MS4PAN: Measurement System for Path-Aware Networks

1st Leonardo Boldrini
University of Amsterdam
Amsterdam, The Netherlands
l.boldrini@uva.nl

2nd Antonio Battipaglia
Amazon
Dublin, Ireland
antobtt@amazon.com

3rd Cristian Hesselman
SIDN Labs and University of Twente
Arnhem, The Netherlands
cristian.hesselman@sidn.nl

4th Paola Grosso
University of Amsterdam
Amsterdam, The Netherlands
p.grosso@uva.nl

***Abstract*—Path-Aware Networks (PANs) allow applications to choose the network paths their data traverses. However, they require application developers to translate their performance requirements (e.g., in terms of latency, throughput and data loss) to a set of paths available through a PAN, which can be a cumbersome process especially when dealing with multiple PANs at the same time. In this paper, we propose MS4PAN, a tool for app developers to select the best paths based on app-specific requirements in Path-Aware Networks. We investigate several network requirements for different use cases as well as the difficulties for app developers to interact with different network architectures. We provide a tool that automatically finds the best paths that fulfill a set of requirements. We evaluate MS4PAN in SCIONLab, an experimental testbed and implementation of a SCION network. Our software helps developers by providing a simple abstraction while dealing with different network architectures.**

I. Introduction

Citizens and governments depend on digital technologies that are severely entangled in the main structure of society [1]. These technologies are built on the traditional Internet architecture and therefore inherit some of its limitations, such as the lack of user control of the network, and a consequent erosion of trust [2]. The Responsible Internet paradigm wants overcome these problems by improving the Internet transparency, accountability and controllability [2]. The UPIN (User-driven Path verification and control in Inter-domain Networks) project, based on the notion of Responsible Internet, provides a framework for users to influence the flow of traffic through the network [3] while integrating with the current Internet architecture. This can be achieved by leveraging on Path-Aware Networks (PANs).

PANs enable end users to choose the path that is then embedded in the header of data packets. The possibility of path selection enables end users to choose paths based on their applications' requirements, e.g., low latency for remote surgeries and high throughput for file transfers.

While these architectures provide a way for users to choose a path for their traffic, it is difficult for app developers to utilize these tools. Every architecture provides a different API to interact with them and this forces developers to know syntax and details of all these APIs. This poses a limitation for the adoption of these architectures. Furthermore, even when working with one specific PAN, satisfying the requirements of one application translates into dealing with many different paths, and path properties change over time. This makes the job of developers working with PANs challenging.

To overcome these limitations, we propose MS4PAN (Measurement System for Path-Aware Networks), the first tool for developers that supports different PAN architectures. MS4PAN deals with the complexity of different network architectures while providing app developers a single, easy to use interface to set the requirements of their apps. MS4PAN constantly probes the underlying network architectures it works on top of in order to store updated information on the performance of every path to a specific destination. This information is stored in a database and it is quickly queried for monitoring of performance. In case a path experiences a change in performance, due to the dynamic nature of the network, an application is assigned a different path automatically, so that applications can respect stringent requirements on their traffic at all times. Finally, MS4PAN provides a user friendly interface to interact with to ease the work of developers and avoiding complex tuning for every PAN.

To this end, we make the following contributions:

- We investigate network requirements from different applications as well as difficulties for developers to manage paths when interacting with PANs.
- We present the design, implementation and evaluation of MS4PAN, a tool for developers to easily set requirements for the traffic of their apps and that works on top of different PANs.

The remainder of this work is structured as follows: in Section II we analyze the network requirements from different applications and the challenges faces by developers when dealing with many paths in PANs. After discussing related work in Section III, we present the design of MS4PAN in

Section IV. Afterwards, we present our implementation of this tool in Section V, followed by its evaluation in Section VI and conclusions in Section VII.

II. Requirements

We analyzed requirements of different use cases, including Wireless sensor and actor networks [4], 5G mobile services and their KPIs [5], [6], Vehicle-to-Everything (V2X) communication systems [7], Survivable network systems [8], [9], critical infrastructures that underpin modern society [10], e-health services [11].

Most requirements that these use cases have from the underlying network they rely on are performance related. They are expressed in terms of `maximum latency`, that is the time it takes for a packet of data to travel from the source to the destination, measured in milliseconds (ms), `minimum throughput`, that refers to the amount of data transmitted in a given period, measured in bits per second (bps), `maximum packet loss rate`, the percentage of packets that are lost during transmission.

Another common finding in our research as a requirement for these use cases is reliability. It refers to the ability of a network to consistently perform its intended functions under normal conditions without failure. It is a key measure of a network's robustness, indicating how dependable and resilient it is over time.

A network paradigm that opens the door to end users to define their own path is Path-Aware Networking. Path-Aware Networks (PANs) enable end users to choose the path that is then embedded in the header of data packets. The possibility of path selection enables PANs' users to choose paths based on their applications' requirements. This comes with its own challenges: in the past two decades, several PAN architectures have been introduced [12], including Platypus [13], [14], PoMo [15], NIRA [16], Pathlets [17], NEBULA [18], and SCION [19].

When interacting with each of these PANs, app developers need to interact with low level network applications that are specific to that PAN. This includes knowledge on the syntax of specific commands and details on many different APIs that PANs define to use them. Moreover, also in the simple case of an application relying on one single PAN, app developers find themselves working with many paths that this PAN provides. This happens because in order to satisfy network requirements that applications have, multiple paths need to be constantly evaluated at all times for ensuring reliable performance. This poses more challenges for app developers and it becomes a reason for the slow adoption of PANs.

We want to provide a tool that allows app developers to fulfill the requirements set by applications belonging to different use cases. In this paper, we will consider the requirements on network performance highlighted in one specific e-health service [11]. This choice is only made as an example and has no consequence on any consideration that will follow. We report here what we will commit to satisfy in the rest of this paper: a maximum latency of 200 ms, a minimum throughput of 6 Mbps and a maximum packet loss rate of 2%. We also fulfill the reliability requirement by making sure that these performances are met in real time.

III. Related Work

Before the advent of PANs, some network technologies such as RSVP-TE (Resource Reservation Protocol - Traffic Engineering) already provided similar functionalities to PANs, in particular bandwidth and latency guarantees [20], [21]. Despite their success, they are generally not usable by application developers because they were intended to be controlled by operators.

More recently, a Path Aware Networking Research Group (PANRG[1]) has been proposed within the Internet Research Task Force (IRTF) to discuss various aspects of PANs with the Internet standards community [22]. While PANRG discusses the development of PANs, which provide a way for users to control paths, users also need easy-to-use methods to be informed on what paths are most suitable for their applications.

In contrast, tools such as performance Service-Oriented Network monitoring ARchitecture (perfSONAR) are deployed and already in use to establish end-to-end usage expectations [23]. There are thousands of perfSONAR instances deployed worldwide, many of which are available for open testing of key measures of network performance. However, these tools are designed for monitoring on the Internet and not tailored to the functionality of PANs.

Other systems such as the Software-Defined Network for End-to-end Networked Science at Exascale (SENSE) [24] are designed for science applications to express their requirements, also in terms of performance. These systems are however limited to the applications they can serve, as they operate only within their domain and are not designed to be implemented with different PANs.

FABRIC [25] is an international infrastructure that enables experimentation and research at-scale in the areas of networking, cybersecurity, distributed computing, storage, virtual reality, 5G, machine learning, and science applications. It allows to set up worldwide networks where many performance measurements can be carried out. While FABRIC is a great tool for researchers in many fields, it is also a closed infrastructure not useful for applications that want to interact with PANs in the Internet.

MS4PAN bridges the gap between the PAN and its user. To the best of our knowledge, the tool that we provide to aid app developers in interacting with PANs and guarantee performance to satisfy their app requirements is the first of its kind.

IV. Architecture

We now present the architecture of MS4PAN, all of its components and how they interact with each other as well as where they stand between a PAN and an application that uses it. Fig. 1 shows the architecture of our system and the order

[1]PANRG: https://datatracker.ietf.org/rg/panrg/about

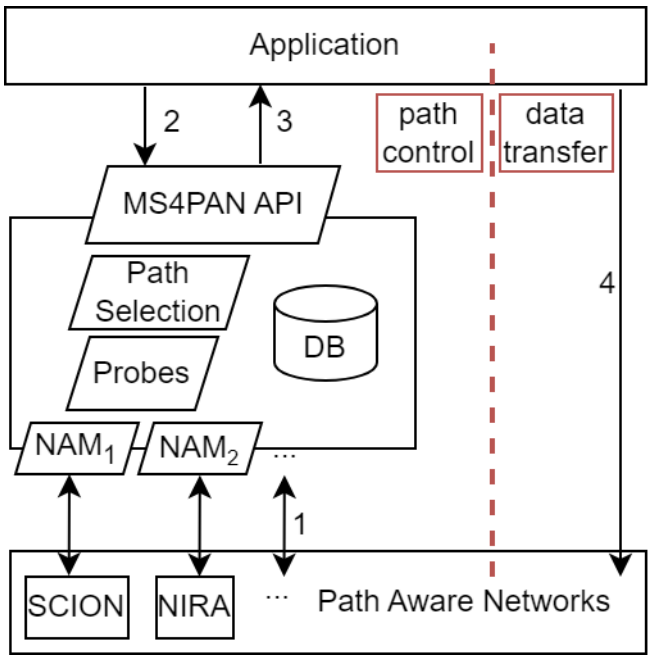

Fig. 1: MS4PAN software architecture. All the components of the software lie between the application and the PANs. The red line separates the path control from the data transfer. The numbers represent the order of operation.

of operation of each of its components. The red line divides the picture in 2 parts, on the left the path control where our software lies, on the right the data transfer.

The bottom part of the picture represents Path-Aware Networks. These include the PANs already mentioned Section II and more. The top part of the picture represents an application that intends to use these networks for its traffic. When this application is designed to work with our tool, it interacts with it first and then sends traffic.

MS4PAN and all of its components sit between the network and the application layers. Our tool interacts with different PANs, and because every PAN provides a different API to work with it, MS4PAN has a Network Abstraction Module (NAM) for each PAN. These components are network specific, they interact directly with the network layer. PANs offer us to choose among a set of different paths available to reach each destination. Every time we want to measure the performance or other aspects of each path, it is ultimately up to the NAM to execute the command for its specific PAN to test that path.

The rest of our tool is network agnostic, as it builds on top of the NAMs to offer an API for the application to select only paths that satisfy its requirements. Probes are used to test paths over time in order to have always updated information on the state and the properties of each path. We use probes to test path characteristics such as latency, throughput and packet loss as experienced by the application. Probes are also used to gather information on paths that do not regard performance, but is still useful to satisfy requirements of some applications. For example, wherever possible we gather information on the geographic location of the ASes traversed by a path. More probes can be implemented in case applications have new requirements.

We store the information gathered by the probes into a database. Since the probes need to retrieve and store data efficiently, this database plays a crucial role in our software in storing information on all available paths to reach the possible destinations.

The Path Selection component queries the database according to the requirements set by the application. It also implements a multi objective optimization algorithm to select the best paths out of the pool available that satisfy the set of requirements.

The last component of our software is an API that raises the abstraction level for app developers when interacting with our system. Applications in fact only need to deal with this component to make use of the full potential of our tool.

The order of operation is as follows, and the numbers correspond to the ones in Fig. 1:

1) The first interaction happens between MS4PAN and the network that lies beneath it. The probes measure the performance of multiple paths to reach a set of destinations at the same time. The database is then populated and ready to be queried for path analysis.
2) The application interacts with the API of MS4PAN to state the requirements on the data it needs to transmit.
3) The tool returns the best paths selected according to those requirements and the command to interact with the PAN to use them.
4) The application can then send its traffic through the network choosing the best (set of) path(s). In case the requirements are no longer fulfilled, the whole process restarts.

V. Implementation

We now discuss our implementation of MS4PAN using SCION as an example of a Path-Aware Network. Our software is available and can be found in the following GitHub repository [26]. We chose SCION because of its recent and widespread adoption and because it offers functionalities useful for our NAM implementation. SCION [19] is an Internet architecture designed to provide endpoints strong control for both inbound and outbound traffic. It is developed to ensure high availability in the presence of adversaries, trust and path transparency, and inter-domain multipath routing [27]. Specifically, we used SCIONLab, a worldwide testbed that allows us to run experiments with the SCION architecture.

A. SCIONLab

SCIONLab is a large-scale testbed designed to provide a fully distributed SCION network infrastructure, made up by different Autonomous Systems organized in isolated domains.

Fig. 2 depicts the global SCIONLab topology currently available [28]. Every node in this topology represents an AS. It is based on 35 ASes widely distributed across the world. SCIONLab organizes ASes into groups of independent routing

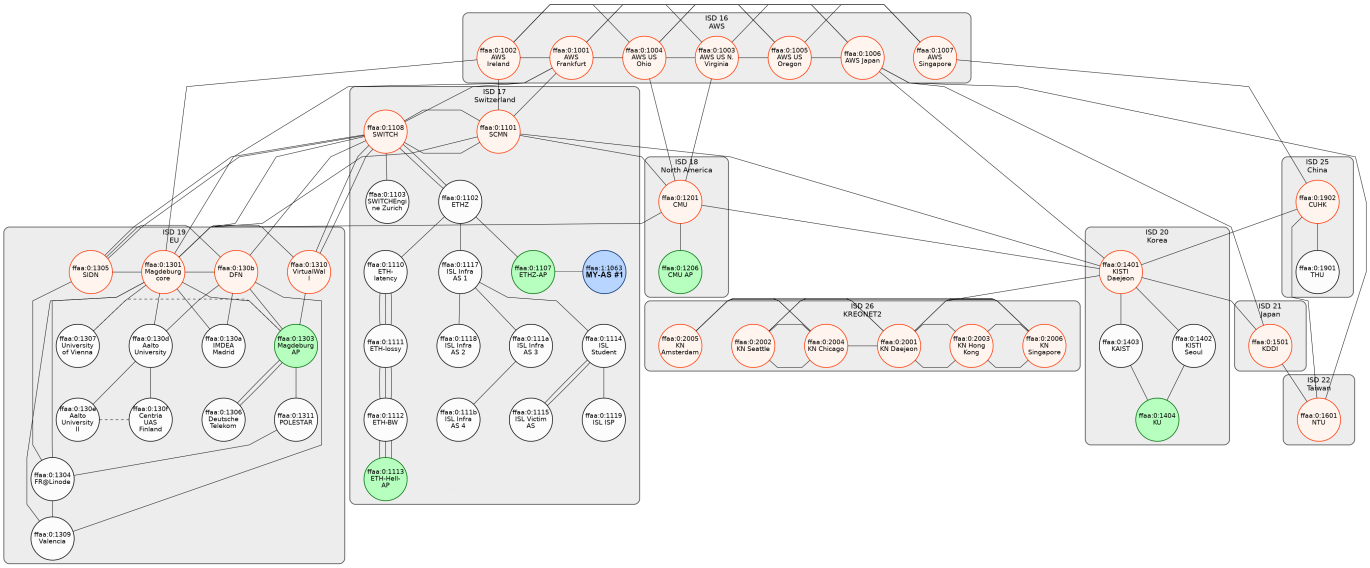

Fig. 2: SCIONLab Topology [28]: Core ASes are colored light orange; Non-Core ASes are white; Attachment Points are green; our AS is blue.

planes, called isolation domains (ISDs), which interconnect to provide global connectivity.

A Core AS, light orange colored in Fig. 2, is the *root of trust* inside the AS, which signs public key certificates of other ASes in the same ISD. Non-core ASes, instead, are white colored. Attachment points (AP) allow users to attach their own ASes to extend the global topology with the experimenters' computational resources; they are light green colored in Fig. 2.

Finally, there is our own AS, colored light blue, that we set up to interact with the SCIONLab network.

B. SCION NAM

In order to interact with SCIONLab, our NAM uses SCION specific applications, such as `scion address`, `scion showpaths`, `scion ping`, `scion traceroute`, `scion-bwtestclient`. These applications are powerful tools but also require knowledge on their specific syntax: for instance, the scion-bwtestclient takes the following arguments:

```
scion-bwtestclient -s server_address
-cs 3,MTU,?,8Mbps -sequence 'AS_list'
```

Other than the destination (*server_address*) and the ordered list of ASes for a path (*AS_list*), we need to specify the time interval for which the throughput needs to be achieved (*3s*), the packet size to send over the path (*MTU*), a wildcard (*?*) for the number of packets automatically computed by the application, and the desired throughput to achieve (in this case *8Mbps*). We defined these parameters for the client-server measurement only (*cs*) and, by default, they are used for the server-client too, resulting in 2 average throughputs to be saved. More information on these applications can be found in [29].

We probe the network by sending traffic with these applications and storing in the database the relevant information

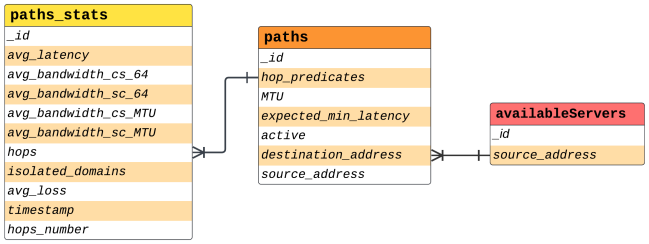

Fig. 3: Database Schema presenting, from left-to-right, collection of paths' statistics, collection of each path for each server, and servers considered for testing.

on all paths to reach the destinations considered for testing. We call these destinations `available servers`. We used MongoDB as the database for its usability and performance as well as the flexibility that comes with a non-relational database. We show its structure in Fig. 3. Notably, for every path, we store the experienced throughput in upload and download as well as with different sizes of packets. This is in line with the requirements that applications have, as the direction and the size of packets they generate might differ.

C. Path Selection

The Path Selection consists at first of a query of the database that targets paths that satisfy requirements set by the application. When more than one path is available, we compare paths with each other to determine if there is a subset of paths that are always better than others. For instance, we can compare two paths that experience the same latency and the same packet loss rate, but the first provides a higher throughput, then we know that the first is better than the

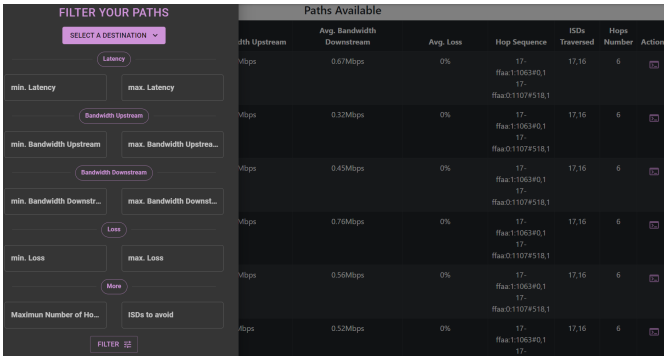

Fig. 4: MS4PAN API implementation as a web interface. On the left, the available filters for the application to choose. On the right, the results of a previous search are a set of paths that satisfy set requirements.

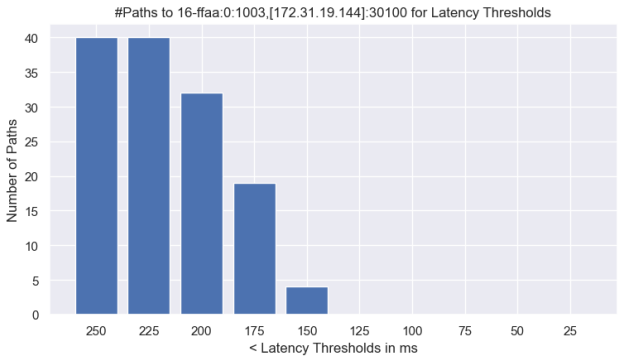

Fig. 5: Number of paths for destination 16-ffaa:0:1003, [172.31.19.144]:30100, located in North Virginia, for decreasing values of latency.

second. By extending this comparison to all possible couples of paths, we can determine which are the best paths out of the pool available. This is the basic principle behind our multi objective optimization process for choosing the best path. This process relies on the Pareto front [30] and it happens every time it is possible to compare the performance of two paths. This is done in order to provide the application with the best path available in a specific PAN. The Pareto front is invaluable in decision-making processes where trade-offs between competing objectives must be considered. We applied this concept to network optimization, to balance latency, throughput and packet loss rate.

D. MS4PAN API

Finally, the MS4PAN API is designed as a web interface and we show the filters available for the application to choose from in Fig. 4. The figure also shows the result of a previous search as a set of paths for the application to choose from.

VI. Evaluation

We discuss our evaluation of MS4PAN in terms of the complexity of managing multiple paths on a Path-Aware Network. This highlights the work that an app developer would have to carry out for every specific scenario, including different applications and different PANs, while everything happens automatically when MS4PAN is utilized.

We started our test by analyzing multiple paths available in SCIONLab to reach the destinations that a specific e-health application could target. We gathered a dataset comprising approximately three thousand samples. This provides a foundation for our following analysis, offering insights into path performance regarding latency, bandwidth and packet loss rate. For the rest of this paper, we will select `16-ffaa:0:1003, [172.31.19.144]:30100` as the destination that the e-health application needs to reach. This is done as an example and has no influence on any consideration that will follow. This destination is located in North Virginia, while our AS, that is where the application will start to send its traffic, is located in Switzerland.

The first requirements of our e-health application is the maximum latency of 200ms. Different use cases can have also stringent requirements on the standard deviation of experienced latency [11]. Because our database stores data on latency for every path and every iteration independently, we can query our database to gather information on the maximum latency ever recorded, or the mean, for example. Fig. 5 uses the mean latency, so it reports the number of paths which mean latency is lower than various thresholds. We only want to select the 34 paths that provide a mean latency lower than 200ms.

The second requirement we focus on is the minimum throughput in download of 6 Mbps. Because the application to test the throughput allows us to target a specific value, we chose a value that is close to the requirement of our use case. It is important to note that this application is specific to SCIONLab, other PANs require monitoring of data in transit to measure the experienced throughput that needs to be satisfied. Furthermore, our e-health application generates bigger packet size in download than upload, so we measured the download throughput with MTU sized packets. We can generate a similar graph to what we showed for the latency also for the throughput, and similar considerations for the maximum packet loss rate of 2%. While showing the number of paths that satisfy a requirement can give us some information on the performance and therefore the possibility of running a specific service on a network, it still doesn't show which path exactly satisfies a requirement, let alone more than one requirement at the same time. Therefore, we plot in Fig. 6 each path as a dot in a three dimensional space, with one performance requirement per axis: throughput, latency and loss.

We chose this representation, just like the unusual axis orientation, so that the bottom part of the graph corresponds to the best performing paths. In this space, every path is represented by a dot. If the dot is outside of the highlighted region, it does not satisfy all the requirements on performance. We indicated these dots in red. If it falls inside of the highlighted region, it is green and it is a suitable path for the use case considered. We can include more metrics in our

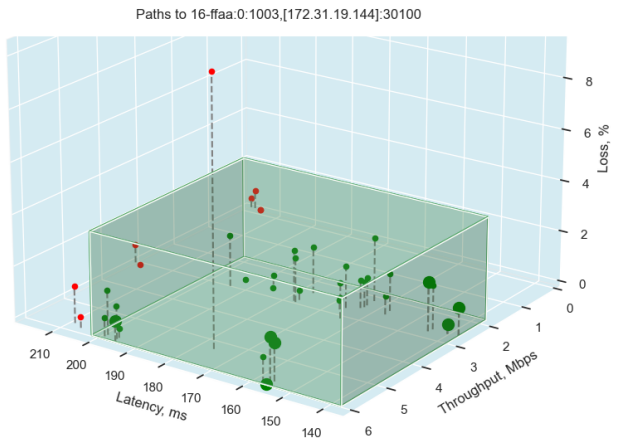

Fig. 6: Plot of each path for destination 16-ffaa:0:1003, [172.31.19.144]:30100, located in North Virginia, in the 3D space of throughput, latency and packet loss. A path is represented by a dot. A highlighted green region corresponds to satisfied requirements from the e-health use case. Thicker dots represents dots that lie in the Pareto front.

evaluation, that is, add more requirements to be satisfied by the application. We decided not to plot them for the sake of clarity.

Every path in this plot is tested "iterations" number of times. We set the number of iterations equal to 100 for our evaluation, but this is a value that changes when monitoring the data in transit. Generally, every measurement iteration provides a different value for each metric. This translates into a dispersion in space of each path, so each path should be represented by a portion of space instead of a dot. However, we noticed the plot to become less clear when showing this pattern, so we instead chose specific values for latency and throughput. Fig. 6 shows the mean value for the latency, and the minimum bandwidth for the downlink with MTU size packets. This choice is only for the sake of clarity in our plot, in fact every application requires a different study when it comes to which values to consider. While some of these requirements, like the direction of traffic flow, is present in most of the literature we came across, the packet size is less investigated in literature. But in our findings, in the SCIONLab testbed, packet size has implications on the overall performance, having consequences on throughput and latency. These consequences can affect the choice of one path over another.

While all green dots in our graph correspond to paths that satisfy all the requirements set by the application, we can determine the subset of these paths that lie on the Pareto front and that provide therefore better performance than all the others. These are represented as thicker green dots in Fig. 6.

The MS4PAN API returns to the application the list of paths that satisfy the requirements, starting with the paths lying on the Pareto front, as well as the command that the application needs to use to select that path when interact directly with the PAN.

VII. Conclusions and Future Work

We proposed MS4PAN (Measurement System for Path-Aware Networks), a tool for application developers that integrates the necessary testing of paths that also deals with the complexity of different PANs. Our software is composed of network specific components such as the NAMs, that interact with a specific PAN, and of network agnostic components, to run testing and interact with an application. MS4PAN stores measurements of several properties of the underlying network and its available paths for each destination. Our tool finds the list of best paths when provided with a destination for the application to use and its requirements on performance such as latency, throughput and packet loss rate.

We evaluated MS4PAN on the SCIONLab testbed for an e-health application. The process of measuring and selecting the best paths is transparent to the application.

Our implementation with the SCION PAN showed promising results in aiding developers and we intend to carry out experiments with different PANs as well as with different applications in the future to make the use of MS4PAN more widespread.

Acknowledgment

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

References

[1] T. Dufva and M. Dufva, "Grasping the future of the digital society," *Futures*, vol. 107, pp. 17–28, 2019.

[2] C. Hesselman, P. Grosso, R. Holz, *et al.*, "A responsible internet to increase trust in the digital world," *Journal of Network and Systems Management*, vol. 28, no. 4, pp. 882–922, 2020.

[3] R. Bazo, L. Boldrini, C. Hesselman, and P. Grosso, "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*, 2021, pp. 8–13.

[4] J. Chen, M. Díaz, L. Llopis, B. Rubio, and J. M. Troya, "A survey on quality of service support in wireless sensor and actor networks: Requirements and challenges in the context of critical infrastructure protection," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1225–1239, 2011, Advanced Topics in Cloud Computing. DOI: https://doi.org/10.1016/j.jnca.2011.01.008.

[5] H. Yu, H. Lee, and H. Jeon, "What is 5g? emerging 5g mobile services and network requirements," *Sustainability*, vol. 9, no. 10, 2017, ISSN: 2071-1050. DOI: 10.3390/su9101848.

[6] T. Norp, "5g requirements and key performance indicators," *Journal of ICT Standardization*, vol. 6, no. 1-2, pp. 15–30, 2018. DOI: 10.13052/jicts2245-800X.612.

[7] Z. Amjad, A. Sikora, B. Hilt, and J.-P. Lauffenburger, "Low latency v2x applications and network requirements: Performance evaluation," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 220–225. DOI: 10.1109/IVS.2018.8500531.

[8] R. C. Linger, N. R. Mead, and H. F. Lipson, "Requirements definition for survivable network systems," in *Proceedings of IEEE International Symposium on Requirements Engineering: RE'98*, IEEE, 1998, pp. 14–23.

[9] Y. Zuo, "A framework of survivability requirement specification for critical information systems," in *2010 43rd Hawaii International Conference on System Sciences*, 2010, pp. 1–10. DOI: 10.1109/HICSS.2010.13.

[10] C. Alcaraz and S. Zeadally, "Critical infrastructure protection: Requirements and challenges for the 21st century," *International journal of critical infrastructure protection*, vol. 8, pp. 53–66, 2015.

[11] L. Skorin-Kapov and M. Matijasevic, "Analysis of qos requirements for e-health services and mapping to evolved packet system qos classes," *International journal of telemedicine and applications*, vol. 2010, no. 1, p. 628 086, 2010.

[12] S. Scherrer, M. Legner, A. Perrig, and S. Schmid, "Enabling novel interconnection agreements with path-aware networking architectures," in *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2021, pp. 116–128. DOI: 10.1109/DSN48987.2021.00027.

[13] B. Raghavan and A. C. Snoeren, "A system for authenticated policy-compliant routing," in *Proceedings of the 2004 conference on Applications, technologies, architectures, and protocols for computer communications*, 2004, pp. 167–178.

[14] B. Raghavan, P. Verkaik, and A. C. Snoeren, "Secure and policy-compliant source routing," *IEEE/ACM Transactions On Networking*, vol. 17, no. 3, pp. 764–777, 2008.

[15] B. Bhattacharjee, K. Calvert, J. Griffioen, N. Spring, and J. P. Sterbenz, "Postmodern internetwork architecture," *NSF Nets FIND Initiative*, pp. 1–18, 2006.

[16] X. Yang, D. Clark, and A. W. Berger, "Nira: A new inter-domain routing architecture," *IEEE/ACM transactions on networking*, vol. 15, no. 4, pp. 775–788, 2007.

[17] P. B. Godfrey, I. Ganichev, S. Shenker, and I. Stoica, "Pathlet routing," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 111–122, 2009.

[18] T. Anderson, K. Birman, R. Broberg, *et al.*, *The nebula future internet architecture*. Springer, 2013.

[19] X. Zhang, H.-C. Hsiao, G. Hasker, H. Chan, A. Perrig, and D. G. Andersen, "Scion: Scalability, control, and isolation on next-generation networks," in *2011 IEEE Symposium on Security and Privacy*, 2011, pp. 212–227. DOI: 10.1109/SP.2011.45.

[20] F. J. Rodríguez-Pérez, J. L. González-Sánchez, and A. Gazo-Cervero, "Rsvp-te extensions to provide guarantee of service to mpls," in *NETWORKING 2007. Ad Hoc and Sensor Networks, Wireless Networks, Next Generation Internet: 6th International IFIP-TC6 Networking Conference, Atlanta, GA, USA, May 14-18, 2007. Proceedings 6*, Springer, 2007, pp. 808–819.

[21] F. Baker, B. Lindell, and M. Talwar, *Rfc2747: Rsvp cryptographic authentication*, 2000.

[22] B. Trammell, J.-P. Smith, and A. Perrig, "Adding path awareness to the internet architecture," *IEEE Internet Computing*, vol. 22, no. 2, pp. 96–102, 2018.

[23] A. Hanemann, J. W. Boote, E. L. Boyd, *et al.*, "Perfsonar: A service oriented architecture for multi-domain network monitoring," in *Service-Oriented Computing-ICSOC 2005: Third International Conference, Amsterdam, The Netherlands, December 12-15, 2005. Proceedings 3*, Springer, 2005, pp. 241–254.

[24] I. Monga, C. Guok, J. MacAuley, *et al.*, "Software-defined network for end-to-end networked science at the exascale," *Future Generation Computer Systems*, vol. 110, pp. 181–201, 2020, ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.04.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X19305618.

[25] I. Baldin, A. Nikolich, J. Griffioen, *et al.*, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019. DOI: 10.1109/MIC.2019.2958545.

[26] L. Boldrini, *Scion test suite*, 2024. [Online]. Available: https://github.com/leonardoboldrini/SCION-Test-Suite.

[27] A. Perrig, P. Szalachowski, R. M. Reischuk, and L. Chuat, *SCION: a secure Internet architecture*. Springer, 2017.

[28] SCIONLab, *Scionlab topology*, 2024. [Online]. Available: https://www.scionlab.org/topology.png.

[29] 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*, 2023, pp. 785–794.

[30] P. Ngatchou, A. Zarei, and A. El-Sharkawi, "Pareto multi objective optimization," in *Proceedings of the 13th International Conference on, Intelligent Systems Application to Power Systems*, 2005, pp. 84–91. DOI: 10.1109/ISAP.2005.1599245.