Evaluation of SCION for User-driven Path Control: a Usability Study

Antonio Battipaglia
a.battipaglia@uva.nl
Politecnico di Torino
Turin, Italy

Leonardo Boldrini
l.boldrini@uva.nl
University of Amsterdam
Amsterdam, The Netherlands

Ralph Koning
ralph.koning@sidn.nl
SIDN Labs
Arnhem, The Netherlands

Paola Grosso
p.grosso@uva.nl
University of Amsterdam
Amsterdam, The Netherlands

ABSTRACT

The UPIN (User-driven Path verification and control in Inter-domain Networks) project aims to implement a way for users of a network to control how their data is traversing it. In this paper we investigate the possibilities and limitations of SCION for user-driven path control. Exploring several aspects of the performance of a SCION network allows us to define the most efficient path to assign to a user, following specific requests. We extensively analyze multiple paths, specifically focusing on latency, bandwidth and data loss, in SCIONLab, an experimental testbed and implementation of a SCION network. We gather data on these paths and store it in a database, that we then query to select the best path to give to a user to reach a destination, following their request on performance or devices to exclude for geographical or sovereignty reasons. Results indicate our software is a viable option to offer users many paths to choose from, following a series of requests, and therefore perform user-driven path control in a SCION network.

ACM Reference Format:
Antonio Battipaglia, Leonardo Boldrini, Ralph Koning, and Paola Grosso. 2023. Evaluation of SCION for User-driven Path Control: a Usability Study. In *Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis (SC-W 2023), November 12–17, 2023, Denver, CO, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3624062.3624592

1 INTRODUCTION

Citizens and governments depend on digital technologies that are severely entangled in the main structure of society [5]. These technologies are built on the traditional Internet architecture and therefore inherit some of its limitations, such as the lack user control of the network, and a consequent erosion of trust [6]. The Responsible Internet paradigm wants overcome these problems by improving the Internet transparency, accountability and controllability [6]. The UPIN (User-driven Path verification and control in Inter-domain Networks) project, based on the notion of Responsible Internet, develops a framework for users to control the behaviour of the network [2] while integrating with the current Internet architecture. However, providing users with a degree of control over network traffic requires the network architecture itself to be designed differently from traditional approaches in use today.

SCION [9] is an Internet architecture designed to provide route control, failure isolation, and explicit trust information for end-to-end communications to end users. SCION addresses some of the problems that the Responsible Internet wants to overcome and provides strong resilience and security properties, as an intrinsic consequence of good design principles. This is achieved by separating Autonomous Systems (ASes) into groups of independent routing sub-planes, called trust domains, which then interconnect to form complete routes. Trust domains provide natural isolation of routing failures and manual misconfiguration. More importantly within our research scope, they give endpoints strong control for both inbound and outbound traffic, provide meaningful and enforceable trust, and enable scalable routing updates.

Our research investigates the possibilities and limitations of relying on a SCION network to provide users with control on how their traffic is steered through the network. In order to achieve this user-driver path control, we need to know some properties of the underlying paths. For example, we analyze how the choice of a specific path that follows the lowest latency to a desired destination, as chosen by a user, affects the available bandwidth within a SCION network. This allows us to examine the impact on performance that shifting network control from operators to end users has on traffic.

This paper first provides an overview of an existing SCION network and its capabilities, such as applications that run on it to show how different paths are affected by latency, bandwidth and packet loss. We then present our software that leverages on these applications to build a database that contains extensive information on paths available in the SCION network we tested. This database is then queried to provide users with the best possible path they can choose for reaching a specific destination, based on performance, geographic placement of devices traversed, and operators that run them.

The rest of the paper is structured as follows. First, the concepts of SCION and the UPIN project are explained in section 2 to grasp the range of this research. Our experimental setup and its capabilities are reviewed in section 3. In section 4 the design considerations

SC-W 2023, November 12–17, 2023, Denver, CO, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0785-8/23/11.
https://doi.org/10.1145/3624062.3624592

for the implemented software are clarified. The implementation of the approaches is further detailed in section 5. Last, the path selection process and the performance analysis are explained in section 6 before making conclusions in section 7.

2 BACKGROUND

SCION [9] is an Internet architecture designed to provide route control, failure isolation, and explicit trust information for end-to-end communications to end users, thus giving endpoints strong control for both inbound and outbound traffic. It is designed to provide high availability in the presence of adversaries, trust and path transparency, and inter-domain multipath routing [8]. The SCION architecture provides strong resilience and security properties as an intrinsic consequence of good design principles, avoiding piecemeal add-on protocols as security patches. Meanwhile, SCION only assumes that a few top-tier Internet service providers (ISPs) in the trust domain are trusted for providing reliable end-to-end communications, thus achieving a small Trusted Computing Base. We will rely on a SCION network, specifically its testbed SCIONLab, to evaluate its capabilities and limitations when performing user-driven path control.

2.1 UPIN

The UPIN project provides the concrete implementation for the Responsible Internet, namely transparency, controllability and accountability for the users of the network infrastructure [3][2].

UPIN introduces an Internet framework that allows users to define a specific network behaviour. The framework utilizes existing network paradigms, for example Software-Defined Networks, to blend with other Internet infrastructures [3].

The UPIN framework consists of a Domain Explorer, Path Controller, Path Tracer, Path Verifier, and Front-end [2]. The Domain Explorer obtains metadata about properties of the network, including security and environmental details. It stores detailed knowledge on the nodes in the network. The Path Controller is in charge of setting the forwarding rules based on the desires of the user. The Controller is only able to influence the nodes in its own domain. The Path Tracer gathers measurements on the traffic in the UPIN domain. The goal is to store important details for the possible verification. The Path Verifier examines whether the desires of the user are satisfied. However, if the path traverses a non-UPIN enabled domain, the Path Verifier cannot be certain whether the intent is satisfied over the full path. The Front-end provides a method of communication between the user and the domain. Other applications of UPIN can be found in [7]. The work we present here relates closely to the Path Controller component in the framework, and investigates a new network environment where the UPIN concept can be applied.

3 EXPERIMENTAL SETUP

Network testbeds have been essential in advancing networking research and enabling scientific breakthroughs. These ad hoc environments provide researchers with a controlled platform to conduct experiments and evaluate novel network protocols, algorithms, and technologies. They are crucial to understand the intricacies of network behavior and exploring innovative solutions. However, it is worth noting that the majority of existing testbeds is intended to experiment with the current Internet. While this focus has undoubtedly yielded valuable insights and advancements, there is a growing need to broaden the scope of testbeds to encompass emerging network paradigms and technologies. For instance, next-generation networks that support new networking approaches like path-aware networking, multipath communication or novel security techniques, require specifically designed testbeds.

The network under evaluation is SCION [9]. Specifically, the idea is to explore its path-aware feature and its limitations. For this purpose, SCIONLab has been developed to enlarge research opportunities and experimentation with SCION.

3.1 Architecture: SCIONLab

SCIONLab is an architecture designed to provide a fully distributed SCION network infrastructure, made up by different Autonomous Systems organized in isolated domains. Users can define their own ASes and connect them to the SCION network, for running experiments. This global topology's main goal is to provide a variety of paths between different ASes to support multipath operations. It is worth noting that a SCIONLab AS network typically is made up by a single host, unless differently specified. Simultaneously, it operates control plane services, border routers, and end host applications, hence in this work we will interchangeably use "ASes" and "hosts" to refer to network entities.

Fig. 1 depicts the global SCIONLab topology currently available[1]. Every node in this topology represents an AS. Each AS is assigned a globally unique AS number (ASN) and a public/private key pair. This key pair is certified through the issuance of a public key certificate (PKC). The SCIONLAB network infrastructure is based on 35 ASes widely distributed across the world. SCIONLab organizes ASes into groups of independent routing planes, called *isolation domains* (ISDs), which interconnect to provide global connectivity.

There are three different types of ASes:

- Core ASes: in Fig. 1 they are light orange colored. A Core AS is the *root of trust* inside the AS, which is the entity that signs PKC of other ASes in the same ISD.
- Non-core ASes: these are standard components of the SCIONLab infrastructure, having no specific role. They are white colored in Fig. 1.
- Attachment points (AP): these are the most interesting components of SCIONLab for us because they allow users to attach their own ASes. In this way it is possible to extend the global topology with the experimenters' computational resources. In Fig. 1, they are light green colored.

Finally, there is our own AS that we had to define in order to interact with the SCIONLab network; it is light blue colored in Fig. 1.

3.2 Initialization and Configuration

In order to start experimenting with SCIONLab, we have to define one AS to attach to one endpoint. We created one AS through the SCIONLab web interface[2] and attached it to `ETHZ-AP`. We were free to choose any of the access points in the topology. We chose `ETHZ-AP` because of its centered position in the network that will

[1]Source: https://www.scionlab.org/topology
[2]Source: https://www.scionlab.org/

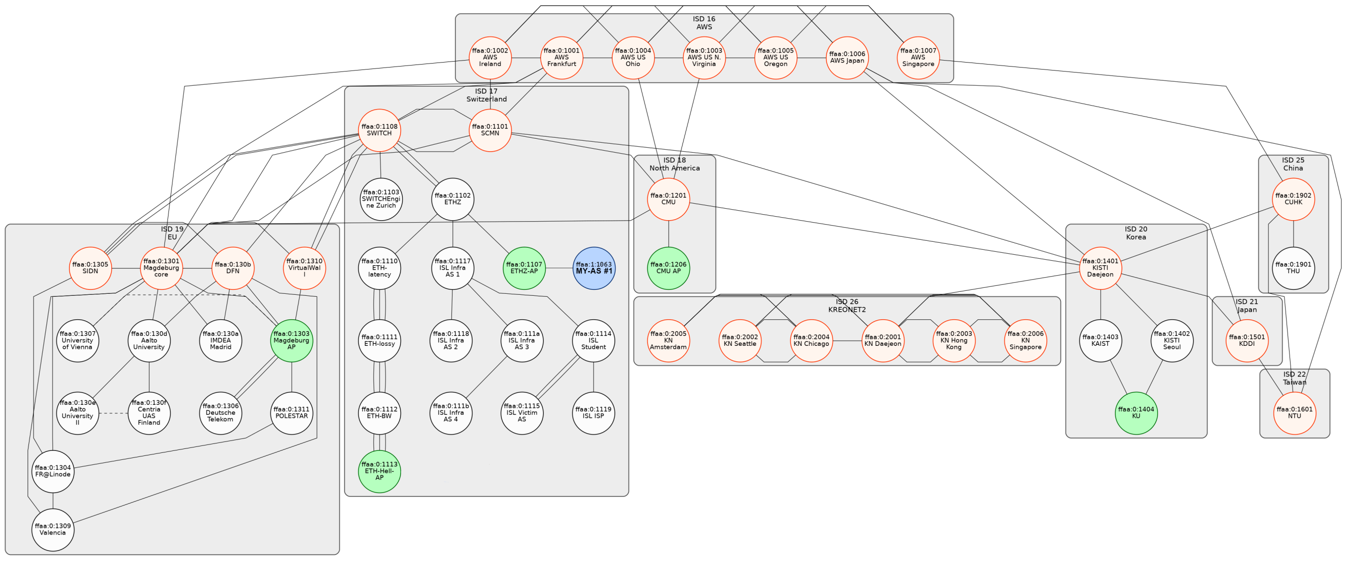

Figure 1: SCIONLab Topology: in light orange there are Core ASes; Non-Core ASes are white colored; Attachment Points are green; our AS is blue.

help us run specific experiments we will discuss in later sections. Once this configuration phase was completed, SCIONLab web interface provided a unique ASN for our AS, along with cryptographic keys and public-key certificates. Subsequently, a Vagrant file for our AS was generated to instruct the configuration of a Virtual Machine (VM) that represents our AS. This file made the setup process lightweight by automating the installation of SCIONLAB services, relevant packages, and necessary configurations. Finally we were ready to use a fully configured VM belonging to the global SCIONLab topology.

3.3 Available Applications

The VM configuration process also installs a predefined set of SCION applications. The SCION apps that we used in our experiments are:

- **`scion address`**: this command returns the relevant SCION address information for the local host, that is, our AS where we launch commands from.
- **`scion showpaths`**: it lists available paths between the local and the specified AS. By default, the list is set to display 10 paths only, it can be extended using the `-m` option. Moreover, a really useful feature for this work, is the `−extended` option, which provides additional information for each path (e.g. MTU, Path Status, Latency info).
- **`scion ping`**: it tests connectivity to a remote SCION host using SCMP echo packets[4]. When the `−count` option is enabled, the ping command sends a specific number of SCMP echo packets and provides a report with corresponding statistics. Furthermore, the real innovation is the `−interactive` mode option, which displays all the available paths for the specified destination allowing the user to select the desired traffic route.
- **`scion traceroute`**: it traces the SCION path to a remote AS using SCMP traceroute packets. It is particularly useful to test how the latency is affected by each link. Even this command makes interactive mode available.
- **`scion-bwtestclient`**: it is the only application presented in this work that is not installed by default in the VM. `Bwtestclient` is part of a bigger bandwidth testing application named `bwtester` which allows a variety of bandwidth tests on the SCION network. The application enables specification of the test duration (up to 10 seconds), the packet size to be used (at least 4 bytes), the total number of packets that will be sent, and the target bandwidth. For example, 5,100,?,150Mbps specifies that the packet size is 100 bytes, sent over 5 seconds, resulting in a bandwidth of 150Mbps. The question mark ? character can be used as wildcard for any of these parameters, in this case the number of packets sent. Its value is then computed according to the other parameters. The parameters for the test in the client-to-server direction are specified with `-cs`, and the server-to-client direction with `-sc`.

We will analyze further these scion commands and how we used them in the next section.

4 SOFTWARE DESIGN

We now present our software to test SCION features of path awareness and path selection. We will also test network performances such as: latency, bandwidth and packet loss in order to provide UPIN users with paths that fulfill requirements on these properties.

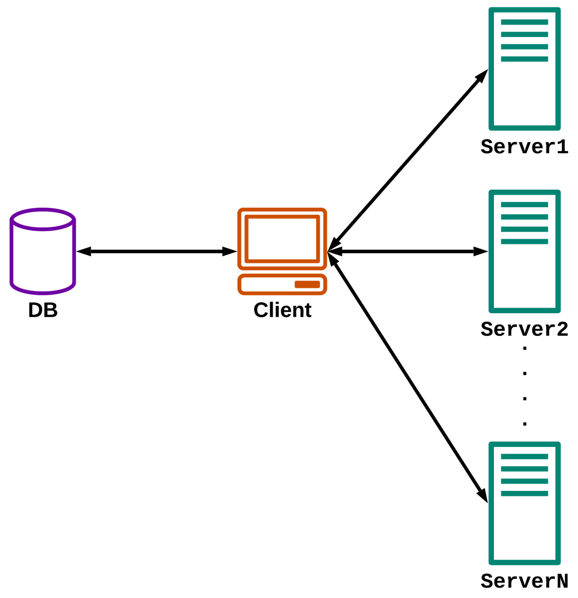

Figure 2: Overview of the software architecture: the client interacts with each server to gather information about paths and then stores them in the database.

The software relies on a 3-tier architecture: there is a client-server interaction model, along with a database where information are retrieved and stored by the client. In particular, the previously configured AS in the SCIONLab network acts as a client interacting with a pool of servers. These servers are globally distributed around the network and belong to different ISDs. The interaction model is simple: a client which wants to test paths performances to reach different destinations, performs the following actions by running the test-suite:

(1) Paths Collection: the client gathers information about all the possible paths to reach each destination and their known characteristics.
(2) Paths Test Execution: the client has to test, for each of the retrieved paths, network performances in terms of latency, loss and bandwidth available.
(3) Stats Storage: the final step is the storage of the previous statistics. One entry for each path is inserted in the database and contributes to provide samples for path analysis and evaluation.

Figure 2 provides an overview of the software architecture and summarizes the steps described above.

4.1 Technical Requirements

4.1.1 Scalability. since the test-suite is based on testing network performances, one of the most important requirements is scalability, which means the system's capability to adapt to a larger workload or user base without compromising performance, responsiveness, or reliability. The test-suite is designed to perform network measurements and evaluations, so it inevitably generates a significant amount of data. This data includes information about path performances, network statistics, and other relevant metrics collected from a multitude of test runs. The amount of data generated grows both with the number of tests performed per destination, as well as the number of destinations tested.

4.1.2 Fault Tolerance. it is the ability of the software to continue functioning properly in the presence of faults or failures. In our case, we can identify many types of failure:

- Data Loss: since the application is based on retrieving and storing data in a database, and it relies on a dynamic network, sometimes it may happen that some data gets lost due to a malfunction in the network or in the software.
- Server Failure: as our source is not the only actor in this architecture, it is not the single point of failure. Destinations can be up or down and in some cases they could not answer to our requests.
- Error Messages: it can also happen that a server is not down but it provides a bad response. As in the previous case, we should handle also this kind of responses.

Fault tolerance is a key point of this architecture since, continuous measurements require continuous functioning.

4.1.3 Portability. it is a feature of a software application that describes how easily it can be transferred or adapted to different computing environments or platforms without requiring significant modifications. Our application is intended to be working on all the SCION-based networks, with minimal modifications required. Portability problems that may arise are:

- Different Commands Specification: the whole software architecture relies on the latest SCION built-in commands available in SCIONLab, hence, there might be updates or previous versions of SCION that may not recognize the commands used.
- Flexibility to changes in metrics: in the future, a user could desire to add more metrics for the assessment of a path. These metrics must be easy to integrate in the software.

Making a software portable is essential to speed its adoption and improvements, but also to provide a plug-and-play system that requires at most minor changes.

4.1.4 Security. it plays a crucial role in this architecture, given the multitude of interactions that could potentially introduce vulnerabilities. Ensuring data integrity and authentication, managing database access, and safeguarding against Denial-Of-Service attacks are the baseline for a secure test-suite:

- Data Authentication and Integrity: as the software conducts measurements across the network and store them in a database, it is crucial to establish the legitimacy of data and verify whether any tampering has occurred.
- Database Access Management: it is equally important to perform access control to store, read and modify data. Only authorized users, following an authentication process, should be granted these privileges.
- Denial-Of-Service Attacks: The threat of DoS attacks looms large over systems that require continuous operation, as such attacks have the potential to disrupt services for considerable

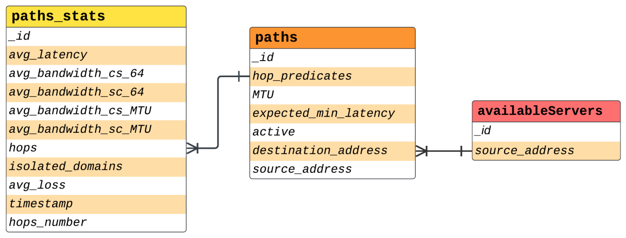

Figure 3: Database Schema presenting, from left-to-right, collection of paths' statistics, collection of each path for each server, and servers considered for the assessment.

periods. The development of a resilient architecture capable of mitigating these attacks is indispensable.

We consider security into every phase of the development process, starting from the design stage to avoid undesired consequences and future complications.

4.2 Design Choices

4.2.1 Database Design. Since the software is based on retrieving and store data efficiently, the first thing to design was the database and take all the decisions needed to achieve good performance. For this purpose, the choice fell on a non-relational database (DB) because of two considerations:

- Massive dataset organization: a non-relational database can easily store huge quantities of data and query them, since it uses horizontal scaling and distributed architecture to handle large datasets efficiently.
- Flexible database expansion: as networks are dynamically changing, so is our data, and a non-relational DB can absorb new data points, enrich the existing database with new levels of granularity and extend previous data.

Among non-relational DBs, we selected MongoDB for its usability and performances. For the sake of clarity, groups of data logically recalling the same concept are named *collections*, they hold the same role of tables in a relational database, though in MongoDB, data structures belonging to the same collection can be heterogeneous. An entry in a collection, is instead called *document*. The designed database is composed of the following collections:

- **`availableServers`**: it stores information about servers with which the test-suite can connect and perform tests. It is a subset of 21 of all the 35 servers in the topology. Notably, the norm in SCIONLab is for each AS to house only one server; however, certain ASes contain multiple servers. We later present them as different destinations.
- **`paths`**: this collection holds the information gathered for each path to each destination in availableServers, it includes the list of hops but also known characteristics.
- **`paths_stats`**: it collects all the statistics gained after the test-suite runs but also further information related to the path (e.g. ISDs traversed, number of hops, ...).

The database schema, along with documents' structure, is shown in Fig. 3.

"AvailableServers" collection has 2 fields: server's source IP address, along with an id. This identifier is a progressive integer and in our case it can be a number between 1 and 21, since we only have 21 destinations fully testable in our topology.Each *"paths"* document, instead, has its own identifier (_id), that is built by combining the server id and a progressive number for the path (e.g. a path whose id is **2_15** identifies the path **15** of the destination **2**). Other fields are used to describe some known properties about the path. Finally, each document of *"paths_stats"* collection has its own identifier built by combining the path identifier with a timestamp, in order to identify the measurement in time over a specific path for a specific destination. Other fields are performances related, such as: average latency, average loss or average bandwidth in upstream and downstream, considering packets of only 64 bytes or packets of MTU size.

4.2.2 Technical Requirements Design. we made several choices in order to achieve scalability, fault tolerance, portability, and security properties. The choice of a non-relational DB can strongly improve scalability in querying and storing operations as well as in distributing data automatically. Another choice was reducing I/O operations' overhead by preferring multiple insertions of path statistics to single ones. There is a trade-off between fault tolerance and scalability in terms of insertions. Preferring multiple insertions means also that if a crash happens all the statistics are lost and not saved. On the other hand saving one measurement at time decreases performances dramatically and makes the system less scalable. We decided to insert all the measurements after testing once all the paths for one destination. In this way, a loss of data can be negligible since one sample for each path would be lost without unbalancing the number of samples for each path; this leads to a growth in fault tolerance. Moreover, since nodes can be up and down and sometimes they might be unreachable, the software architecture was provided with error handling to reduce crashes and keep the system working with a dynamic and fallible network. Regarding portability, the software uses commands available in SCIONLab, therefore its usage over a different SCION based network may require a few adjustments to adapt them, though the whole architecture would mostly be the same. Furthermore, the choice of MongoDB enables effortless modification or addition of metrics, resulting in a highly extensible and portable system.

In terms of security, many solutions have been designed, though some of them are not implemented yet. The primary focus has been given to:

- Database Access Management: the first thing to constraint is database access, particularly during the statistics' saving operation, to avoid fake performances injection that may alter analysis and provide misleading results. A possible way of doing this is the usage of public key certificates to get write access to the DB.
- Statistics Authentication and Integrity: similarly, also produced measurements should be authenticated with a PKC to provide data integrity and authentication.We assumed that data authentication was granted by the development of the SCIONLab commands that our software relies on.

- Denial-Of-Service: DoS attacks can be really hard to manage. Fortunately, SCION's inherent properties offer some relief by minimizing the likelihood of DoS attacks [9].

5 IMPLEMENTATION

The test-suite is composed of three components: a shell script and two python scripts. Precisely, the shell script that we used is written in Bash *(Bourne Again Shell)*, which is the default shell for most Unix-like systems, including Linux. Bash scripts are versatile and widely used for various automation tasks.

5.1 Bash Script

The bash script, `test_suite.sh`, can be seen as a container or wrapper of the python scripts. It provides a command line interface (CLI) to the user and execute other units to test each path and store the results. From the CLI, the user specifies the execution parameters and options necessary to define the behavior of the application at run-time.

We can observe three main parameters:

- **`<iterations>`**: it is an integer that specifies the number of times that tests must be executed for each path. Its value is propagated to the **`run_tests.py`** script.
- **`–skip`**: optional argument used to bypass the collection of paths to each destination and speed up the test execution. Its usage is meaningful only if paths have already been collected and have not changed.
- **`–some_only`**: another parameter useful to accelerate testing. Its application constraints the test execution to only the first destination. Therefore, all the paths of the first destination in the `availableServers` collection will be tested `<iterations>` times.

This script, as well as the rest of our software suite, is available and can be found in the following GitHub repository [1]. An example of its usage could be the following:

```
./test_suite.sh 100 --skip
```

The result will be the execution of the tests, for each path available, 100 times and skipping the path collection phase.

5.2 Paths Collection Script

One of the two sub-units of the bash script is the `collect_paths.py` component. It serves as the first internal script with the purpose of gathering information about all the paths available to reach each destination. Its purpose is to populate the paths collection as shown in Fig. 3. The user does not interface directly with it, since the only part visible is the external wrapper. Even if its main role is to discover paths, it also performs other operations like data pre-processing, data insertion and deletion:

- Paths Collection: the main goal of this script is to discover and retrieve paths information to reach the desired destinations stored in the `availableServers` collection. Hence, the initial step involves querying the database and collect the set of destinations to test. For each of them the application spawns a sub-process that runs the SCION command:

  ```
  scion showpaths --extended -m 40
  ```

  This command provides a maximum of 40 paths for each destination, ranked by hop count, along with all their details: hops predicates (hops traversed in the path), MTU, path status and minimum latency expected over the path. From this output, we have decided to retain only paths with a number of hops at most equal to the minimum required plus one. This selection strategy is aimed at conserving time by excluding paths that are overly lengthy and fail to meet our latency criteria.
- Data Pre-processing: because the output of the `showpaths` differs from what is expected as input for the testing commands, a parsing operation is required before storing the paths in the db.
- Data Storage: once data have been correctly pre-processed, paths are inserted in the database and no longer available paths for one destination are deleted.

5.3 Tests Execution Script

This last script, known as `run_test.py`[3], represents the core of the whole application. There are 3 nested `for` loops, used to run tests over each path, for each destination, "`iterations`" number of times. Three functions are invoked, each of them generates a sub-process which tests one or more metrics of the path by running the respective SCION command. Specifically, the three sub-processes perform the following actions:

- Latency and Loss Measurement: this operation is performed by the first sub-process which executes the following command:

  ```
  scion ping {server_address} -c 30 --sequence
  '{hop_predicates}' --interval 0.1s
  ```

  The destination is reached using SCMP packets, measuring the latency and the packet loss. We set the interval between each packet to 0.1s and we send 30 packets. The destination address and the sequence of hops determining the path to test, are dynamically set at each iteration of the three nested loops. The output values of the ping command are: the average latency measured by the 30 packets sent in milliseconds and the packet loss percentage.
- Bandwidth Measurement with 64 bytes Packets: the second sub-process executes the following command:

  ```
  scion-bwtestclient -s {server_address} -cs
   3,64,?,12Mbps -sequence '{hop_predicates}'
  ```

  This is the bandwidth tester application available in SCIONLab. Other than destination and hop predicates, we add the time interval for which the bandwidth needs to be achieved (*3s*), the packet size to send over the path (*64 bytes*), a wildcard for the number of packets automatically computed by the application, and the desired bandwidth to achieve (in this case *12Mbps*). We defined these parameters for the client-server measurement only and, by default, they are used for the server-client too, resulting in 2 average bandwidths to be saved.

[3]Can be found under the `Test` folder in the root directory.

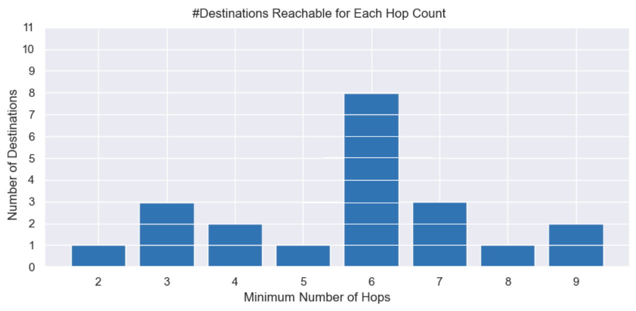

Figure 4: Server Reachability from `MY_AS#1`. In blue is displayed the number of destinations reachable requiring minimum a certain hop count.

- Bandwidth Measurement with MTU bytes Packets: this is the last operation performed by the third sub-process before the storage of the measurements. It executes the following command:

  ```
  scion−bwtestclient −s {server_address} −cs
  3,MTU,?,12Mbps −sequence '{hop_predicates}'
  ```

  The command is similar to the previous one but now sending packets with the size of the maximum transmission unit, resulting in two more average bandwidths, one per destination, with MTU sized packets.

Another information we provide and store in the DB is the set of ISDs traversed. It can be interesting to enlarge the context of measurements with the ISDs and see if they influence somehow the performance, and allow for one more interesting constraint to be set by the end user at time of choosing one path. We have now gained all the desired information about our path, including latency, loss, bandwidth, and the set of traversed ISDs. The final step involves storing this data. This operation takes place after each path has been tested for a specific destination. As previously discussed in the Technical Requirements Design subsection (4.2.2), this storage approach enhances fault tolerance and reduces overhead in I/O operations.

6 PATH SELECTION

We perform a preliminary analysis about the reachability on the full server set. There are 21 reachable destinations. Fig. 4 depicts the number of destinations that require a minimum number of hops to be reached. It offers valuable information regarding the average path length and the distribution of servers across the network. Notably, the average path length is 5.66 hops and about 70% of paths can be reached within 6 hops. This two data highlight the central position of our AS within the server distribution. This insight can easily be confirmed by looking again at the topology in Fig. 1. In order to analyze further the performance of this network, we present our experiments run on a subset of 5 destinations from the 21 available in SCIONLab.

These servers were selected from different geographical locations and different ISDs. The idea is to assess how much the geographical position and the belonging to a specific ISD can influence performance. Hence, we picked servers placed in the following countries: Germany, Ireland, North Virginia, Singapore and Korea.

In the course of this analysis, the test-suite gathered a substantial dataset comprising approximately three thousand samples. This wide volume of data provides a robust foundation for our following analysis, offering meaningful insights into path performance regarding latency, bandwidth and path loss.

6.1 Latency Assessment

Latency refers to the time delay between sending a data packet from a source host to a destination one, and receiving a response. It measures the round-trip time for data to travel between two locations in a network. For instance, low latency is crucial for real-time applications like video conferencing and online gaming.

We chose whisker plots to visually represent the distribution of latency values and highlight key statistical measures in a concise manner. A whisker plot is graphical representation useful to display the distribution of a dataset along with its central tendency and variability.

Firstly, our assessment focused on the evaluation of average latency values for each path leading to the five destinations. Fig. 5 illustrates the whisker plots of latency values for each path of destination `16-ffaa:0:1002,[172.31.43.7]`, which stands for the Ireland AS. On the x-axis, there are the path identifiers of routes having a number of hops less than or equal to the minimum plus one; while, on the y-axis, there are the average latency values. Hence, paths are categorized into two groups: 6 hops paths (in red) and 7 hops paths (in purple).

The most interesting aspect in this graph is the clear separation of latency values into three main layers, each with nearly the same average values. From an accurate analysis of paths “10” and “15” we have observed that the second-last hop of both paths is placed in Ohio, USA, while paths “9” and “14” deviate through an AS in Singapore. All the other nodes are located in Europe. This observation suggests that paths with geographically diverse hops have a more significant impact on latency than the sheer number of hops.

The relation between hops location and latency is very useful to determine which paths should be discarded in a path selection based on low latency routes.

We performed a further analysis on latency by grouping, for each destination, paths traversing the same set of isolated domains and having the same hop count. Fig. 6 shows a graph describing this analysis. The x-axis provides the different sets of ISDs traversed to reach the destination (AWS Ireland `16-ffaa:0:1002,[172.31.43.7]`). The interesting insight here is that only the hops number is not enough to determine the latency variance or increment. By looking at the second column of the graph on the left side, which considers the same set of ISDs but with paths having one more hop, we can see a much bigger gap in latency values. This may lead us to think that latency is affected also by the number of hops. However, if we take out long distance paths like those passing through Singapore or Ohio (which are geographically far from the destination in Ireland) we can observe a smaller variance and comparable values, as observable from the graph on the right side of Fig. 6.

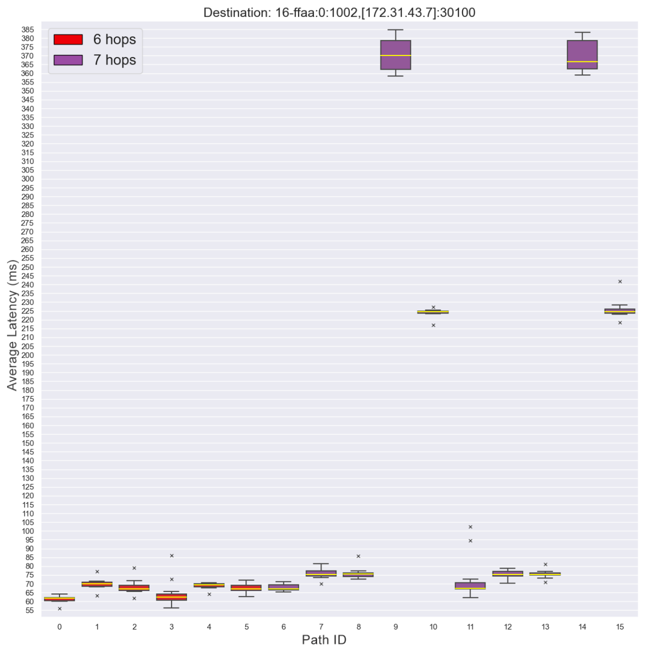

Figure 5: Average Latency Values measured for each path of destination `16-ffaa:0:1002,[172.31.43.7]` (AWS - Ireland). Box plots are split into 6 hops paths length, in red, and 7 hops paths length, in purple.

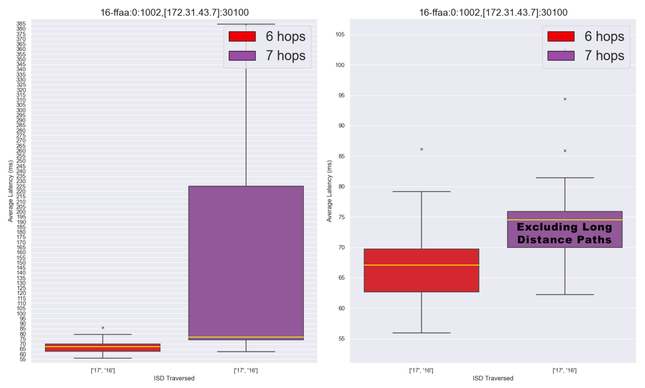

Figure 6: Average latency for each ISD set grouped by hop count. On the left side, the plot includes all the measurements. On the right side, long distance paths have been excluded from the second ISDs set.

Hence, the physical distance between hops confirms to be the predominant component in the latency assessment. Moreover, it is worth noting that by removing long distance paths we do not only get a latency reduction but also a more compact box plot. Therefore, it seems that ASes `16-ffaa:0:1007` and `16-ffaa:0:1004` introduce a wide jitter other than high latency peeks. This assessment helps us to exclude routes passing through these ASes for streaming audio and video services, as well as, for example, VoIP calls,

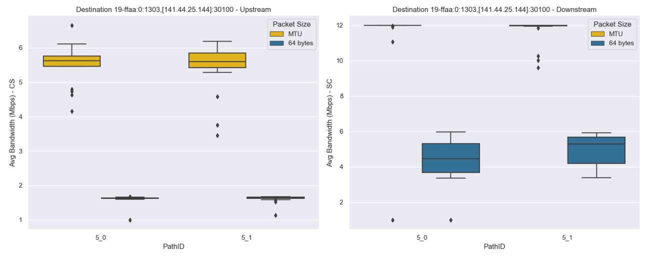

Figure 7: Average bandwidth values for each path, requiring a bandwidth of 12Mbps from and to a Germany Server (address on the top). On the left side there are the upstream measurements, while on the right side the downstream ones.

in which latency consistency is more important than low latency values.

6.2 Bandwidth Assessment

The second parameter to evaluate for a conscious path selection is the bandwidth. It represents the capacity or throughput of the channel and is typically measured in bits per second (*bps*). We conducted two distinct tests: one with a lower bandwidth requirement of 12Mbps, and another requiring for 150Mbps. Each evaluation involved two scenarios: utilizing packets of both MTU size and 64 bytes size, and examining interactions from both client to server and server to client. The aim of this approach was to comprehensively assess network behavior from various angles, including upstream and downstream perspectives, as well as under different conditions such as high bandwidth demands, small packet transmission, and average usage at 12Mbps. Considering the first test at 12Mbps, we achieved a consistent trend across all five destinations. Fig. 7 depict the average bandwidth values tested for the Magdeburg AP in Germany (AS `19-ffaa:0:1303,[141.44.25.144]`. The two graphs illustrate the distribution of bandwidth for each path (measured in Mbps), showcasing downstream measurements on the right and upstream measurements on the left. Additionally, each path is represented by two whiskers: the yellow whisker corresponds to bandwidth values obtained using MTU-sized packets, while the blue whisker represents values obtained with 64-byte packets.

As we would expect, in upstream they achieve a lower bandwidth compared to the downstream counterpart. This phenomenon is in line with the internet's inherent asymmetry, where user data consumption typically exceeds data upload. Moreover, all the paths get a lower bandwidth by sending 64-byte packets compared to the MTU packets. This is an expected outcome, as using smaller packets increases the total packet count, subsequently amplifying the overhead of packet headers.

This trend reverses when we require a higher bandwidth of 150Mbps, highlighting the limitations of bandwidth in the SCIONLab network. Fig. 8 shows the average bandwidth for the same destination but with a requirement of 150Mbps.

In Fig. 8 we observe a higher achieved bandwidth by sending smaller packets instead of bigger ones. This looks counter-intuitive because of the overhead of packet headers. Hence, the network may not have sufficient capacity to handle the desired 150Mbps

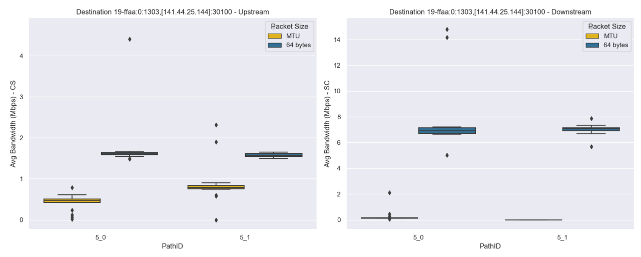

Figure 8: Average bandwidth values for each path, requiring a bandwidth of 150Mbps from and to a Germany Server (address on the top). On the left side there are the upstream measurements, downstream on the right.

bandwidth for MTU-sized packets. Even though MTU-sized packets are more efficient in terms of payload-to-overhead ratio, the network may be congested or have limited capacity, causing packet drops and resulting in a lower achieved bandwidth. Indeed, dropping 64 bytes packets does not decrease the achieved bandwidth as dropping MTU-sized packets. This is an insight that requires further analysis even though it is suggested by all the destinations involved.

6.3 Packet Loss Assessment

When a data packet is sent from a source to a destination, it traverses various network devices and links. If any of these devices or links are overloaded, faulty, or experiencing high traffic, packet loss can occur. Monitoring and managing packet loss is essential for maintaining network reliability and performance. Hence, we want to assess this phenomenon to prevent a user by choosing routes with a high packet loss ratio. Fig. 9 shows the average packet loss percentage for each path available to reach AWS destination in Northern Virginia, USA (AS `16-ffaa:0:1003,[172.31.19.144]`). Each path is represented with a different colored dot[4]. The dot size stands for the number of measurements having the same packet loss ratio. The majority of paths exhibits a loss ratio of 0%, with a few instances occasionally reaching almost the 10% mark. However, within this context, there are particular paths that notably register a complete 100% loss rate, as evidenced by paths 2_16, 2_17, 2_18, 2_19, 2_22, and 2_23. These instances of complete loss merit our attention, warranting exploration into potential factors at play, such as network congestion. By looking at the sequence of hops for each of these paths, a commonality emerges: the shared nodes are only those concentrated in the first half of the path. Moreover, since these measurements were carried out in succession (due to the consecutive nature of the paths), our hypothesis is that one or more of these common nodes experienced a period of congestion.

7 CONCLUSION

In order to achieve user-driver path control, we need to know several properties of the underlying network and what paths we can build in it. We investigated the possibilities and limitations of SCIONLab for user-driven path control. We explored several aspects

[4]After an interval of 9 paths, colors are re-used.

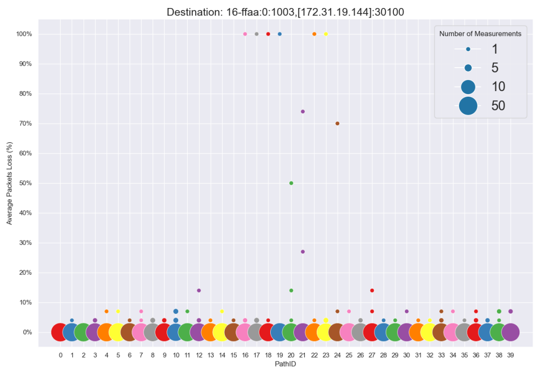

Figure 9: Average packet loss percentage for each path of AWS US N. Virginia AS. Each dot color represents a path and its size the number of measurements having the same loss ratio. Dots legend is on the upper right corner.

of its performance, specifically focusing on latency, bandwidth and data loss, that left us with interesting insights, showcasing the role that the test-suite can play in fostering informed and thoughtful path selections. The software design is based on a database where we store data gathered on many paths, that we then query to select the best one to give to a user. This is a crucial step in our investigation on the consequences that shifting control from operators to end users of a network has on its performance.

Specifically, we confirmed that latency in SCIONLab is affected mostly by the physical distance among the nodes building the path, rather than and that the number of hops or the ISDs traversed. A more interesting aspect we measured was related to the bandwidth limitations of the SCIONLab network, where the capacity decreases when trying to target a higher bandwidth from a path. We intend to investigate this behaviour further in the future. Packet loss appears to be stable in most cases.

Finally, the path selection feature of SCION, when coupled with a robust test-suite and data analysis techniques, blends into a powerful tool that helps to fulfill the controllability requirement of a UPIN user. We intend to proceed with our goal to satisfy this mandate not only through the implementation of a test-suite, but also by providing a user interface and a path recommendation feature, that remains our main direction for future research.

ACKNOWLEDGMENTS

This research received funding from the Dutch Research Council (NWO) under the project UPIN.

REFERENCES

[1] Antonio Battipaglia. 2023. SCION Test Suite. https://github.com/MrR0b0t14/SCION-Test-Suite
[2] Rodrigo Bazo, Leonardo Boldrini, Cristian Hesselman, and Paola Grosso. 2021. Increasing the Transparency, Accountability and Controllability of multi-domain networks with the UPIN framework. In *Proceedings of the ACM SIGCOMM 2021 Workshop on Technologies, Applications, and Uses of a Responsible Internet*. 8–13.
[3] Leonardo Boldrini, Rodrigo Bazo, Cristian E.W. Hesselman, and Paola Grosso. 2021. UPIN - A shift in network control from operator to end user. ICT Open 2021.

[4] Laurent Chuat, Markus Legner, David Basin, David Hausheer, Samuel Hitz, Peter Müller, and Adrian Perrig. 2022. *The Complete Guide to SCION.* Springer.
[5] Tomi Dufva and Mikko Dufva. 2019. Grasping the future of the digital society. *Futures* 107 (2019), 17–28.
[6] Cristian Hesselman, Paola Grosso, Ralph Holz, Fernando Kuipers, Janet Hui Xue, Mattijs Jonker, Joeri de Ruiter, Anna Sperotto, Roland van Rijswijk-Deij, Giovane Moura, et al. 2020. A responsible internet to increase trust in the digital world. *Journal of Network and Systems Management* 28, 4 (2020), 882–922.
[7] Anne-Ruth Meijer, Leonardo Boldrini, Ralph Koning, and Paola Grosso. 2022. User-driven Path Control through Intent-Based Networking. In *2022 IEEE/ACM International Workshop on Innovating the Network for Data-Intensive Science (INDIS).* IEEE, 9–19.
[8] Adrian Perrig, Pawel Szalachowski, Raphael M Reischuk, and Laurent Chuat. 2017. *SCION: a secure Internet architecture.* Springer.
[9] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David G. Andersen. 2011. SCION: Scalability, Control, and Isolation on Next-Generation Networks. In *2011 IEEE Symposium on Security and Privacy.* 212–227. https://doi.org/10.1109/SP.2011.45