Master thesis
Computing Science

Radboud University & SIDN Labs

Achieving Application-Level Requirement-Based Path Selection Within SCION

Author:
Alessandra van Veen
s4683382

First supervisor/assessor:
prof. dr. ir. H.P.E. Vranken
harald.vranken@ru.nl

Supervisors SIDN Labs:
dr. ing. R. Koning
ralph.koning@sidn.nl

ir. C.J.T.M. Schutijser
caspar.schutijser@sidn.nl

Second assessor:
dr. ir. E. Poll
e.poll@cs.ru.nl

September 29, 2024

Abstract

In today's world, the Internet has become quite essential to daily life, yet there are also many security problems concerning the Internet. Some important internet protocols, such as routing, were never built with security in mind. The most important routing protocol is named the Border Gateway Protocol (BGP), which lacks security features and provides no guarantee for the integrity and authenticity of the messages sent using BGP. Over the years, there have been many improvements to BGP, like BGPsec and RPKI, which only solve some of BGP's issues. At the same time, researchers have been looking into alternative internet architectures. One such architecture is SCION (Scalability, Control, and Isolation On Next-generation networks), which offers an alternative to BGP using a new routing protocol. We investigated SCION's ability on an application-level for an end-host to choose their paths through the network for packets to take, which is a form of path-aware networking (PAN). This provides greater security for network communications as it increases transparency for endpoints and allows endpoints to avoid potentially insecure paths.

This research aims to achieve application-level requirement-based path selection on the SCION network, where we focused on security requirements such as geolocation and router manufacturer. Specifically, we aim to answer the question *How can we achieve application-level requirement-based path selection on the SCION network?* By achieving this, we can increase the security of our network communications as we can then avoid non-trusted ASes (autonomous systems) or ASes with insecure software versions. To answer our research question, we first answered three sub-questions. Our first sub-question was *how can we extract different properties for possible paths?*. We selected a medical use case, specifically remote surgery, for its interest security properties. We then created an application design for the use case where we specified the path property extraction. We then extended our application design to answer our second sub-question *how can we find the best available path?*. We also created a prototype to prove these different features are possible within SCION. Lastly, we used this prototype to measure the impact on the path selection efficiency and to answer our last sub-question *what is the impact on the efficiency using our method in comparison to path selection without requirements?*

Our research shows that PAN within SCION has much potential and almost all aspects we wish to achieve are possible. However, we also highlight one significant limitation of PAN within SCION, namely that the most important security properties, like the router brand, cannot be verified natively as SCION lacks support for this. In a scenario where security is the

most important aspect, the lack of information verification is an important problem that requires future research.

Contents

List of Acronyms

(D)DoS	(Distributed) Denial of Service.
AP	Attachment Point.
AS	Autonomous System.
BGP	Border Gateway Protocol.
CA	Certificate Authority.
CIA	Confidentiality, integrity, and availability.
ELECTRE	ÉLimination Et Choix Traduisant la REalité.
EPIC	Every Packet Is Checked.
FABRID	Flexible Attestation-Based Routing for Inter-Domain Networks.
IP	Internet Protocol.
ISD	Isolation Domain.
MAC	Message Authentication Code.
MCDM	Multiple-Criteria Decision Making.
MTU	Maximum transmission unit.
PAN	Path-aware networking.
PCB	Path-segment construction beacon.
PROMETHEE	Preference Ranking Organization Method for Enrichment Evaluation.

RIM	Reference Ideal Method.
RPKI	Resource Public Key Infrastructure.
SCION	Scalability, Control, and Isolation On Next-Generation Networks.
TOPSIS	Technique for Order Preference by Similarity to Ideal Solution.
TRC	Trust Root Configuration.
WPM	Weighted Product Mode.

Chapter 1

Introduction

Today's internet is an essential part of our society, but it is far from perfect. It was initially not built with security in mind, but rather as an open network to freely exchange data. As the Internet became more widely accessible, there was a need to improve routing protocols. Eventually, there was BGP (Border Gateway Protocol) which is still one of the most important routing protocols to this day. However, much like the early days of the internet, BGP relies on trust, and the systems that utilize BGP implicitly trust all the routes shared with them. This led to a variety of issues over the years. A famous example is when in 2008 Pakistan Telecom accidentally brought down YouTube for several hours for the entire world [1]. Pakistan Telecom announced a prefix that redirected YouTube traffic to Pakistan Telecom. Two hours later, YouTube announced the same prefix, but as the shortest path is preferred, most YouTube traffic was still redirected to Pakistan Telecom. This case of prefix hijacking was fully solved another hour later, but it made the headlines all around the world [2].

This example is just one of many problems with today's internet. Over the years, besides improving BGP [3][4], there has been much research into new internet architectures that help resolve the issues of the Internet. One such architecture is SCION (Scalability, Control, and Isolation On Next-generation networks) which aims to tackle the current Internet's problems with security and availability [5]. SCION offers an alternative to BGP [6] by implementing a new routing protocol between ASes (autonomous systems). One important difference between BGP and SCION is who makes the routing decisions. In BGP, the AS decides which path to take, often based on what is the most efficient. In SCION, this decision is made by the end host.

This routing control is what makes SCION a path-aware network. It enables a lot of interesting opportunities. For example, what if you only want to send your packets through the European Union, because you are sending very sensitive data? What if you want to send data over multiple paths at once, in case one path fails? Some of these aspects have been

researched within SCION [7] [8] [9] [10] and even implemented, but there is still much more that can be investigated. There is still research that can be done into routing based on requirements, and more specifically security requirements. For example, what if you have a server and a user application and wish to exchange data securely? There may be a distrust towards certain router brands and towards countries, so the two end hosts may wish to avoid these. Furthermore, there may be a requirement that a data transfer may not be interrupted. There are many aspects to consider, and limited research has been done yet to fully explore this.

Our aim is to investigate how we can achieve a requirement-based path-aware networking (PAN) on an application level, where we focus on security-based requirements such as geolocation and router manufacturer. Investigating PAN on an application level allows us to see what is possible within SCION itself, without building further on top of it. Our research question is as follows:
How can we achieve application-level requirement-based path selection on the SCION network?
Our sub-questions are as follows:

1. How can we extract different properties for possible paths?
   To establish a requirement-based PAN, we need to be able to extract different properties so we can consider if the properties fulfil the requirements. Not all properties for a path are provided, or easily measured.

2. How can we find the best available path?
   It may be possible that there is no path available that meets all requirements. As we focus on security-based requirements, an end host may wish to instead have a path that meets the requirements as best as possible, even if not fully, to ensure the highest possible security.

3. What is the impact on the efficiency using our method in comparison to path selection without requirements?
   Our application has several steps before it can send data, such as property extraction and selecting the best path. This increases the time it takes for a packet to be made, a path to be chosen, and the packet to then be sent. As latency is still an important aspect of networking, we want to investigate what the impact is in milliseconds.

To answer these questions, we will first select a use case with various security requirements. The use case allows us to explore what kind of requirements might be needed for a PAN application. We will use a literature study to create a theoretical design of an application for the use case. For this design, we choose several path properties to show how these can be extracted within SCION. The chosen properties will be relevant to the use

case, but will be varied enough to be applicable to other use cases. We also need to choose the best available path. The specific method is chosen specifically for our use case, but it can be applied to other use cases. After we made our design, we will also implement several features in a proof of concept application as evidence that our idea is possible within SCION. This also allows us to measure the efficiency of the path selection.

This thesis is structured as follows; Chapter 2 provides a background on how SCION works with a high-level overview and then some more detail into two main aspects of SCION. We will also describe the related work of our research in this chapter. In Chapter 3, we motivate a remote surgery use case. Then we describe the requirements the application has based on this use case and the scenarios it should be able to deal with. In Chapter 4, we provide a high-level overview of our application design and then we describe the technical details, which help us answer our sub-questions. In Chapter 5, we evaluate our results. In Chapter 6 we if and how we met our requirements, and the limitations of our research. Finally, in Chapter 7 we will answer our research question and make recommendations for future work.

Chapter 2

Background

In this chapter, we will provide the background on what SCION (Scalability, Control, and Isolation On Next-generation networks) is and how it works. First, we will give a high-level overview of SCION. Next, we will explain two of SCION's main components that are both essential to routing; the control plane and the data plane. Lastly, we will show the related work that our research is based on.

2.1 High-level Overview of SCION

2.1.1 Goals of SCION

In essence, SCION is an internet architecture of which a new inter-domain routing protocol is an important part. The routing protocol is built from the ground up as opposed to improving any present day protocols like the Border Gateway Protocol (BGP). SCION has been made with several goals in mind [11, Ch. 1].

The first of six goals is **availability in the presence of adversaries**. This means that as long as there exists a path between end hosts, it should be possible to discover this path and guarantee there is some bandwidth available between the end hosts. This should be possible even with potential attackers that, for example, try to hijack the route or delay packets.

Goal two is that there should be **transparency and control** for the forwarding paths and the trust roots. This allows for verifiable path control. In turn, this enables greater protection against certain network attacks, like DoS and DDoS, and also allows for aspects such as geofencing and multi-path communication.

The third goal of SCION is **efficiency and scalability**. SCION should at least be as efficient as present day IP forwarding in terms of latency and throughput. Furthermore, compared to the current Internet, SCION should be more scalable with respect to BGP and the forwarding table size.

The fourth goal of SCION is **extensibility and algorithm agility**. With this, the aim is to future-proof the architecture better. The extensibility ensures the codebase can easily be expanded with new features. Algorithm agility ensures that it is easy to switch from one cryptographic algorithm to another.

The fifth goal is **deployability**, meaning that migrating to SCION should not be complex and the costs should be kept minimal for ISPs especially.

The final goal is **formal verification**. The security and the availability of the architecture should also be ensured by formal proofs.

2.1.2 Architecture of SCION

SCION, just like the internet we know, consists of many different ASes, and SCION's protocol facilitates the communication between the ASes themselves. ASes are big networks consisting of many routers where each AS determines their own routing policies for any traffic through the AS. In SCION, ASes are grouped together in something called isolation domains (ISDs), often based on geographical location and/or function. For example, an ISD can consist of ASes in the Netherlands, and another ISD which consists of European Healthcare ASes. This also means that one AS can be a part of several ISDs. ISDs are represented by a number from 0 to 65535, whereas ASes are represented by a 16-bit colon-separated hexadecimal encoding with the leading zeros omitted.

To understand why ISDs are used, we will first explain how they are organised. Each ISD has a trust root configuration (TRC), which is a collection of signed certificates. These certificates define the roots of trust of the ISD and can also contain policies, which can for example define when a TRC is valid. The TRC also defines which roles ASes have. All ASes facilitate communication, however, they can have additional roles:

- **Core ASes** are an important part of the ISD. Each ISD generally has a few core ASes that are on the top of the routing domain and are responsible for connecting the ASes within the ISD with ASes of other ISDs.
- **Certification Authorities (CAs)** issue AS certificates to other ASes and are also able to issue certificates to themselves.
- **Voting ASes** have the ability to vote to accept changes to TRCs.
- **Authoritative ASes** have the latest TRCs of the ISD and are also able to announce changes to the TRC and thus initiate the voting.

The structure of the ISD ensures transparent trust relationships between ASes. It also helps reduce the chance of external attacks as the routing process is isolated within the ISD itself to limit any external influences.

To be able to communicate with each other, ASes have links with other ASes. In SCION, we define three different types of links: core links, parent-child links, and peering links. Core links are the link between two core ASes within an ISD. A parent-child link is a link between a non-core AS and another AS, which can be a core AS, within the same ISD. Peering links are also links between a non-core AS and another AS, but they signify a peering relationship. The peering link is only available to use by ASes which are a child of any of the two linked ASes. This does not only mean direct children, but also the children of the children and so forth.

In Fig. 2.1 we show how a SCION architecture can be structured. The example consists of two ISDs which each consist of two core ASes and several child ASes. AS 1-6 and AS 1-7 also have a peering link between them.

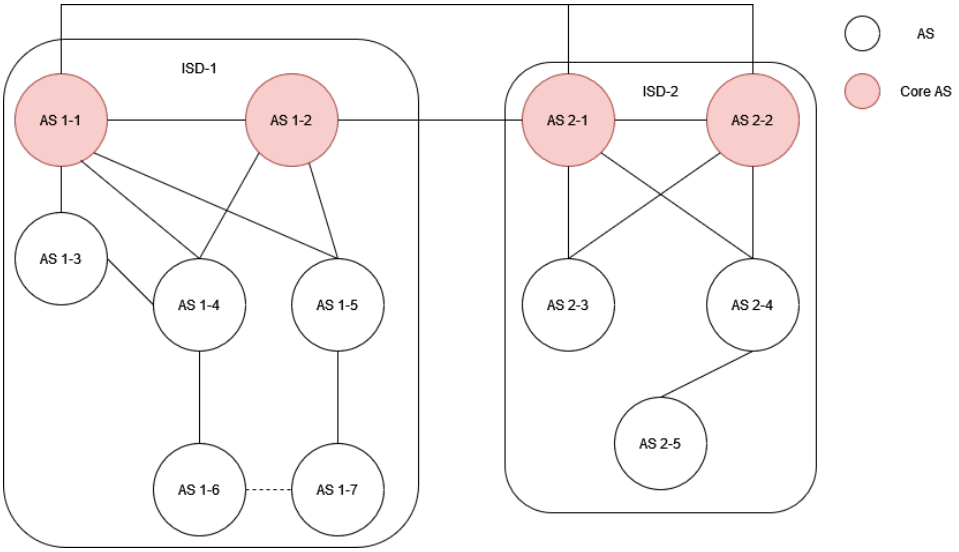


Figure 2.1: A SCION architecture consisting of two ISDs and several ASes

2.1.3 How does SCION work?

Now that we established the architecture of SCION, we will explain how routing within SCION works. Before a router can actually send packets to a router in another AS, the AS it is in must first discover the possible paths to that AS. For example, a router in AS 1-6 wants to exchange data with a router in AS 1-3.

The discovery of paths happens within the control plane of SCION. An AS discovers paths as follows:

- The beaconing process, or path exploration, is the process where an AS creates a path-segment construction beacon (PCB) through a beacon service. This contains information about the AS. It will eventually

send this PCB to its child links, which will add their own information and then send it on to their child links until there are no child links left. Each AS also saves each PCB it received.

- Path-segment registration is a step that is part of the beaconing process. Before an AS forwards a PCB, it will first select the best PCBs it has saved according to some defined policy. The AS will use its beacon service to forward the selected PCBs and the path service to register the PCB to itself. With this, an AS now has a path towards at least one core AS.

- The last step of the control plane is path look up and construction. In this step, AS 1-6 would try to find a path to AS 1-3. AS 1-6 now knows how to reach core AS 1-2 and core AS 1-1. It can query these ASes for possible paths from the core to AS 1-3. Both ASes can reply with several path segments. With these segments, AS 1-6 can choose to combine one of those segments with a segment to reach the core AS and together those segments form a complete path.

Once a path has been chosen, the data plane of SCION actually facilitates the forwarding of packets along the selected path. It does this by first creating a SCION packet, similar to an IP packet, and then each AS will forward the packet using the path defined by the sender AS.

In Section 2.2 and 2.3 we will go into further detail about these processes, like beaconing, path registration, and the creation of the end-to-end forwarding paths.

2.2 Control Plane

The role of the control plane is to discover path segments and to make them available to end hosts. There are several processes part of this, which we briefly touched upon in the previous section. Especially important for our work are the beacon extensions which contain metadata on properties that we rely on in Chapter 4.

2.2.1 Path Exploration (Beaconing)

The first process is path exploration, which can also be called beaconing. First, the AS initiating the process will create PCBs through its beacon service. Every AS has set its own propagation period that indicates when and how often it should do the beaconing. PCBs represent a single path segment which can later be used to construct an end-to-end forwarding path. Each PCB is composed of an info field and AS entries. The info field provides the basic information on the PCB, specifically the set flags to indicate the type and direction of the constructed path, a random value for

MAC-chaining, and a timestamp to indicate when the propagation started. Each AS entry consists of a signed component, a signature, and an unsigned component. The signed component provides information on the AS itself like the ISD-AS number of the entry, the ISD-AS number of the AS to which the PCB will be forwarded, potential peering links, signed beacon extensions, and more. The unsigned component is optional and may contain unsigned beacon extensions. Beacon extensions are optional entries that allow the PCB to convey metadata, such as properties like latency, bandwidth, and geolocation.

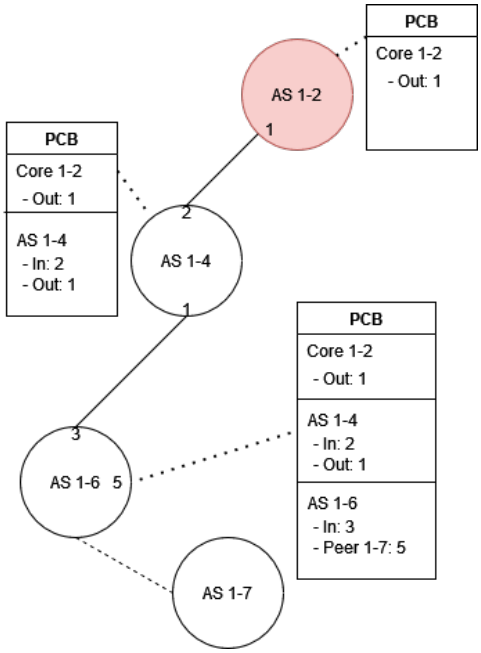


Figure 2.2: A single PCB going from AS 1-2 to AS 1-6, where each AS entry describes the router interface the PCB arrives from and exits from.

After the PCB has been signed, the beacon service passes along the PCB to its border router. This PCB is then propagated to either the AS's parent-child links, which we call intra-ISD beaconing, which is also shown in Fig. 2.2, and is to create up-segments and down-segments or to other core ASes, which is called core beaconing and is to create core segments. Each AS that receives a PCB will verify the structure of the PCB and its signatures. If everything is correct, the AS's beacon service will add the PCB to its local database. Once it is time for propagation, set by the AS, it will add an AS entry of itself to the previous PCB and propagate this to its relevant links.

2.2.2 Path-Segment Registration

After the path exploration process, path segments can now be created from PCBs. The process of this differs between intra-ISD segment registration and core path-segment registration.

First, we will describe intra-ISD segment registration. This process is necessary to enable ASes to communicate with the core ASes and other ASes within the ISD. Every AS has a self-determined registration period where the beacon service selects up-segments and down-segments. Up-segments are path segments that allow for communication with the core ASes within the ISD. Down-segments allow other entities to reach the AS. Every AS has a selection policy for up-segments and a policy for down-segments so that it can decide which traffic is routed and how. The PCBs from the local database are examined and the selection policies are used to determine which of these might serve as up-segments and which as down-segments. If a PCB gets selected, an AS entry will be added which specifies that the path ends at this AS. Any peering links are also added to the PCB. This is then signed and the signature is added to the PCB. All up-segments will be registered with the path service of the AS, and the down-segments will be forwarded to the path service of the core AS that sent the PCB originally.

Core path-segment registration works similarly to the intra-ISD version. It will select some PCBs based on its selection policy. Then it will add a new AS entry to every selected PCB and sign these. Instead of up- or down-segments, these are called core-segments. Lastly, the core-segments are only registered with its path service.

2.2.3 Path Lookup and Construction

When a source host wants to connect to another end host, it needs a full path. This is where path lookup plays an important role. The goal of path lookup is to find suitable path segments from between two end hosts so that the segments can be combined into a path of at most three segments. These three path-segments are an up-segment so the core of the ISD be reached, a core-segment to reach the ISD of the AS that has to be reached, and finally, a down-segment to reach the AS. We also show this in Fig. 2.3 if AS 1-6 were to query AS 1-1 and AS 1-2 for path segments.

To start the process, the source host will query the path service of its AS for the relevant segments. The AS stores the up-segments in its database and is easily able to return these. Core- and down-segments may be stored in its cache, and if this is the case, it can return these. If not, the path service must query the path service of a core AS for core-segments to the ISD of the destination AS. With these segments, they can query the core AS of the destination ISD to retrieve the relevant down-segments. Once it has collected all the path-segments, they will be returned to the source host.

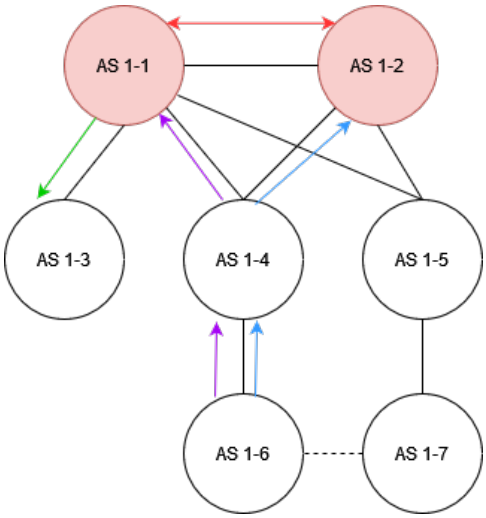


Figure 2.3: An example of up-segments (purple and blue), a core segment (red), and a down-segment (green) for AS 1-6 to reach AS 1-3.

With these path-segments, the source host can combine these segments to create one or more paths to the end host.

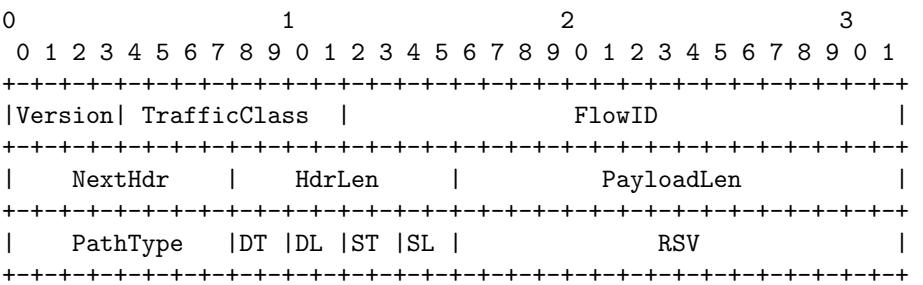
Now at most one of each type of segment can be chosen to be combined into a path. Each AS will have its own policy on how to select the best path, for example, based on which path provides the lowest latency or highest bandwidth.

2.3 Data Plane

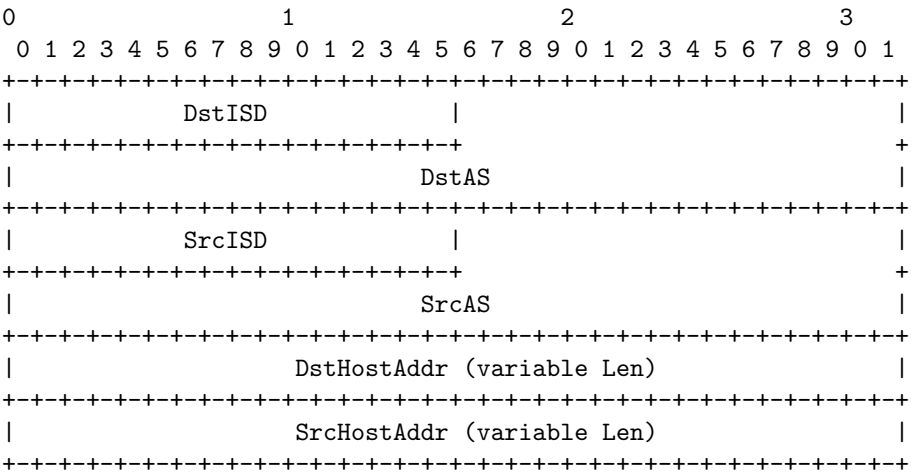
The main purpose of the data plane is to ensure that the packets are forwarded along the selected path.

2.3.1 The SCION Packet

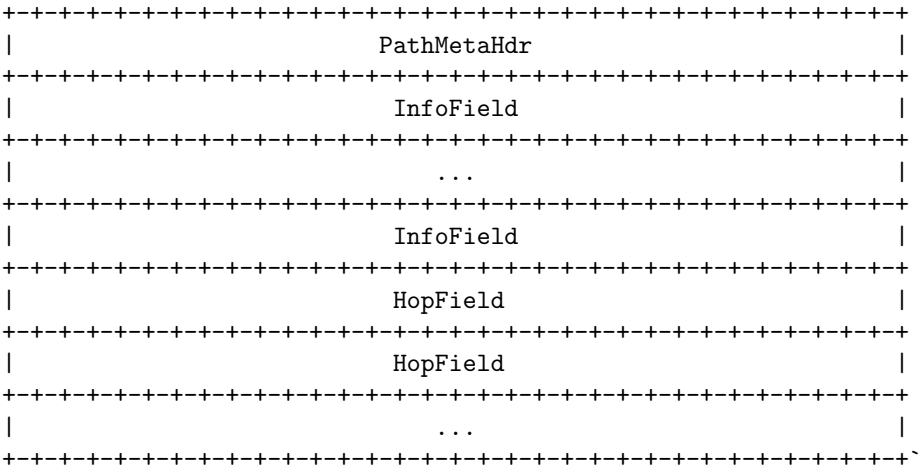
Once a path has been constructed, a SCION packet can be created. These are similar to modern internet packets, but these packets will have a SCION header on the network layer. The SCION header consists of several parts. The first is the common header, which contains meta information such as version number, payload and header length, and flags, which looks as follows [12]:



The next header is the address header, which contains information like the ISD, the AS, and the end-host addresses of both source and destination [12]:



The path header contains the AS-level forwarding path based on the path that was constructed during the different processes in the control plane [12]:



This also means that SCION's forwarding decisions are based on this information, as opposed to local information from routers themselves. While

in our application design we do not pay special attention to the headers, they could be used for future work on path verification. Finally, there is also a possibility of an extension header which has various options regarding hop-by-hop and end-to-end. This header is optional.

2.3.2 Packet Forwarding

Once a packet has been initialized, the egress border router will forward the packet to the next AS specified in the path. Every AS along the path will first parse and validate the SCION header. Validating also includes verifying whether or not the path contained in the header was authorized by the AS. Once validated and parsed, some forwarding within the AS itself will happen so the packet goes from the correct ingress border router to the correct egress border router. Eventually, the packet reaches the end host. The end host is able to use the path specified in the SCION header to send a reply back, as the path can simply be reversed. However, an end host is also able to perform its own lookup and find a new path.

2.4 Related Work

Over the years, there have been a few papers that looked at the application of PAN with SCION. In 2021, Krüger and Hausheer introduced PANAPI [9] in which they propose the design of a networking API for PAN in SCION and explore what capabilities such an API should have. Specifically, they performed an initial investigation into property collection, property processing, and path selection and exploration and made some proposals into how they could be done. The aspects the paper looks at are relevant to our work, however, there are some gaps we aim to fill. The properties mentioned for property collection are all properties that can be expressed in numbers and easily be compared in that way. There are no properties like the router manufacturer or geolocation. Furthermore, we would also like to improve upon the path selection by allowing the setting of requirements and choosing the best path based on more than just latency.

Also in 2021, John et al. [7] introduced a communication gateway for industrial applications called Linc. Linc utilizes SCION's path awareness for a path selection that ensures all traffic only crosses trusted network infrastructure, determined by blacklists and whitelists. They also introduce three different failure and redundancy modes, which help increase recovery time if a path fails. These modes are interesting features to ensure availability and can be a part of our final design. Similarly to PANAPI, Linc has a knowledge gap when it comes to best path selection based on several requirements and non-quantifiable properties.

In the following year, Davidson et al. [13] introduced a browser plugin for Brave that introduced PAN in SCION. It also investigates which layer should

make the path decisions and on the basis of which properties. They looked at three possible possibilities: OS, Application, or User. For the browser plugin, they use SCION's path policy language where paths can be sorted and selected based on criteria like bandwidth and latency. Furthermore, they implemented geofencing on an ISD level. However, it should be noted that in version 0.0.3, which is the most recent version as of writing, the user can only control geofencing and decide between a strict or not strict SCION mode. There is no implementation to decide on what criteria a path should be chosen beyond geolocation.

In 2023, FABRID or Flexible Attestation-Based Routing for Inter-Domain Networks [8] introduced a system built on top of SCION that enables inter-domain path selection based on attested router properties. The two most important contributions that are also relevant to our work are their extensible policy language to describe those properties and an enhanced path selection. One important issue they encountered is the lack of being able to verify unquantifiable properties as an AS may not conform to its announced properties. While FABRID is similar to what we wish to achieve, it differs in the fact it is built on top of SCION rather than an application running on SCION. FABRID has to be deployed on each SCION router the end host may wish to know the properties of. Our application design does not have this limitation, and rather we wish to look at what is possible with the current version of SCION.

UPIN [10] created a method to perform measurements on paths to measure latency, bandwidth, and packet loss to eventually achieve user-driven path control. One conclusion they made is that in SCION, latency is mostly affected by the physical distance between ASes, and not the number of hops travelled. A point of future research was to use this data for a path recommendation feature.

In table 2.1, we show a comparison between the different papers and their features. We also included our work to show what features we would like to include in our application design. These features are chosen to show the capabilities of SCION on an application-level.

	Our work	PANAPI	Linc	Browser Plugin	FABRID	UPIN
Active property collection	✓	✓			✓	✓
Path selection based on quantifiable properties	✓	✓		✓	✓	
Path selection based on geolocation	✓		✓	✓	✓	
Path selection based on non-quantifiable properties except geolocation	✓				✓	
Multi-path mode	✓		✓			

Table 2.1: A comparison of the discussed papers and their features compared to our paper and its features

Chapter 3

Use Case: Remote Surgery on the SCION Infrastructure

In this chapter, we will discuss our use case, specifically about remote surgery. We do this by first explaining our motivation behind choosing this use case. Afterwards, we explain the requirements and then we establish several scenarios that our application must be able to handle.

3.1 Motivation

To investigate how we can achieve a path-aware application with requirement-based path selection, we can utilize a use case to explore what sort of requirements might be needed and to make considerations regarding path-aware applications. The use case should have interesting security requirements, so we can test the limitations of SCION's path control as it is. Furthermore, we would also like to utilize some of SCION's other security related properties (see Section 2.1.1), such as *availability in the presence of adversaries* and *transparency and control*. We would also like the use case to be something relevant, meaning it is an area of active research.

There are several fields of studies that could be suitable such as banking, healthcare, and national defense. All three of those have a high need for security, especially the three CIA principles: confidentiality, integrity, and availability. We decided upon a remote surgery use case. Remote surgery is the ability for a doctor to perform surgery while not physically present in the same location as the patient, for example through the use of a remote-controlled robot. Remote surgery is a relevant topic within healthcare research [14][15], especially as human-controlled robots become more advanced and precise. As the word *remote* implies, network security is an important aspect of remote surgery. Remote surgery consists of a constant network connection between a controller and a robot, which may not be disrupted or hijacked. If that does happen, the robot can make a mistake that can

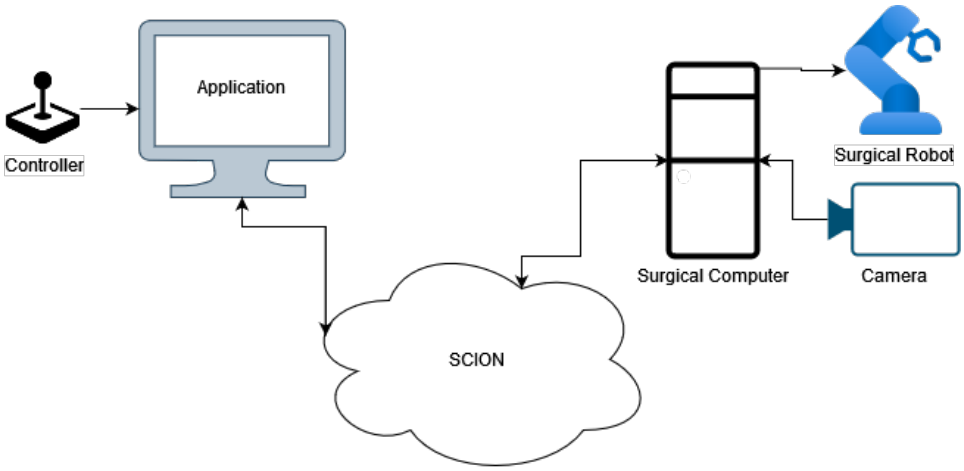


Figure 3.1: Abstract diagram of the setup of a remote surgery

have severe consequences for the person being operated on. With SCION's path selection, the controller is able to decide which path to take. That means that for our remote surgery use case, we can set the requirements in such a way that we can improve upon connection availability and security. For example, with a requirement to avoid routers with a specific firmware version. For the use case, we will focus on the literature and a prototype to imitate the use case, however, we will not perform remote surgery.

3.2 Requirements

A remote surgery application and its network must fulfil several requirements. The remote surgery application must consist of two different clients: the surgeon side and the robot side. The surgeon's client needs to receive high-quality video imaging from the robot's client and the surgeon's client must be able to send different commands to the robot's client. The robot's client will then use this to further communicate with the robot. In Fig. 3.1 we show a diagram of how this would look like.

A remote surgery application and its network must fulfil several requirements. Some of these requirements will be relevant for why SCION is appropriate and some are more relevant for the application design itself. In healthcare, the principles of confidentiality, integrity, and availability (CIA) are all important [16]. Availability especially is a hard requirement, as a loss of availability while surgery goes on can lead to severe consequences. Integrity plays a similarly important role. When someone operates a surgical robot from a distance, it is important the control of the robot is not tampered with in some way. Confidentiality is important as healthcare data is very sensitive data. SCION itself already guarantees many aspects of the

CIA, however, we would like to use path selection to further improve the availability.

The healthcare network must be resilient against several attacks [16][15] that violate any of the three principles. One important attack is the (Distributed) Denial of Service ((D)DoS) [14], which threatens the availability of connection between the controller and the surgical robot. Some attacks that threaten the integrity of the data are (IP) spoofing [16] or a masquerading attack[15]. Confidentiality could be threatened through eavesdropping [15]. SCION already provides some protection against these attacks by design, however, path selection allows us to avoid ASes which have an increased chance of being attacked or causing an attack.

Beyond attacks, it is also important that the connection has a low latency, a high bandwidth, and ultra-high reliability [17]. Remote surgery has a requirement of a 200ms end-to-end latency while keeping in mind robotic systems have an inherent latency of almost 100ms. Any actions done by the controller should be nearly instant. Surgical skill deterioration is noticed at a latency of 300ms and above [18], meaning that if the latency is 300ms and above a surgeon may not be able to work as accurately. The bandwidth should be high to allow for high-quality video to be sent through. Ideally, the bandwidth is around 1 Gbps download and 250 Mbps upload [18] to allow for this. However, some loss of video quality is allowed if it ensures a higher latency. Lastly, reliability should be high to ensure the proper availability of the network and to ensure all data arrives. It also means that the connection may not be disrupted at any point.

Our design should also take into account the goals of SCION and it should uphold the same principles.

3.3 Scenarios

In the perfect world, there is always a path available that meets the requirements, that is always available, and no other hiccups happen along the way. In reality, this is of course not the case. We will describe several artificial scenarios that may happen and that allow us to explore what SCION can do, and where it is limited.

3.3.1 Basic Flow

To establish a base, we will first describe the theoretical setup and connection requirement and afterwards, we will describe the basic flow.

Fig. 3.2 is the SCION structure we will use to help describe our different scenarios. The structure consists of 4 ISDs. The AS that is utilized by the surgeon's application is AS 1-7 in ISD-1. The AS that has to be reached, and where the robot's application is hosted, is AS 4-2 in ISD-4.

There are five paths made available to AS 1-7 to reach AS 4-2:

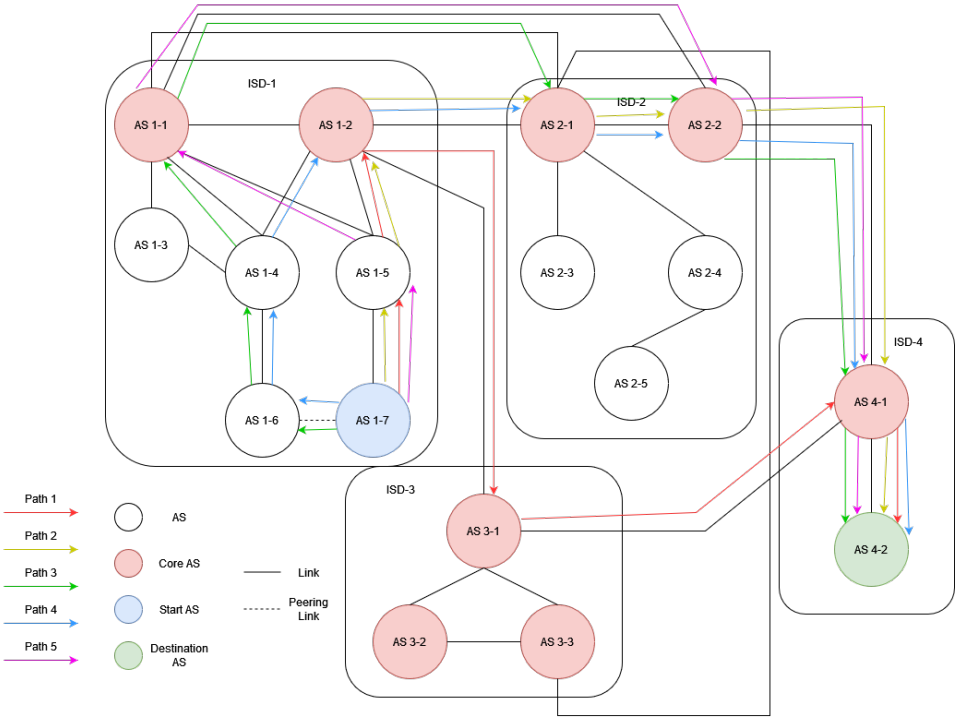


Figure 3.2: SCION structure of our use case where each coloured arrow represents a possible path

Path 1: AS 1-7 → AS 1-5 → AS 1-2 → AS 3-1 → AS 4-1 → AS 4-2
Latency: 100 ms, Bandwidth Up/Down: 600/1200 Mbps
Minimal Router Firmware Version: 2.1.1

Path 2: AS 1-7 → AS 1-5 → AS 1-2 → AS 2-1 → AS 2-2 → AS 4-1 → AS 4-2
Latency: 120 ms, Bandwidth Up/Down: 500/1100 Mbps
Minimal Router Firmware Version: 2.1.1

Path 3: AS 1-7 → AS 1-6 → AS 1-4 → AS 1-1 → AS 2-1 → AS 2-2 → AS 4-1 → AS 4-2
Latency: 170 ms, Bandwidth Up/Down: 400/800 Mbps
Minimal Router Firmware Version: 2.0.5

Path 4: AS 1-7 → AS 1-6 → AS 1-4 → AS 1-2 → AS 2-1 → AS 2-2 → AS 4-1 → AS 4-2
Latency: 170 ms, Bandwidth Up/Down: 600/1200 Mbps
Minimal Router Firmware Version: 2.0.5

Path 5: AS 1-7 → AS 1-5 → AS 1-1 → AS 2-2 → AS 4-1 → AS 4-2
Latency: 150 ms, Bandwidth Up/Down: 300/800 Mbps
Minimal Router Firmware Version: 2.1.1

Our path requirements are as follows:

- Maximal Latency: 150 ms
- Minimum Bandwidth Up/Down: 500/1000 Mbps
- ISD Whitelist: ISD-1, ISD-2, ISD-4
- Minimal Router Firmware Version: 2.1.0

In the case not all requirements can be met, latency holds the highest weight, followed by the ISD whitelist, then the firmware version, and then the bandwidth.

The basic flow is where the actor surgeon utilizes the client program to establish a connection with the server program that controls the surgery robot. The scenario is successful when the surgeon has established a connection, sends three commands, and then shuts down the connection. The three commands are "up 5", "down 2", and "left 2". In a real-world scenario, a joystick would be used to send more precise coordinates for movement, however for the simplification we use more simple commands.

1. The client program starts.
2. A policy file is read by the program that indicates the requirements for the path selection.

3. The program attempts to make a connection with the server on AS 4-2, port 1234.
4. A connection is established using a path that meets all requirements, specifically Path 2.
5. The surgeon sends the command "up 5".
6. The surgeon sends the command "down 2".
7. The surgeon sends the command "left 2".
8. The surgeon sends a finish signal and closes the connection.

Now that we established a basic flow, we can describe alternative flows and what actions should be taken then.

3.3.2 Alternative Flow 1

Our first alternative flow describes a situation where a path cannot be found that meets all requirements.

3.1 The surgeon attempts to make a connection with the server on AS 4-2, port 1234.

3.2 Of the five earlier described paths, all but Path 2 are available. These four paths do not meet all requirements.

3.3 Path 5 is chosen as it meets all requirements except bandwidth and the scenario continues with step 4.

3.3.3 Alternative Flow 2

Our second alternative flow describes a situation where an AS along the chosen path is no longer available while the surgery is in progress.

6.1 The surgeon sends the command "down 2".

6.2 AS 2-2, which is on the current path, is no longer available.

6.3 A new connection is established using path 1 as that is the only available path.

6.4 The client program sends the command down 2 again and the scenario continues with step 7.

3.3.4 Relationship to Reality

It is important to note that we simplified the processes within our scenarios.

In a real-life scenario, a joystick would be used that sends many larger messages, for example, detailed coordinates, in a very short period of time. In our flows, we send three simple commands as our main purpose is to test whether the different aspects of our design are possible. We have also simplified our scenarios by excluding a video stream as it does not add anything new to our test scenarios. The most important aspect of our requirement for a video stream is the bandwidth, which is looked at when a path is selected.

The minimal router firmware version would be a vague requirement in a real-world scenario, as there would be many different brands of routers each with their own version scheme. For the use case, we assume every single router of every AS is of the same type. Our aim is to explore how this information can be included and how we can use this to choose a path, which we can explore more easily with this simplification.

The topology of our use case is also different from the topology from SCIONLab. The SCIONLab topology is quite big, so for the use case topology, we took inspiration from it. Specifically, ISD-1 of the use case is an adjusted version of ISD-19. ISD-2 is a cut-down version of ISD-17. ISD-3 is a reduced version of ISD-16. ISD-4 has the same nodes as ISD-18, however we reduced some of the connections with other ISDs. This was done to improve the readability of the topology while still keeping interesting scenarios.

Chapter 4

Application Design

In this chapter, we will discuss the application design. First, we will give a high-level overview of the application itself and we follow this by explaining the different features of the application and the choices we made for the design.

4.1 Application Overview

Our application is present on two devices: that of the controller and the receiver. The controller is used by the surgeon performing the remote surgery. The receiver is a computer that is directly connected to a camera, to present a live feed to the surgeon, and the robot, to perform the commands that the controller sends. The application directly runs on the SCION network with its own AS. This situation we also described and showed in Section 3.2 and Fig. 3.1.

The controller application has several features. First, there is a status indicator that indicates which AS the application is communicating with and it indicates whether all requirements are fully met, partially met, or not met at all. Further, there is a policy selector that allows the controller to set the requirements, such as maximal latency and the permitted ISDs. This allows for different policies during different steps of the surgery. Next, there is a separate screen for video that shows what is on the receiver's camera. Lastly, there are several buttons that allow the application to send commands such as "up", "down", "left", and "right". In a real-life scenario, a different system, like an advanced joystick, might be used. However, for the theoretical exploration of SCION's capabilities, we chose a simplified version.

The receiver application is meant to run without human interaction. Once it is started, it is connected to the robot and the camera. Once it is connected to another application, it will actively stream the camera feed to it. It should also automatically authenticate a connection attempt to ver-

ify the connection is made by an application that has permission to connect.

The controller application has to go through several steps to be able to send a packet:

1. First it needs to find all possible paths to the destination AS and extract different properties from each path, specifically latency, bandwidth, geolocation and minimal router firmware version. We describe the extraction process in detail in Section 4.2.

2. Once the properties of all available paths are known, a path has to be selected (Section 4.3). It does this by first reading from the preference policy file (Section 4.3.2) and then by creating a ranking using the Reference Ideal Method (RIM) (Section 4.3.1).

3. It then creates a connection with this path, which allows the user to send commands and see the path compliance. Path compliance is a status that indicates which requirements are met and which are not.

4. If the user enabled multi-path mode (Section 4.4), data will also be sent through a second connection if the primary path has a significant performance loss.

4.2 Extracting Path Properties

To evaluate if a path matches the requirements set by the user, the application first has to extract properties for the path. For the use case, we look at four different requirements: latency, bandwidth, geolocation, and router firmware version.

4.2.1 Latency

Latency, or rather two-way latency, is the time delay between sending a packet from a source to a destination, and receiving a response. This can be measured by, for example, sending a ping on a specific path several times and taking the average. The latency is also available as static information in the path metadata, however, within the time that a PCB is active and thus the path metadata valid, the latency can greatly change. Battipaglia et al. [10] uses the first mentioned method and outputs the average latency of 30 pings in milliseconds (ms). It then stores this data as an entry in a MongoDB for each path. MongoDB is a NoSQL database product that utilizes JSON-like documents. We will be using the UPIN method [19] to measure the average latency of the paths available to the application.

It is important to note that latency can change at any given moment. For example, if the volume of traffic increases at an AS along the path, the

latency may increase. This makes it important to continuously measure the latency. Since the UPIN application only performs the measurements once and this work requires continuous results, we can modify the existing code to loop and continuously update the MongoDB.

4.2.2 Bandwidth

Bandwidth is the capacity or throughput of a channel and this can be measured both upstream and downstream in bits per second (bps). Like latency, bandwidth is information available in the path metadata but it is a value that can change based on how much traffic there is. To test bandwidth within SCION, you can also use a SCIONLab application called bwtester. The UPIN project [10] also measures average bandwidth and uses bwtester to do so. In the paper, the average bandwidth is measured both with 64 byte packets and with MTU-sized packets. In our use case, we will also deal with smaller packets, like for commands to the surgery robot, and larger packets for video. This means we would want to measure the bandwidth both for 64-byte packets and for MTU-sized packets.

We use the UPIN method [19] to measure the average upstream and downstream bandwidth for each path. For each path, we measure the bandwidth with smaller packets and larger packets, and then we select the bandwidth for the larger packets. If this is 0, due to measuring faults, we take the bandwidth of the smaller packets. Alternatively, we could use two paths, one for a low bandwidth and one for a high bandwidth, but this also doubles the number of paths needed for multi-path and means the receiver has to set up additional servers for receiving. Because that would require additional resources and the SCIONLab network is not always stable enough for high bandwidth paths, we use one path for both small and large packets. Similar to latency, we need to measure bandwidth continuously. This can be done in the same loop as when we measure bandwidth.

4.2.3 Geolocation

The geolocation refers to the physical location of the AS. In SCION this information can be extracted in two ways: Path metadata from beacons or the ISD number.

The path metadata from beacons is more specifically the geographic position as GPS coordinates of each router along the path. This is highly specific information, which allows for better tuning of a geolocation requirement. However, it also requires a more complex specification of the requirement, as each specific GPS coordinate has to be translated to a country or region. Also, an AS can fill in information that they like in the path metadata. This means that the information provided can be inaccurate. While it is possible to ensure an AS is not travelled on if they provide false information, it may

not be detected immediately that the information is false.

A second method to extract the geolocation is based on the ISD number of an AS. In SCION, all ISDs are a logical grouping of ASes that share a uniform trust environment, for example, a common jurisdiction or country, or they are all ASes related to healthcare providers. An AS is also not just able to join an ISD and instead needs permission to join one. ISD information cannot be manipulated, and this means an end host is sure where an AS is located. While there is still a translation from ISD to country or region, it is a much smaller translation table compared to GPS coordinates.

For our design, we use the ISD method to extract properties. From the perspective of our use case, the ASes we trust or do not trust are generally based on country. The ISD based grouping should provide enough accuracy to allow for selecting a secure path, while not needing to verify further if the extracted geolocation is correct.

4.2.4 Router Firmware Version

As of now, SCION does not offer a direct method to obtain information like a router firmware version. Without adjusting SCION itself, it is possible to utilize the path metadata. The path metadata set by ASes has a notes section, which can be used to convey information outside of the already set values. This can be used to, for example, insert information about the router. Of course, this does mean that each AS along the paths we may want to use has to set this information. An AS may be unwilling to provide such information as it can be sensitive information. Not all ASes may be willing to share they run on an unsafe firmware version. It is also possible that ASes misconfigure this information accidentally. SCION also provides no native method to verify this information, so another AS has to trust the information provided is correct.

For our design, we attempt to detect if there is information in notes following the format `Router Firmware Version: x.x.x` where each x corresponds to an integer that is 0 or higher. This must be checked for each AS in the path. If no firmware version is available for any AS, the value of the path is nil. If for every AS in the path there is a firmware version available, the lowest firmware version is used to describe the path. We will also adjust the MongoDB to store this information.

4.3 Automatic Path Selection

One of the main features of our application is an automatic path selection based on requirements. Ideally, we will always find at least one path that meets all requirements, however, there is no guarantee such a path is available. If a remote surgery is actively happening, it is also important that a

Method	Description
WPM	Uses weights to create a ranking based on several criteria.
ELECTRE	Select a set of alternatives that represent the best trade-off based on different criteria.
PROMETHEE	An outranking method that goes over several iterations to achieve a ranking between alternatives.
TOPSIS	Select alternative based on shortest distance to the positive ideal solution and farthest distance from negative-ideal solution.
RIM	Extends TOPSIS to be able to specify and handle value constraints.

Table 4.1: Table describing the different considered Multi-Criteria Decision Methods.

connection is available at all times, even if it means traversing a path that does not meet all requirements. This means we need a selection algorithm to choose the best possible path out of the available paths. Once we select a method, we can describe a selection policy. The policy can be used to specify the requirements and to give any additional input the algorithm might need.

4.3.1 Selection Algorithm

To select a path that best meets the requirements, we need a method that is able to take several requirements as input, both quantifiable and non-quantifiable, and give us a ranking of paths based on how well they meet the requirements. The advantage of a ranking instead of a singular path is that we can more quickly switch paths if the current path fails by performing a lookup in the ranking. Multiple-criteria decision making (MCDM) methods are used when there is a decision to be made where a selection of the best alternative can be complex [20]. There are also many methods within MCDM that can provide a ranking based on quantifiable and non-quantifiable requirements.

There are many MCDM methods, each with its own advantages and disadvantages. We have looked at several methods to determine which method is most suitable for our path selection. In Table 4.1, we made an overview of the different methods we considered.

The Weighted Product Model (WPM) [21] is a fairly simple but popular method. It is adapted from the Weighted Sum Model (WSM), but WPM uses multiplication rather than addition. A comparison is made between alternatives by multiplying the alternative with one ratio per criterion. Each

ratio is also raised to the power of the weight of the criterion. The advantage of this method is that it allows for different units of measurement. It is also fairly simple to use and quick to calculate. However, there is no clear normalization step and the model doesn't allow for a good comparison to a set of requirements. Also, weights are quite subjective and can be difficult to set correctly.

ELECTRE, or *ÉLimination Et Choix Traduisant la REalité* [21], is an MCDM method that uses outranking. Outranking methods compare pairs of alternatives and then assign them a score based on how well they satisfy the criteria. There are many variations of ELECTRE, such as ELECTRE I, ELECTRE II, and so on. We specifically consider ELECTRE I which is intended for selection problems. Its goal is to select a set of alternatives that represent the best trade-off based on different evaluation criteria. The model allows for a comparison between a set of requirements, however, it is not a complete method and only reduces the original set of alternatives by eliminating those that are not favourable. This is convenient when you have few criteria and a large set of alternatives. Also similarly to WPM, it uses weights that are very subjective. Lastly, ELECTRE can be susceptible to the rank reversal problem. This means that if an alternative is removed or added from the set, the ranking changes in a significant way. For example, if you remove an alternative, and the alternative previously ranked number one is now ranked number three.

PROMETHEE [22] (Preference Ranking Organization Method for Enrichment Evaluation) is another outranking-based MCDM method. Similarly to ELECTRE, PROMETHEE has many variations. We consider PROMETHEE II, which is able to provide a complete ranking. It is a method similar to ELECTRE in that it aims to provide a ranking based on multiple criteria. There are several preference functions available that can be used based on the characteristics of the criteria. This allows more complex problems to be solved more easily, however, it can be hard to determine which preference function should be used. It is also susceptible to the rank reversal problem and like the other methods, setting the weights can be difficult.

TOPSIS [21] (Technique for Order Preference by Similarity to Ideal Solution) is a method where the alternative is selected based on the shortest distance to the positive ideal solution and farthest distance from the negative-ideal solution. It uses an Euclidean distance approach to determine the distances. The method itself is straightforward in use. It is also possible to avoid the rank reversal problem by adjusting the normalization and adding two fictional alternatives that represent the maximal and minimal criteria. However, the method focuses on a maximal or minimal value, whereas the ideal solution may lie somewhere in between. For example, if the range of possible values is from 0 to 60, but the ideal alternative lies somewhere between 20 and 30, TOPSIS would be unable to find such an

alternative. Also similarly to all other methods, there is some weight sensitivity.

The last method we consider is RIM [23] (Reference Ideal Method) which is partially based on TOPSIS. The main difference between TOPSIS and RIM is that RIM uses an ideal range as opposed to a maximal and minimal value as the ideals. Also, an interesting quality of RIM is that when an alternative fully matches all criteria, it gets a score of 1, which is the highest achievable score. This also means that if there are several alternatives that match all criteria, they will rank equally on top. RIM also computes each alternative independently from each other, which means it does not suffer from the rank reversal problem. Just like the other methods, RIM utilizes weights, which can be sensitive to variation.

Each method has its own strengths and weaknesses. We decided on RIM to select the best path. One important aspect is that it does not suffer the rank reversal problem. As our selection of paths can change at any time, it is important we can quickly switch paths without needing to recalculate all paths again to gain a new ranking. While TOPSIS is also a strong contender and can be adjusted such that the rank reversal problem is not an issue, it only deals with extremes. The advantage of RIM, compared to TOPSIS, is that we can have varying criteria where we might want to have an ideal range as opposed to an extreme.

4.3.2 Policy Specification

A policy will be needed for the surgeon, or the IT team with the surgeon, to specify the requirements. RIM needs several inputs per criteria, or property.

The first input is the weight. This is a number between 0 and 1 that determines how important the criterion is. The higher the weight, the more influence the criterion will have on the ranking. It should be noted that all weights together should add up to a total of 1.

The second input is the range. The range is the range of values that a property can be. For quantifiable methods, this is for example 0 to 1000. However, infinity is unsupported in Golang's JSON encoder. This can be an issue for values like bandwidth or latency, where technically the max range can be infinite. To work around this, we instead choose a very high number that those values realistically do not hit. For example, a latency of 100000ms. With non-quantifiable properties, you assign each option a number from 1 to n, where n is the number of options. The better the option, the higher the number. For example, if the options are "bad", "ok", and "good", "bad" would be equal to 1, "ok" would be equal to 2, and "good" is equal to 3, and so the range is 1 to 3. We can also use the range for preprocessing. If there are any hard requirements, for example, a latency that may not be above 700 ms, any paths that are outside this range can be

filtered out before.

The third input is the ideal range. This is the range we want the associated path's property to be in. If we look at the requirements example from our use case, the latency requirement was 150 ms. In RIM, this would be represented as 0 to 150ms, as a lower latency is better. A bandwidth of 1000 Mbps would be a range of 1000 to infinity. To work around Golang's JSON encoder limitations, we set the maximum ideal that should be infinite to the same value as the maximum range.
In Appendix A.1, we show an example of how the policy file would look like for the requirements set for the use case from Chapter 3. Each entry describes a property. The `rangeMin` and `rangeMax` describe the actual range of the property. For latency and bandwidth, we set the maximum very high to work around Golang's limitations. Geolocation and router firmware version are quantifiable properties, but for these properties, we are only interested in whether the requirement is fully met or not. For example, if we have a whitelist of ISDs, we are interested if all ASes along the path are whitelisted. If one or more is not whitelisted, the path fails to meet the ideal. Because of this, we set the range from 1 to 2. The `idealMin` and `idealMax` describe our requirements. The `weight` values describe the weight given to each property and have been arbitrarily chosen. They will need tuning to ensure the correct path is chosen. Lastly, `additionalInfo` describes how to translate non-quantifiable properties. Each non-quantifiable property will need its own parser to properly translate the additional information and the properties given by the path to a number that can be used in the formula. The disadvantage of this is that it makes it harder to extend the preference policy file with new properties. However, it does allow developers to define properties the way they want to. For example, one hospital may want to work with a whitelist for geolocation and another with a blacklist.

4.3.3 Selecting a Path

To send packets over a path, a path can now be chosen. This is done by first gathering all properties of each available path from our database. Then, we enter the preprocessing stage where all paths which are below or above the specified range are excluded. Furthermore, all quantifiable properties have to be converted to a number. We then use RIM to obtain a ranking. In the case there are multiple paths that share the first place, which can happen if multiple paths meet all requirements, we choose the path with the best latency. A connection will then be established using this path and packets will be sent over this path. As latency and bandwidth change with each loop of the property collector, a new ranking will be made every single time. If a better path becomes available, a new connection will be made with that path. Once the new connection is established, the old connection will close.

4.4 Multi-pathing

To further ensure the availability of a path, we can create a multi-path mode. In multi-path mode, data is sent over two or more paths at once. Linc [7] describes two different modes: redundant and adaptive. In redundant multi-path mode, traffic is always sent over two or more paths, which makes it ideal for situations where guaranteed availability is a must. In adaptive multi-path mode, a second path is only used when the performance of the primary path is below a certain threshold. This allows for a quick change of primary path if needed.

Before we can choose a mode, we need to consider how a second path is selected. The most obvious solution would be to pick whatever path is next in the ranking. However, you then run into the chance the path is almost the same as the one first in the ranking, but with one AS different. This means that if an AS fails on path 1, there is a high chance the second path also fails. To solve this, we would need to implement some kind of distance measurement that finds a balance between the best available path and the difference in which ASes are visited. We would need to use this method if we use redundant multi-path mode.

With adaptive multi-path mode, we can utilize the performance drop to select a second path. If the performance decreases, it means the ranking would likely also change, if aspects like latency and bandwidth have some weight. With this knowledge, we can expect that the path currently ranked at 1 or 2 when the performance decreases will no longer contain the AS that caused the performance decrease. Because of this, we decided on the adaptive multi-path mode. This is also a simple adjustment on our earlier mentioned path selection without multi-pathing, where we switch from the path once a better path becomes available. We can utilize the same method, but instead, we add a performance threshold, for example, a latency increase of 20%, and we keep the second path open rather than immediately switch to it.

Chapter 5

Evaluation

In this chapter, we first explain the test setup and the capabilities of our prototype. Next, we measure the efficiency of our prototype. Lastly, we evaluate our results.

5.1 Test Setup

Our test setup of our prototype application [24] consists of two laptops both attached to the SCIONLab network; one laptop which represents the application on the surgeon's side and one laptop which represents the application on the robot's side. This setup is also shown in Fig. 5.1 with the SCIONLab topology in Fig. 5.2[1]. As the figure shows, the SCIONLab topology is different from our use case. We used the SCIONLab network as opposed to a local topology so we could test our prototype on a wider scale and with large physical distances between ASes. Variable network properties cannot be imitated well with a local topology. For our ASes, we used the scionproto branch, specifically version 0.9.0[2]. At the time we started the experiments, the SCIONLab network used a few years old branch of the scionproto open source code and we wanted to potentially utilize some of the newer features and adjustments.

The Laptop Client runs a local instance of the MongoDB on 127.0.0.1:27017 and it runs the AS with address 18-ffaa:1:10bc. This AS is connected to an attachment point (AP) in the US called CMU AP. It also runs a client application that features the surgeon's side of the prototype and a separate shell client that runs the path collection. For our tests, we run the path property extraction once before we start the client application. The bandwidth tester occasionally gets stuck and needs manual interference to restart it. Because of limitations of the SCIONLab network, we limited the bandwidth for the tests to 1 Mbps and we limited the MTU-sized packets to 1000 bytes.

[1] `https://www.scionlab.org/topology.png`
[2] `https://github.com/scionproto/scion/releases/tag/v0.9.0`

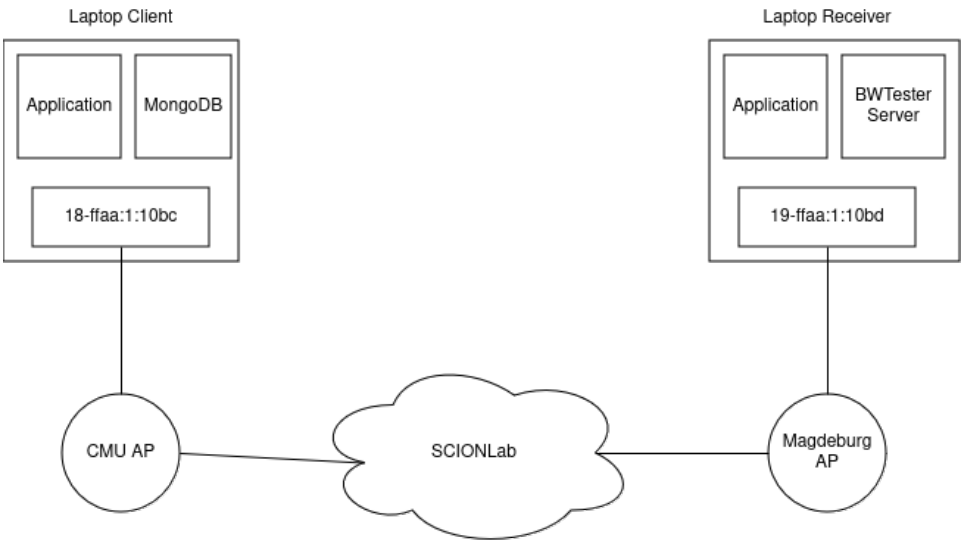


Figure 5.1: Topology of the test setup of the prototype.

The Laptop Receiver runs the AS with address 19-ffaa:1:10bd, which is connected to an AP in Germany called the Magdeburg AP. The receiver runs the application which features the robot's side of the prototype. It also runs a bwtester server on 127.0.0.1:30100 to allow the client to constantly test the bandwidth of the different paths.

Our prototype is a terminal-based application to show path property extraction, path selection based on requirements, and a multi-path feature. Its main purpose is to show these features are possible within SCION, so it is only a simplified version of our original design. The client terminal has the following commands:

- *move dir num*: This command sends the command *move* with a direction *dir* (Up, Right, Down, Left) and *num* steps to the receiver. Once the receiver receives this, it will send a confirmation back.
- *policy policylocation*: The command *policy* sets the location of the policy file to *policylocation*. If this is a valid file path, a new best possible path will be found and the connection will be changed to use the new path.
- *multipath true/false*: The command *multipath* when set to *true* will create a second connection. If this is enabled, future *move* commands will send the command using both connections. The receiver will reply to the first one it receives. If the command *multipath* is set to *false*, the second connection will be closed if it exists.
- *status*: The command *status* displays if the current path meets all

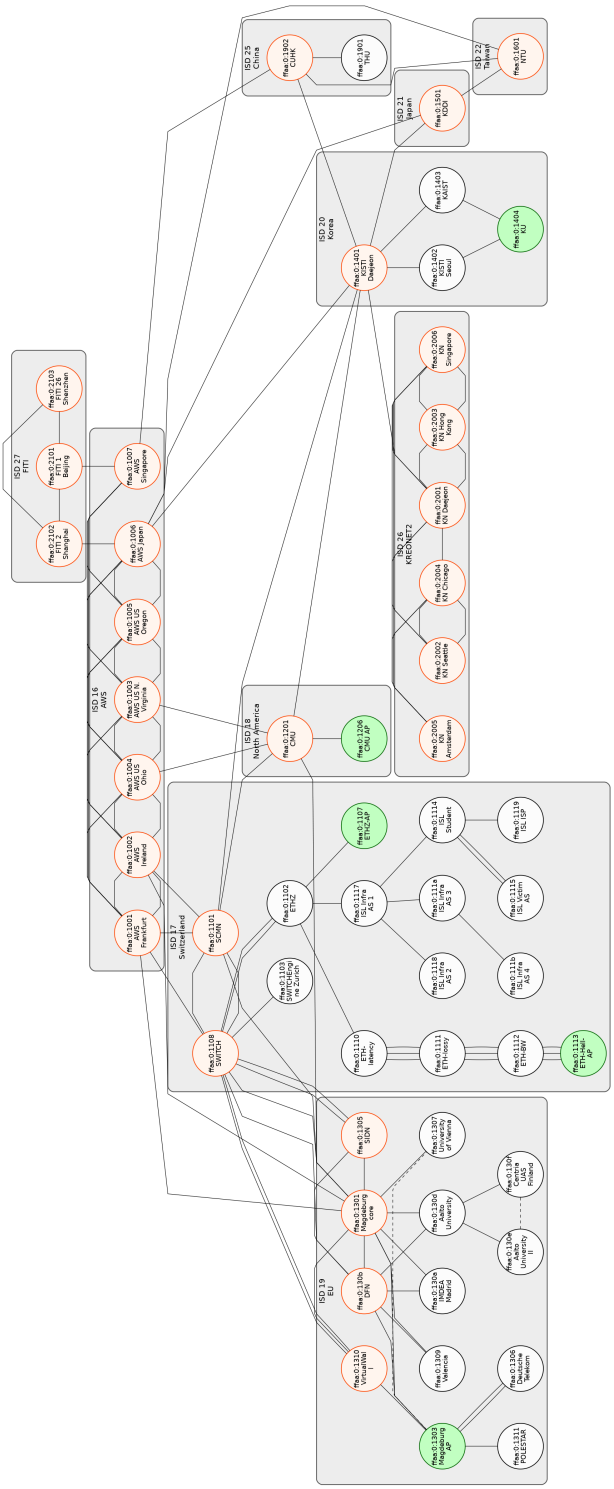


Figure 5.2: Topology of the SCIONLab network, which shows where CMU AP and Magdeburg AP are located. Red nodes are core ASes and green nodes are APs.

requirements. It will also display which requirements it does not meet if there are any. If two connections are present, it will display this information for both connections.

- *help*: The command *help* displays all commands and how to use them.
- *exit*: The command *exit* exists the application.

For further verification, after each input, the application shows the current coordinates. Up increases the y coordinate with the specified amount whereas Down decreases it. Similarly, Right increases the x coordinate and Left decreases it.

The receiver terminal reads the incoming move command message and also keeps track of the coordinates. Whenever it updates the coordinates, it will print the coordinates as confirmation. After, it will also send a verification message to the client that it received the message. The receiver will also always have two ports open at once, in case multi-path is enabled. This allows the second port to receive messages through the alternate path.

5.2 Test Results

First, we performed three tests to verify the functionality of our system. While we do not have the same topology as our use case scenarios (Section 3.3), we can imitate the circumstances. We first performed the basic flow by setting the appropriate policy file, which resulted in a new path *8-ffaa:1:10bc#0,1 18-ffaa:0:1206#110,1 18-ffaa:0:1201#8,5 19-ffaa:0:1301#3,5 19-ffaa:0:1303#1,463 19-ffaa:1:10bd#1*. Here, a path is represented by each AS and its interfaces, so it follows the construction of AS#interfaceInbound, InterfaceOutbound. Afterwards, we sent the three commands and verified by reading the coordinates on both ends that all three commands from the scenario were successfully sent. At the end, the coordinates are $x = -2, y = 3$.

Next, we adjusted the policy file to have stricter latency requirements such that no path meets all requirements. We did this by decreasing the idealMax for latency from 300 to 30 and by only whitelisting ISD 26 to ensure no path meets all requirements. The system found and set the path to *18-ffaa:1:10bc#0,1 18-ffaa:0:1206#110,1 18-ffaa:0:1201#8,5 19-ffaa:0:1301#3,5 19-ffaa:0:1303#1,463 19-ffaa:1:10bd#1* and by using the status checker we verified it met none of the requirements. Interestingly, this was the same path as with our original policy. Likely this is because the latency is given a high weight, and this path had the lowest latency of all 40 paths.

Lastly, we tested what happens when an AS is no longer available. As we cannot manipulate the availability of the ASes within SCIONLab, we removed all database entries that had a path containing the AS 19-ffaa:0:1301. This left us with 6 possible paths of the 40. In a real-life

scenario if the AS would go down, the AS being unavailable would get picked up by the property collector as paths containing the AS cannot be pinged and these entries would be removed from the database. Now after removing the AS, instead of the default path from our basic flow test, the new path *18-ffaa:1:10bc#0,1 18-ffaa:0:1206#110,1 18-ffaa:0:1201#8,4 17-ffaa:0:1101#5,7 17-ffaa:0:1108#1,8 19-ffaa:0:130b#1,4 19-ffaa:0:1303#286,463 19-ffaa:1:10bd#1* was chosen.

We also tested the efficiency of the path selection. First, we measured the time it takes to perform the path collection, which consists of two parts. The first part is to collect 40 paths, which is a sufficient number of paths to find the best available path. This was also proven by UPIN [10]. The second half performs the ping test and bandwidth test on each path to collect the properties. The ping test sends 30 pings at 0.1 second intervals. The bandwidth test sends a 1000 byte packet over 3 seconds while targeting a bandwidth of 1 Mbps. This second part, as described in the previous section, can get stuck. Due to this, it took us 20 attempts at 1 Mbps to gain 10 results. In Table A.1a we show the time in seconds for each part of a completed property collection run and the total amount of time it takes. Taking the median, one completed run takes close to 9 minutes (547 seconds) with 40 paths. There was one outlier of around 7 minutes (417 seconds) and we suspect several paths timed out during the test and were then skipped.

We ran the same test with all paths of 6, 7, and 8 hops, which was a total of 18 paths. It took us 17 attempts to gain 10 results as 7 of the attempts got stuck during the bandwidth test. The successful results are shown in Table A.1b. In both cases, the average time for path testing was between 13 and 14 seconds per path.

We also set up an AS attached to the Swiss ETHZ-AP and measured property collection again to see if physical distance has a significant impact on the speed. Physical distance is reduced by using a Swiss AS in two ways. First, the physical distance between the Swiss AS and the German AS is less compared to the US AS and the German AS. Second, the laptop hosting the AS is in the Netherlands, which is much closer to Switzerland than it is to the US. The results of this can be seen in Table A.2a. To get those 10 results, it took us 38 attempts. The last three results of the runs are significantly lower than the other results. While all of these runs were completed, they did not provide 40 path results. We suspect that while we ran our tests, one of the ASes in the SCIONLab network was temporarily unavailable. Fig. 5.3 shows a comparison between the runs of the US AS with 18 paths, the US AS with 40 paths, and the Swiss AS with 40 paths. We note that physical distance has a noticeable effect. On average, using the Swiss AS improved the speed by 0.5 to 1 second per path. The physical distance from Magdeburg to Zurich is around 575 km and the physical distance from Magdeburg to Pittsburgh, where CMU is located, is around 6680km, so while

distance has a noticeable effect, it is proportionally less than expected.

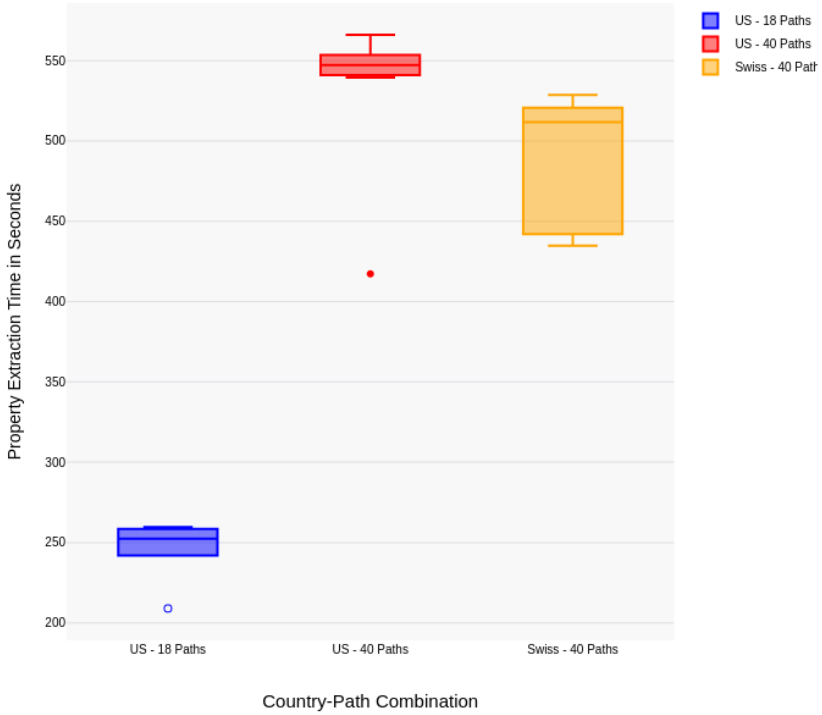


Figure 5.3: A comparison of path extraction runs in seconds with the US AS (18 and 40 paths) and the Swiss AS (40 paths)

After, we used a short script [24] to measure the time it takes to set up a path without path selection, a path with path selection, and a path with path selection and multi-path. All three measurements were performed ten times. Measuring with path selection includes setting a policy, calculating the scores, and then setting the correct path. The results are displayed in Fig. 5.4, Fig. 5.5 and Fig. 5.6 and the full measurements can be seen in Table A.2. Overall, it can be seen that having more paths can increase the time it takes to select the best available path, but the difference comes down to around 1-2 milliseconds.

5.3 Evaluating the Results

Our final solution is an application design with automatic path selection where the path is selected based on a set of requirements described by a

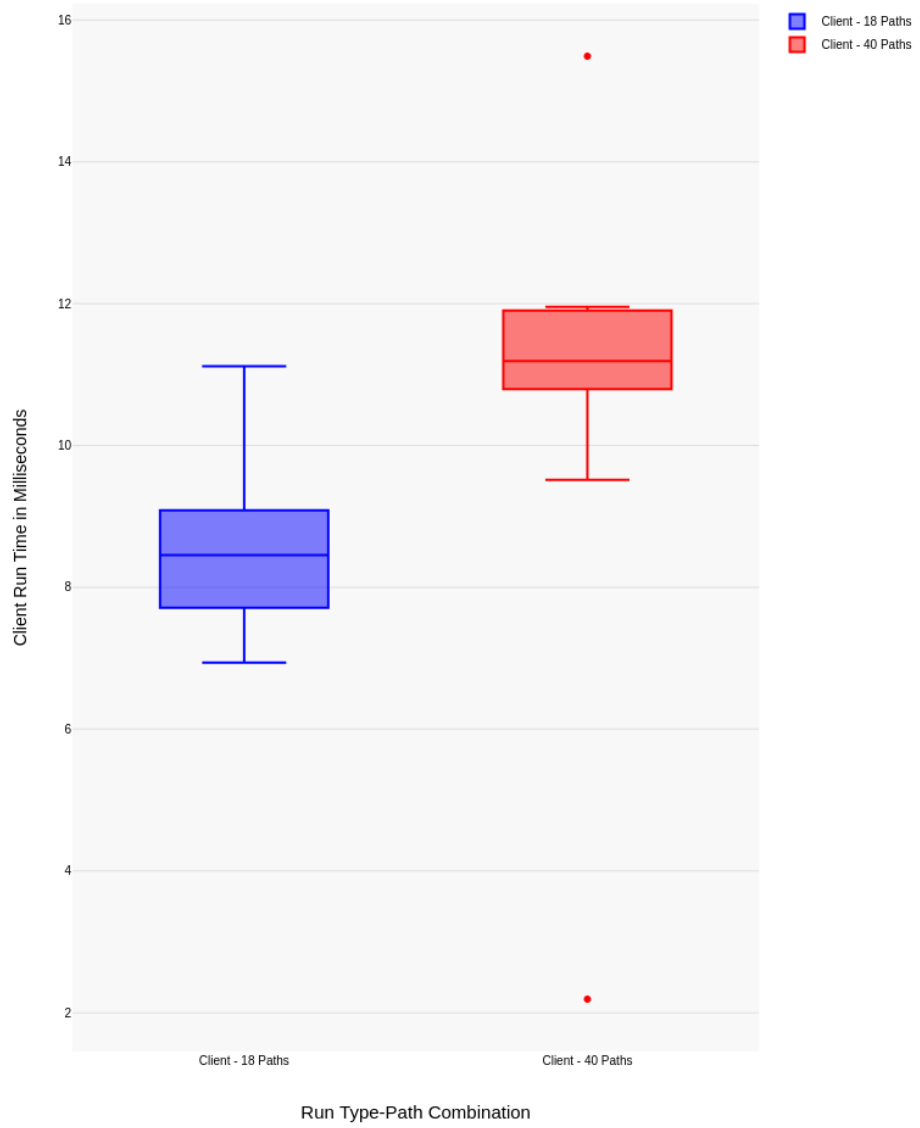


Figure 5.4: Run time comparison of client's path selection without a policy set with 18 paths and 40 paths

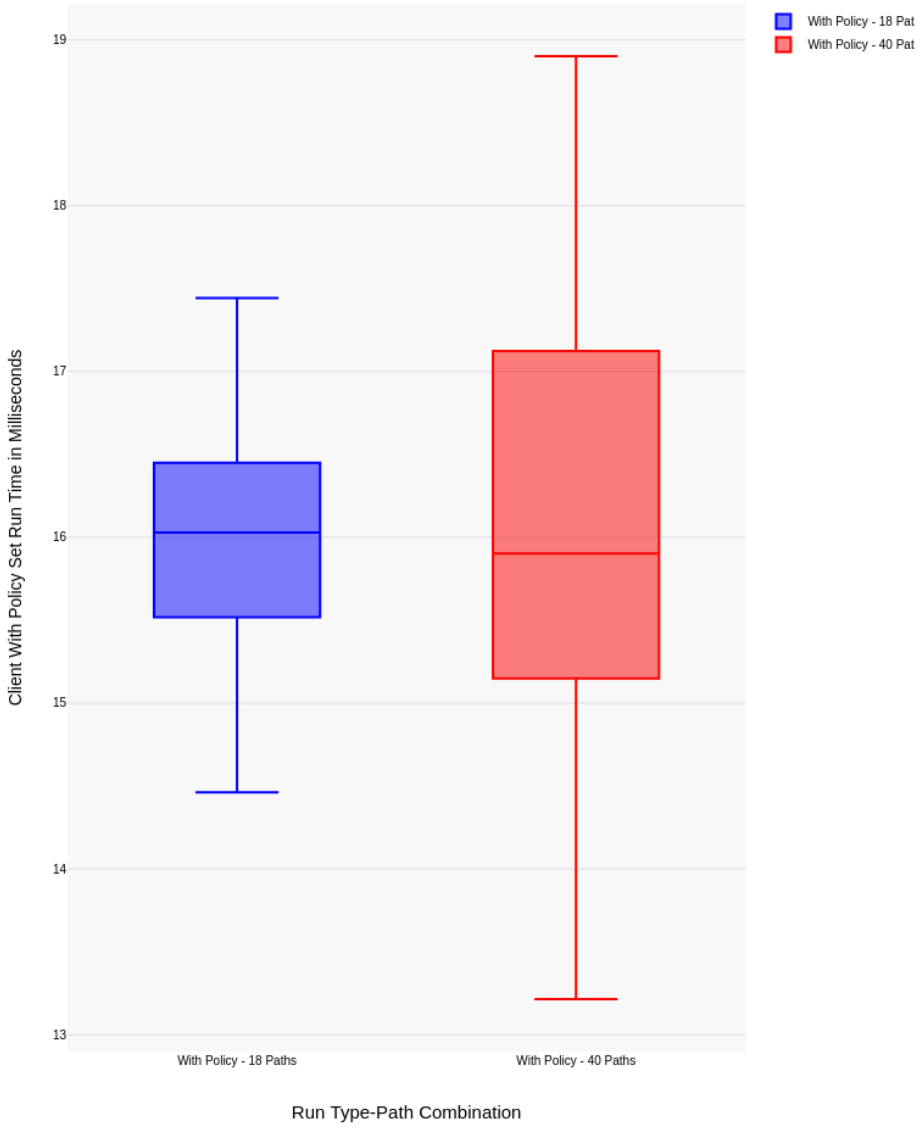


Figure 5.5: Run time comparison of client's path selection with a policy set with 18 paths and 40 paths

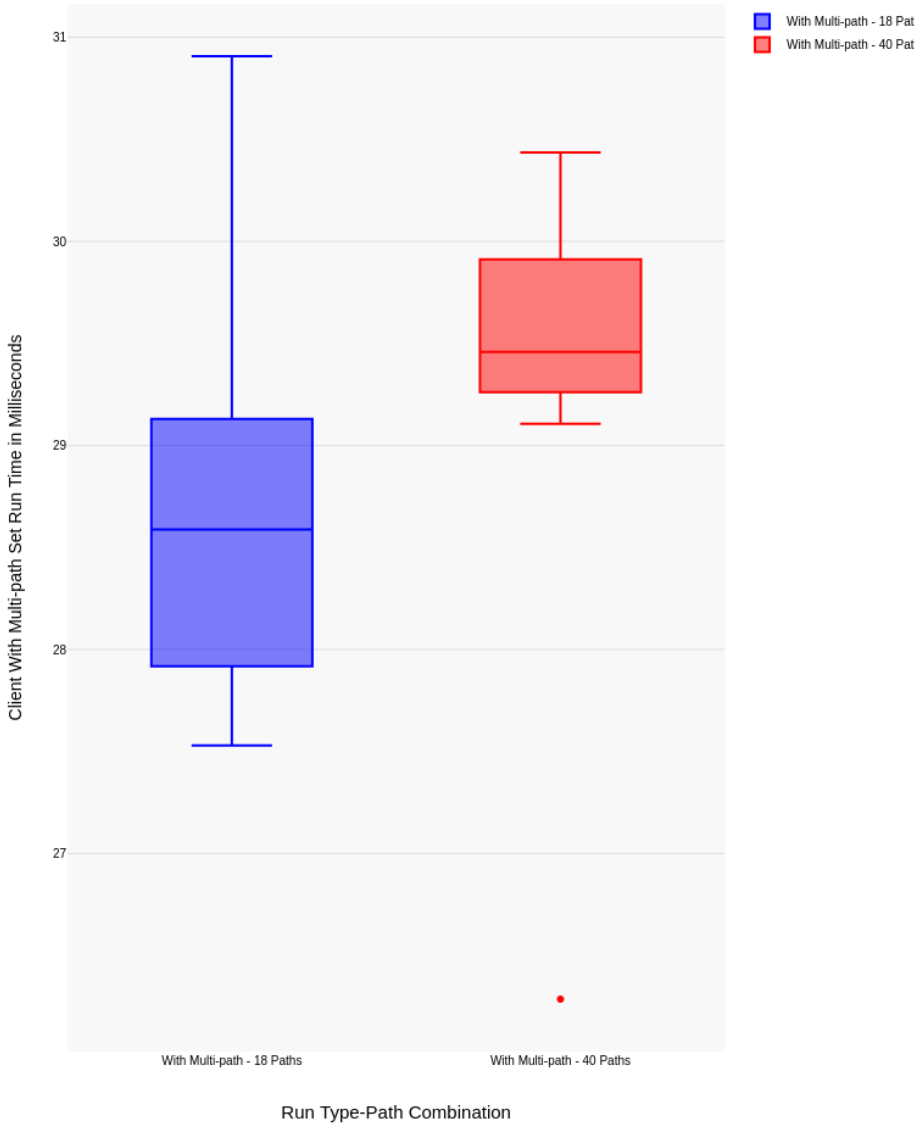


Figure 5.6: Run time comparison of client's path selection with a policy set and multi-path enabled with 18 paths and 40 paths

preference policy file. We had several requirements our application design (Section 3.2) had to comply with, which we show in Table 5.1.

	Requirement	**Fulfilled?**
1	Design choices should uphold the goals of SCION itself.	Partially
2	It should have property collection and then be able to automatically select a path based on latency, bandwidth, geolocation, and router firmware version.	Yes
3	There should be a method to specify a set of requirements a path should comply with where possible.	Yes
4	Availability should be guaranteed at all times, even if it means not fully complying with the requirements.	Yes

Table 5.1: Overview of the requirements of the application design and whether our design fulfilled the requirement.

Our first requirement was partially met. Our solution improved the availability and did not affect transparency, control and deployability, and formal verification. Efficiency and scalability were slightly negatively affected. Finding the best available path using our method is around 1.5 times slower compared to SCION's standard method of choosing a path. With multi-path enabled, this was close to 3 times slower compared to the standard method. Property collection is also not very scalable. Each path to be tested increased the time for property collection by around 13 to 14 seconds. That means the more paths there are, the longer it takes for a full cycle to complete and thus be able to do a health check on the present paths. Our solution does not fully meet the goal of extensibility and algorithm agility, as it is currently not straightforward to add new properties. Now, if the code has to be extended, custom code has to be written for each property, both for extraction and for processing. We also did not add formal verification to verify if a packet was sent along the selected path.

In Section 4.2 we describe how to extract paths and in Section 4.3 we describe automatic path selection, which means we fulfill requirement 2. We also describe a way to specify requirements in subsection 4.3.2 which allows us to fulfil requirement 3.

We fulfil requirement 4 in several ways. First, we always aim to select the best available path. This can mean using a path that does not meet all requirements to ensure availability. Next, the path status is constantly monitored. When there is a new best available path, this will be switched

to. We also added a multi-path mode (Section 4.4) that can be utilized to further ensure availability. When the current path's performance degrades, all data is also sent over a second path.

Chapter 6

Discussion

Now that we evaluated our design based on the use case, we will look at if, and how, our design and prototype are applicable to other potential use cases. In the Related Work (Section 2.4), we looked at the features our design should have: active property collection, path selection based on quantifiable properties, path selection based on geolocation, path selection based on non-quantifiable properties except geolocation, and a multi-path mode.

We achieved active property collection utilizing UPIN's code with minor adjustments. Specifically, we adjusted the locations of SCION, which were hard-coded, we increased the number of paths to be found, and we decreased the size of the large packets to 1000 bytes instead of MTU size. With this, we were able to gather quantifiable properties, and the geolocation in the form of ISDs. While an ISD is not a perfect geolocation, SCION aims to group ISDs in such a way that they share a common purpose or jurisdiction. This may not be enough for all use cases, but SCION does provide a way to supply coordinate data to an AS to specify the exact location.

In Section 4.2 we described a method to use SCION's path metadata to communicate non-quantifiable properties. This does, however, come with some limitations. The most major one is the inability to verify non-quantifiable properties. While ASes can utilize the notes section to add information, like a router firmware version, there is no way to verify whether the information is correct. With properties like latency, bandwidth, and geolocation, we have methods to verify the information. From a security perspective, this is a significant risk. Even when there are no malicious ASes present on the path, the information can be misconfigured. If our aim is to avoid certain routers or certain firmware versions of routers due to a vulnerability, accidentally going over paths with these properties is an issue.

Also, ASes may not be willing to share such information with all other ASes. Publically sharing information can reveal vulnerabilities, which they

may only with to share with certain ASes. This means a system would be needed where ASes can selectively share information with other ASes and encrypt this so only the end host is able to extract the shared path properties.

Our next three features, which can be combined for path selection with various requirements, are applicable to many other use cases. The RIM method is flexible as it allows you to set ideal values and hard requirements. RIM does require some extensive preprocessing, especially for non-quantifiable properties, but the method itself is fast as it scales to the number of paths and properties. The RIM method does require some tuning due to its usage of weights. Improperly tuned weights can lead to undesirable rankings. It requires some trial and error with balancing the weights in the preference policy to achieve the best available path.

Our chosen multi-path mode was very specifically chosen for our use case as availability was a significant security requirement. For some use cases, redundant multi-path mode may be more suitable. In that case, a method would have to be found to include some distance measurement to compare the amount of overlap any two paths may have. Ideally, you would want a minimal amount of overlap to reduce the chance of both paths failing due to an AS failing that was present on both paths.

We also created a prototype, which was an implementation of a selection of features. While it allowed us to verify property extraction, path selection, and multi-path mode are possible within SCION, it also showed us some of the limitations. Property extraction still has some major limitations. Beyond its limitations for non-quantifiable properties as mentioned earlier, it is also a slow process. The more paths are available, the longer it takes to run a full property extraction cycle. When trying to monitor path health, a cycle of several minutes would be insufficient. Properties like bandwidth and ping can vary greatly in this period of time, and may even decrease significantly enough that other paths become the better path. A solution to this could be to test the different properties in parallel cycles. For example, the ping tester and bandwidth tester run parallel from each other. As the bandwidth tester is what caused the long wait times, running latency in a different cycle would allow the application to monitor path health using latency.

Chapter 7

Conclusions

In this thesis, we presented a method to achieve an application-level requirement-based path selection on the SCION network. We first established a remote surgery use case in Chapter 3 to explore the requirements that were needed and to explore the considerations regarding path-aware applications. Next, we created an application design to answer our first two sub-questions.

1. **How can we extract different properties for possible paths?**
   To extract different properties from a path (Section 4.2), we used the same mechanism as the authors of the UPIN paper [10] to collect the latency, bandwidth, and ISDs from a path. We also described how we can use SCION's path metadata to convey information such as a router version. However, SCION does not offer a method native to SCION to verify path metadata.

2. **How can we find the best available path?**
   In Section 4.3 we compared several multiple-criteria decision making methods to create a path ranking. We chose the RIM method, as it enabled us to set hard and soft requirements. Soft requirements are parameters that a path ideally should have, and hard requirements are parameters that a path must be within to be valid. We also described a multi-path mode to ensure the availability of our application.

Next, we created a prototype to showcase that each of our described features is possible within SCION and to answer our third sub-question:

3. **What is the impact on the efficiency using our method in comparison to path selection without requirements?**
   Overall, path selection itself is slower when you set a policy compared to when you do not. When a policy is set, the path selection takes an average of 16 seconds. When a policy is not set, the path selection takes an average of 10 seconds. The path selection with a policy set and multi-path enabled takes an average of 29 seconds. This means

setting multi-path is almost twice as slow as when you only set the policy, and almost thrice as slow compared to setting no policy. For the majority of use cases, these are acceptable margins, and the code can be optimized to speed this up. However, the path property extraction process currently takes up to 9 minutes with 40 paths. For some properties, like geolocation, it is acceptable if updating that information takes a longer time, as these properties rarely change. However, aspects like latency should be kept up to date more frequently to be able to monitor the path health. It is possible to split up these cycles so that latency and bandwidth are measured separately. With that adjustment, latency would be updated more often.

By answering these questions, we are able to answer our research question:
"**How can we achieve application-level requirement-based path selection on the SCION network?**"
An application-level requirement-based path selection consists of two parts: a path property extraction and the actual path selection.

When considering property extraction, some properties can be measured, such as latency and bandwidth, and properties that are static information such as ISDs and router information. Quantifiable properties can be measured using various tools like ping or a bandwidth tester. Aspects like latency, bandwidth, and even jitter, to monitor the path health and ensure a better availability are straightforward to measure. Not each measurable property is appropriate to monitor path health though. The bandwidth and latency together, for example, take around 13 to 14 seconds per path to measure, which is too long. However, it is possible to split these measurements into their own cycles to gain information more quickly. Static information can be harder to extract as many properties are often not made available. The ISD can be gathered from the AS name. It is also possible to extend the path metadata to include different pieces of information about the AS, such as the router brand or version. However, it can be difficult to verify this information.

When considering path selection, this can be done by using a path ranking method, such as RIM. In the case the aim is not a path that is always available, but rather a path that meets all requirements, RIM can still be applicable as it outputs a 1 for a path if it fulfils all requirements, and then only paths that output a 1 are accepted. This method can also be used to find the best path of all paths that meet the requirements. This can be done by making the requirements the hard requirements, such that only scored paths are those that meet all requirements, and by then using the ideal ranges to set the preferences and gain a ranking.

As it stands, PAN within SCION is an interesting feature, but we recom-

mend some capability be added to extract and verify more path properties. Currently, non-quantifiable properties are difficult to verify or do not have the support to verify them. An adjustment could be to run FABRID [8] natively on SCION, or a new method may be found with more research. For future work, we also recommend that the client is able to verify whether or not the packet it sent travelled along the selected path. While SCION has some validation natively, it does not offer end-host path validation. One method that could be utilized is EPIC [25] (Every Packet Is Checked) which allows end hosts to perform path validation. With these two additions, PAN can greatly improve the security of network communication.

Bibliography

[1] RIPE NCC, "YouTube Hijacking: A RIPE NCC RIS case study," https://www.ripe.net/publications/news/industry-developments/youtube-hijacking-a-ripe-ncc-ris-case-study, 2008, accessed on June 12, 2023.

[2] B. Stone, "Pakistan Cuts Access to YouTube Worldwide," https://www.nytimes.com/2008/02/26/technology/26tube.html, 2008, accessed on December 5, 2023.

[3] M. Lepinski and K. Sriram, "Bgpsec protocol specification - RFC 8205," Tech. Rep., 2017.

[4] T. Hlavacek, P. Jeitner, D. Mirdita, H. Shulman, and M. Waidner, "Behind the scenes of rpki," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 1413–1426.

[5] C. de Kater, N. Rustignoli, and A. Perrig, "SCION Overview," Mar. 2023. [Online]. Available: https://www.ietf.org/archive/id/draft-dekater-panrg-scion-overview-03.html

[6] D. Hausheer, "SCION - A Novel Internet Architecture," https://labs.ripe.net/author/hausheer/scion-a-novel-internet-architecture/, 2021, accessed on January 4, 2023.

[7] T. John, P. De Vaere, C. Schutijser, A. Perrig, and D. Hausheer, "Linc: low-cost inter-domain connectivity for industrial systems," in *Proceedings of the SIGCOMM '21 Poster and Demo Sessions*. Virtual Event: ACM, Aug. 2021, pp. 68–70. [Online]. Available: https://dl.acm.org/doi/10.1145/3472716.3472850

[8] C. Krähenbühl, M. Wyss, D. Basin, V. Lenders, A. Perrig, and M. Strohmeier, "FABRID: Flexible Attestation-Based Routing for Inter-Domain Networks," Apr. 2023, arXiv:2304.03108 [cs]. [Online]. Available: http://arxiv.org/abs/2304.03108

[9] T. Krüger and D. Hausheer, "Towards an API for the Path-Aware Internet," in *Proceedings of the ACM SIGCOMM 2021 Workshop on Network-Application Integration*. Virtual Event USA: ACM, Aug. 2021, pp. 68–72. [Online]. Available: https://dl.acm.org/doi/10.1145/3472727.3472808

[10] A. Battipaglia, L. Boldrini, R. Koning, and P. Grosso, "Evaluation of SCION for User-driven Path Control: a Usability Study," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*. Denver CO USA: ACM, Nov. 2023, pp. 785–794. [Online]. Available: https://dl.acm.org/doi/10.1145/3624062.3624592

[11] L. Chuat, M. Legner, D. Basin, D. Hausheer, S. Hitz, P. Müller, and A. Perrig, *The Complete Guide to SCION: From Design Principles to Formal Verification*, ser. Information Security and Cryptography. Cham: Springer International Publishing, 2022. [Online]. Available: https://link.springer.com/10.1007/978-3-031-05288-0

[12] Anapaya Systems, ETH Zurich, SCION Association, "SCION Header Specification," 2023. [Online]. Available: https://scion.docs.anapaya.net/en/latest/protocols/scion-header.html

[13] A. Davidson, M. Frei, M. Gartner, H. Haddadi, A. Perrig, J. S. Nieto, P. Winter, and F. Wirz, "Tango or square dance? how tightly should we integrate network functionality in browsers?" in *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, ser. HotNets '22. New York, NY, USA: Association for Computing Machinery, Nov. 2022, pp. 205–212. [Online]. Available: https://dl.acm.org/doi/10.1145/3563766.3564111

[14] A. Bernadotte, "Cyber security for surgical remote intelligent robotic systems," in *2023 9th International Conference on Automation, Robotics and Applications (ICARA)*, pp. 65–69, ISSN: 2767-7745.

[15] S. Iqbal, S. Farooq, K. Shahzad, A. W. Malik, M. M. Hamayun, and O. Hasan, "SecureSurgiNET: A framework for ensuring security in telesurgery," vol. 15, no. 9, p. 1550147719873811, publisher: SAGE Publications. [Online]. Available: https://doi.org/10.1177/1550147719873811

[16] D. I. Dogaru and I. Dumitrache, "Cyber security in healthcare networks," Jun. 2017, pp. 414–417.

[17] A. Ahad, M. Tahir, and K.-L. A. Yau, "5G-Based Smart Healthcare Network: Architecture, Taxonomy, Challenges and Future Research

Directions," *IEEE Access*, vol. 7, pp. 100 747–100 762, 2019, conference Name: IEEE Access.

[18] G. Moustris, C. Tzafestas, and K. Konstantinidis, "A long distance telesurgical demonstration on robotic surgery phantoms over 5g," vol. 18, no. 9, pp. 1577–1587. [Online]. Available: https://link.springer.com/10.1007/s11548-023-02913-2

[19] A. Battipaglia, "SCION Test Suite," https://github.com/MrR0b0t14/SCION-Test-Suite, 2023, accessed on February 12, 2024.

[20] M. Aruldoss, T. M. Lakshmi, and V. P. Venkatesan, "A survey on multi criteria decision making methods and its applications," *American Journal of Information Systems*, vol. 1, no. 1, pp. 31–43, 2013. [Online]. Available: http://pubs.sciepub.com/ajis/1/1/5

[21] E. Triantaphyllou, "Multi-Criteria Decision Making Methods," in *Multi-criteria Decision Making Methods: A Comparative Study*, ser. Applied Optimization, E. Triantaphyllou, Ed. Boston, MA: Springer US, 2000, pp. 5–21. [Online]. Available: https://doi.org/10.1007/978-1-4757-3157-6_2

[22] H. Taherdoost and M. Madanchian, "Using PROMETHEE Method for Multi-Criteria Decision Making: Applications and Procedures," *Iris Journal of Economics & Business Management*, vol. 1, no. 1, 2023.

[23] E. Cables, M. T. Lamata, and J. L. Verdegay, "RIM-reference ideal method in multicriteria decision making," *Information Sciences*, vol. 337-338, pp. 1–10, Apr. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0020025515009007

[24] A. van Veen, "Prototype," https://gitlab.science.ru.nl/avveen/prototype, 2024, accessed on April 18, 2024.

[25] M. Legner, T. Klenze, M. Wyss, C. Sprenger, and A. Perrig, "{EPIC}: every packet is checked in the data plane of a {Path-Aware} internet," in *29th USENIX Security Symposium (USENIX Security 20)*, 2020, pp. 541–558.

Appendix A

Appendix

A.1 Preference Policy JSON

```
{
  "properties": [
    {
      "property": "latency",
      "rangeMin": 0,
      "rangeMax": 1000000,
      "idealMin": 0,
      "idealMax": 150,
      "weight": 0.2000
      "additionalInfo": null
    },
    {
      "property": "bandwidthUp",
      "rangeMin": 0,
      "rangeMax": 1000000,
      "idealMin": 250,
      "idealMax": 1000000,
      "weight": 0.1000,
      "additionalInfo": null
    },
    {
      "property": "bandwidthDown",
      "rangeMin": 0,
      "rangeMax": 1000000,
      "idealMin": 1000,
      "idealMax": 1000000,
      "weight": 0.1000,
      "additionalInfo": null
```

```
    },
    {
      "property": "geolocation",
      "rangeMin": 1,
      "rangeMax": 2,
      "idealMin": 2,
      "idealMax": 2,
      "weight": 0.3000
      "additionalInfo": "Whitelist: 1, 2, 4"
    }
    {
      "property": "router-firmware-version",
      "rangeMin": 1,
      "rangeMax": 2,
      "idealMin": 2,
      "idealMax": 2,
      "weight": 0.3000
      "additionalInfo": "minimal version: 2.1.0"
    }
  ]
}
```

A.2 Experiment Timing Data

Run	Path Collection (s)	Path Testing (s)	Total (s)
1	0.419	553.073	553.492
2	2.556	544.566	547.122
3	0.415	546.948	547.363
4	0.415	551.806	552.221
5	0.411	546.687	547.098
6	0.433	555.256	555.689
7	2.734	414.547	417.281
8	2.567	536.838	539.405
9	0.42	540.544	540.964
10	0.429	565.573	566.002

(a) Property collection from AS 18-ffaa:1:10bc to AS 19-ffaa:1:10bd with 40 available paths

Run	Path Collection (s)	Path Testing (s)	Total (s)
1	2.393	257.243	259.636
2	0.374	241.491	241.865
3	2.28	248.904	251.184
4	2.676	250.907	253.583
5	2.233	246.306	248.539
6	2.211	254.922	257.133
7	2.176	257.252	259.428
8	5.052	203.92	208.972
9	0.371	241.582	241.953
10	5.035	253.435	258.47

(b) Property collection from AS 18-ffaa:1:10bc to AS 19-ffaa:1:10bd with 18 available paths

Table A.1: Property extraction times of completed runs in seconds where Path Collection is the time to find the available paths and Path Testing is the time to run the ping and bandwidth tests on the found paths

Run	Path Collection (s)	Path Testing (s)	Total (s)
1	1.757	526.815	528.572
2	1.886	509.146	511.032
3	0.427	526.637	527.064
4	0.349	513.948	514.297
5	0.376	520.218	520.594
6	0.804	500.162	500.966
7	0.37	512.076	512.446
8	5.039	436.929	441.968
9	5.034	429.671	434.705
10	5.052	431.845	436.897

(a) Property collection from AS 17-ffaa:1:10bb to AS 19-ffaa:1:10bd with 40 available paths

Table A.2: Property extraction times of completed runs in seconds

Client (ms)	With Policy (ms)	With Multi-path (ms)
11.117	17.443	30.908
9.205	14.461	28.657
8.92	16.429	29.13
8.347	15.558	27.918
7.424	17.018	27.714
6.938	15.518	30.572
8.559	16.405	27.531
8.327	15.651	29.0518
7.713	16.449	28.306
9.084	15.298	28.521

(a) Run Time in ms for Client, Client with a Set Policy, and Client with Multi-path Enabled on the US AS with 18 Paths

Client (ms)	With Policy (ms)	With Multi-path (ms)
2.193	15.624	29.458
15.488	17.122	29.415
10.81	15.149	29.107
10.794	13.214	29.912
11.102	15.755	26.287
11.275	18.901	30.387
11.623	17.448	29.711
9.513	16.868	30.436
11.955	14.144	29.459
11.904	16.046	29.262

(b) Run Time in ms for Client, Client with a Set Policy, and Client with Multi-path Enabled on the US AS with 40 Paths.

Table A.3: Run times in milliseconds