for specific traffic classes. It could be used to schedule different flows, and/or different priorities, and thus provide a form of prioritization, and, to an extent, traffic shaping. Flows can be assigned different windows (or assigned windows based on their priority) and congestion due to a greedy flow is avoided. However one issue is that taprio fails to provide fair queuing, and thus would need to be configured and possibly adapted as new flows are added and removed, depending on their priority. Greedy flows of the same priority would have to fight 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 WLAN networks, this may prove to be loosely analogous to a 5G campus network (consider the User Equipment as the Stations, and the Base Station as the Access Point). CAKE also allows the use of priority bins, 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 [**eBPF**] programs. Specifically within a 5G context, this could allow one to attach an eBPF program to allow 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 flow, and thus none of these algorithms would work as expected on the flows being tunneled through GTP. 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 CHOKe qdisc, which does perform queue management, however fails to provide fair queuing.

### 3.2.2 Jitter Reduction and Latency Guarantees

In the Deterministic Networking 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. To this extent the PTP protocol may serve well, especially since there is an existing implementation, the linuxptp project, which extends it to work over IP networks. As for the time of execution fields, it may be possible to place these in a GTP header, or to combine them with timestamping.

Latency guarantees are not mentioned within the specification but they are important for the 5G campus setting. Specifically when backhaul options may include satellite links it becomes important to consider latencies. Additionally, in a multipath setting, latency can be a useful criteria for selecting between links, and guaranteeing latency becomes easier to do when it is possible to switch links should one of them start to experience greater latency than before.

### 3.2.3 Service Protection

In order to protect against equipment failure, may be recommendable to perform packet duplication and/or encoding. The specification speaks of both duplication of flows, and network coding [**network-coding**]. This can be a clever solution, however like other parts of the Deterministic Networking specifications, Network Coding requires control over and co-ordination between all the nodes in the network. However to guard against equipment failure and/or packet loss, duplication does provide an option. So too does forward error correction (FEC). Many FEC schemes work on a continuous stream of bytes, providing correction for bit errors, as opposed to correcting the loss of entire packets. There are schemes which address this, fountain codes - such as Raptor [**raptor**], but unfortunately their ecosystem is not as developed and because of the complexity involved in implementing them efficiently there are relatively few freely available projects or libraries which can be used to encode packets using such a scheme. For reference, [1] mainaints a list of C
C++ FEC libraries. Even if it were more feasible to integrate FEC into the backhaul solution it bears questioning how much benefit it brings. FEC is very powerful in situations with consistently lossy links, however internet links tend t experience loss in bursts, as opposed to consistently of the same degree. To this extent it may make more sense to duplicate those flows which require a high degree of reliability. However flow duplication means a packet ordering and elimination function will be required, to eliminate those packets which arrive twice, and to re-order other ones.

TODO!! : detnet POF algorithm

### 3.2.4 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

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

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. For the WAN Connector's scenario routing considerations are not made, its purpose is to backhaul the traffic from the RAN / Edge to the Core, where the network's own internet gateway can 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 destined to the same source. In multipathing the same link may lead to multiple paths to *different* destinations. Multipathing over the internet requires complex data collection about all the routers in between the source and the destination, for example [**multipath**].

The goal of this thesis is of course to use the link and/or path selection to provide determinism for the flows being backhauled. To this extent then, the link selection algorithm must attempt to forward flows over paths which can provide at worst the maximum allowed latency and jitter, as well as meeting the minimum reliability. Here it is worth noting that while duplication cannot really be used to guarantee latency, it can be used to improve reliability since a flow which is duplicated across two paths is far less likely to experience packet loss for the same exact packet on both paths.

### 3.2.5 Summary of Requirements

Looking back on the previous subsections, the following points (in no particular order of priority) were 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. The ways in which these can be addressed, or quite simply the way in which they were implemented, will be discussed in the following section

## 3.3 Overview of the WAN Connector's Features and Components

### 3.3.1 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. The optimization problem can be posed as such: "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:

$$\text{Minimize } \sum_{i=1}^{P} w(x_i) \tag{3.1}$$

$$\text{Where, } d(i) * x_i \leq D \tag{3.2}$$

$$j(i) * x_i \leq J \tag{3.3}$$

$$1 - \prod_{i=1}^{P} (1 - r(i) * x_i) \geq R \tag{3.4}$$

$$\text{for } x_i \in \{0, 1\} \tag{3.5}$$

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 [2], however there is room here to use more advanced metrics to predict the future link quality and thus perform preemptive path switching. 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 [**tsp-np-hard**]. However this hardness arises primarily because of line 3.1, 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 [1] have shown that multihomed approaches experience diminishing returns after more than 4 links, attempting to brute force the solution by limiting the number of paths 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 paths. This can be expanded on by limiting duplication to only take place over disjoint interfaces. No duplication on the same outgoing interface. 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 from paths to include from a set of paths using the same outgoing link, only the path with the greatest reliability to weight ratio is taken into consideration.

### 3.3.2 Packet Ordering and De-Duplication 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. For the packet ordering algorithm the Deterministic Networking Working group provides both a basic and an advanced algorithm in their specification [**detnet-pof**]. This thesis uses the basic algorithm.

### 3.3.3 Internal Architecture

#### Control - Data Plane Communication

A protocol is desired which can provide reliable communication over multiple paths in order to communicate between the control plane, where the statistics are collected, flows are added or removed, and the multipath decisions are made, and the data plane. 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 two feasible options - either multipath TCP or SCTP. SCTP supports multihoming and intelligent failover to the multihomed link, 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 will be GTP [**GTP-spec**] version 1. 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 atomic clock is used in this implementation because it does not have leap seconds, and so it is a monotonic function, which is an important guarantee for time sensitive applications. Normally, in linux on 64 bit 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 as an 8 bytes value, thus wrapping around the interval $[0, 255]$. 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 usually 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 self-balancing binary tree, such as a red-black tree [**red-black-trees**], 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 overhead may not be worth the gain in performance.

It is important to make a good choice of a hash function. Ideally it should provide less Since the hashes used are only for the purpose of lookups it makes no sense to pick a hashing algorithm designed for us 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 hash algorithms designed for

looking up values. 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, in general it is usually recommended to use prime number. In this implementation, the MurmurHash algorithm was chosen due to its strong performance for lookup-based hashing. Specifically, the MurmurHash3 [3] 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.

### 3.3.4 Recap and Overview

TODO: figure

# Bibliography


[1] 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.

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

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

[4] 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.

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

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

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

[8] 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.

[9] 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.