DISS. ETH. NO. 30877

FOUNDATIONS FOR AN OPEN MARKET FOR INTER-DOMAIN PATHS

A thesis submitted to attain the degree of

DOCTOR OF SCIENCES

(Dr. sc. ETH Zurich)

presented by

SEYEDALI TABAEIAGHDAEI

MSc ETH in Computer Science, ETH Zurich

born on 03.04.1996

accepted on the recommendation of

Prof. Dr. Adrian Perrig (doctoral thesis supervisor)
Prof. Dr. Nick Feamster
Prof. Dr. Yih-Chun Hu
Prof. Dr. João Luis Sobrinho

2025

ABSTRACT

An open market for wide-area network paths, such that customers can not only choose their last-mile provider but also their wide-area provider, has been a long-awaited vision. The fundamental limitations of the border gateway protocol and the lack of incentives to deploy an inter-domain market place have been the main impediments to such an open market. However, there are signs of change in the status quo: the SCION path-aware Internet architecture is being increasingly adopted by ISPs around the world. As a path-aware network, SCION provides endpoints with multiple inter-domain paths to choose which provides a fundamental ingredient supporting an open market for wide-area network paths.

We argue that the mere existence of more than one path and a platform to trade them is insufficient for a competitive open market for paths. Like in any open market, customers should be able to make an informed decision on which paths to use. An informed decision in the context of path selection means that customers should be provided with paths optimized for their needs, detailed path information relevant to their desired optimality criteria, and the means to detect and avoid wide-area providers with poor performance. Thus, the SCION Internet architecture needs additional mechanisms to support a truly open and competitive market for paths. In this thesis, we identify and build the required underlying foundations for a competitive market for inter-domain paths leveraging the framework of the SCION path-aware Internet architecture.

We first propose inter-domain routing with extensible criteria, a new control-plane architecture for SCION that allows for the optimization of inter-domain paths tailored to the criteria of each customer. The most important property of this system is its extensibility, allowing the addition, removal, or change of routing criteria in real time without the need for standardization, vendor support, or human involvement at ISPs.

The second problem we tackle is how to disseminate path-quality information across the network in a scalable manner. Many different quality metrics can be of interest to customers. Every metric requires a tailored system for measurement and dissemination. In this thesis, we design such systems for two different path-quality metrics: (1) the environmental impact of paths in terms of their carbon emission intensity, and (2) the propagation latency of paths. We show how these

systems can improve customers' carbon footprint and the quality of communication, respectively.

Lastly, we address the challenge of examining the performance of wide-area network providers on inter-domain paths. We design a network debugging architecture that leverages remote code execution to allow customers to reproduce network faults at each wide-area provider on inter-domain paths. Using this architecture, customers can pinpoint network problems with high accuracy. Thus, they can detect and avoid segments and providers with poor performance.

ZUSAMMENFASSUNG

Ein offener Markt für Weitverkehrsnetzpfade, auf dem die Kunden nicht nur ihren Anbieter für die letzte Meile, sondern auch ihren Weitverkehrsanbieter wählen können, ist eine lang ersehnte Vision. Die grundlegenden Beschränkungen des Border-Gateway-Protokolls und die fehlenden Anreize für die Einrichtung eines domänenübergreifenden Marktplatzes waren die Haupthindernisse für einen solchen offenen Markt. Es gibt jedoch Anzeichen für eine Änderung des Status quo: Die pfadabhängige Internet-Architektur von SCION wird von Internet-Diensteanbietern auf der ganzen Welt zunehmend übernommen. Als pfadorientiertes Netz bietet SCION den Endpunkten mehrere Inter-Domain-Pfade zur Auswahl, was eine grundlegende Voraussetzung für einen offenen Markt für Weitverkehrsnetzpfade darstellt.

Wir argumentieren, dass das bloße Vorhandensein von mehr als einem Pfad und einer Plattform für den Handel damit nicht ausreicht, um einen wettbewerbsfähigen offenen Markt für Pfade zu schaffen. Wie in jedem offenen Markt sollten die Kunden in der Lage sein, eine informierte Entscheidung darüber zu treffen, welche Trassen sie nutzen wollen. Eine informierte Entscheidung im Zusammenhang mit der Auswahl von Trassen bedeutet, dass den Kunden für ihre Bedürfnisse optimierte Trassen, detaillierte Trasseninformationen, die für ihre gewünschten Optimalitätskriterien relevant sind, und die Mittel zur Erkennung und Vermeidung von Wide-Area-Anbietern mit schlechter Leistung zur Verfügung gestellt werden sollten. Daher benötigt die SCION-Internetarchitektur zusätzliche Mechanismen, um einen wirklich offenen und wettbewerbsfähigen Markt für Pfade zu unterstützen. In dieser Arbeit werden die notwendigen Grundlagen für einen wettbewerbsfähigen Markt für Inter-Domain-Pfade identifiziert und aufgebaut, wobei der Rahmen der SCION path-aware Internet-Architektur genutzt wird.

Wir schlagen zunächst ein Inter-Domain-Routing mit erweiterbaren Kriterien vor, eine neue Control-Plane-Architektur für SCION, die die Optimierung von Inter-Domain-Pfaden ermöglicht, die auf die Kriterien der einzelnen Kunden zugeschnitten sind. Die wichtigste Eigenschaft dieses Systems ist seine Erweiterbarkeit, die das Hinzufügen, Entfernen oder Ändern von Routing-Kriterien in Echtzeit ermöglicht, ohne dass eine Standardisierung, Herstellerunterstützung oder menschliches Engagement bei ISPs erforderlich ist.

Das zweite Problem, das wir angehen, ist die skalierbare Verbreitung von Pfadqualitätsinformationen über das Netzwerk. Für die Kunden können viele verschiedene Qualitätsmetriken von Interesse sein. Jede Metrik erfordert ein maßgeschneidertes System zur Messung und Verbreitung. In dieser Arbeit entwerfen wir solche Systeme für zwei verschiedene Pfadqualitätsmetriken: (1) die Umweltauswirkungen von Pfaden in Form ihrer Kohlenstoffemissionsintensität und (2) die Ausbreitungslatenz von Pfaden. Wir zeigen, wie diese Systeme die CO_2-Bilanz der Kunden und die Qualität der Kommunikation verbessern können.

Schließlich befassen wir uns mit der Herausforderung, die Leistung von Weitverkehrsnetzbetreibern auf Inter-Domain-Pfaden zu untersuchen. Wir entwickeln eine Netzwerk-Debugging-Architektur, die die Remote-Code-Ausführung nutzt, um Kunden die Möglichkeit zu geben, Netzwerkfehler bei jedem Weitverkehrsnetzanbieter auf Inter-Domain-Pfaden zu reproduzieren. Mit dieser Architektur können Kunden Netzwerkprobleme mit hoher Genauigkeit lokalisieren. So können sie Segmente und Anbieter mit schlechter Leistung erkennen und vermeiden.

CURRICULUM VITAE

Seyedali Tabaeiaghdaei
born on 3 April 1996 in Hamedan, Iran

Education

ETH Zürich, Switzerland

Doctor of Sciences (Dr. sc. ETH) in Computer Science *2021 – 2025*
Network Security Group, supervised by Prof. Dr. Adrian Perrig
Thesis defense on January 31, 2025

Master of Science (MSc) in Computer Science *2019 – 2021*
Focus: Compueter Networks

Sharif University of Technology, Tehran, Iran *2014 – 2019*
Bachelor of Science (BSc) in Computer Engineering

Allameh Helli High School and Pre-University Center *2010 – 2014*
Hamedan, Iran
high-school diploma in Physics and Mathematics

CONTENTS

1

INTRODUCTION

In recent years, path-aware networking (PAN) [9] has not only emerged as a new paradigm in networking research but has also received much attention from the industry. The SCION next-generation Internet architecture [10] is experiencing rapidly expanding development and commercial deployment by Internet service providers (ISPs) around the world [16, 13, 14, 17, 11, 12, 15]. The fundamental difference between PANs and today's Internet lies in how the forwarding path is chosen and used in the data plane: In today's Internet, the source of traffic has no (or, at best, negligible) control over the forwarding decision, i.e., on which path to send a packet, as this decision is performed by the ISPs. On the contrary, in path-aware networks, the sender of the traffic selects the forwarding path—among the ones provided by the control plane—and encodes it in packet headers. In a path-aware network, routers are committed to following the path specified by the sender. This increase in control over inter-domain paths provides a promising opportunity to create an open competitive market for *wide-area* network paths where customers can choose the wide-area network providers based on the most suitable paths for their applications [18]. This contrasts with today's Internet, where customers can only choose their local network providers, and some can choose at most the first-hop wide-area provider through multihoming. Therefore, a PAN architecture provides the potential for expanding the competition space among wide-area providers: While in today's Internet, wide-area providers can only compete for direct connections to customer networks, in a PAN architecture, wide-area providers can *also* compete for attracting traffic from any endpoint located in networks further away than just their directly connected customer networks.

We ask the fundamental question of whether path-awareness is all that a *competitive market* for paths needs. To answer this question, we first elaborate on what we mean by a *competitive* market for paths. The goal of such a market is to provide endpoints with paths that suit their needs or preferences as closely as possible.

The participating ISPs in the market compete with each other to sell their paths to more customers, attract more traffic, and increase their revenue. The main driver of this competition is the different qualities of different paths, each suited to the needs of a specific group of users or applications. Thus, different users are attracted to different paths. This competition incentivizes ISPs to further optimize their networks for qualities that are popular among users and applications [19].

We argue that bare path-awareness is insufficient for such a competitive market, and a path-aware architecture needs to be augmented with mechanisms that allow for this competition. In this argument, we provide an analogy to a conventional online store. We take inspiration from the features existing in an online store that are still missing in a bare path-aware Internet architecture and propose systems for a path-aware architecture with equivalent functionalities to the features of an online store. In this analogy, we consider the destination of a path equivalent to the type of product in an online store (e.g., suitcase) and the ISPs on a path equivalent to the brand of a product.

Online stores usually provide customers with (1) filters to narrow the search space based on customers' preferences, (2) more information about each product, e.g., dimensions, color, weight, and functionalities, that also allows filtering, and (3) a return mechanism in case wrong products are delivered. Without these mechanisms, customers would not be able to evaluate and compare products based on their needs; instead, they would have to select one randomly without knowing what they are getting. The equivalents of these three mechanisms for a market for inter-domain paths are (1) a mechanism to filter paths based on the communication quality criteria of endpoints, (2) mechanisms to provide path information regarding various quality metrics, and (3) a mechanism allowing endpoints to detect and avoid ISPs with undesirable performance. The SCION Internet architecture has not yet been equipped with such mechanisms. This means that the control plane provides endpoints with a set of paths to a destination that are not optimized for path performance metrics, similar to the absence of performance optimization in the border gateway protocol (BGP) in today's Internet. The provided paths are not augmented with quality-metric information, and no mechanism is in place to examine the performance of ISPs. Similar to an online store, the lack of such mechanisms hinders competition in the path market. The aim of this thesis is to equip the SCION architecture with the mentioned mechanisms for an open wide-area path market.

1.1 CHALLENGES

What makes the design of these mechanisms challenging is the distributed nature of inter-domain networking: An inter-domain path crosses the network of a variable number of autonomous systems (ASes) run by independent and competing entities. Each AS only knows about its own internal topology and its connections to the neighboring ASes. ASes are not willing to share fine-grained information about their internal topology. They only share restricted information with each other through the distributed inter-domain routing protocol, using which inter-domain paths are computed to enable global connectivity.

Challenges of Filtering Paths In a federated distributed network, filtering paths according to endpoints' criteria requires *routing on multiple optimality criteria*, meaning that the distributed routing protocol should be modified to be aware of those criteria and compute optimal paths for different criteria. Note that computing all possible paths and then performing the filtering only in the end domains[1] cannot be considered an option due to the overwhelming overhead of computing all paths in the dense topology of the Internet. Another implication of the distributed routing protocol is that all ASes, which are independent entities, should be informed about the criteria and optimize paths on the same set of criteria. Traditionally, this is achieved only after standardization, vendor implementation, and deployment by every individual AS. However, the process of standardization and implementation by vendors can take years [20]. This means that the evolving needs and preferences of applications and users cannot be accommodated by the network in a timely manner, hindering the *extensibility* of routing optimality criteria.

Challenges of Providing Performance-Metric Information Every inter-domain path in a distributed network is a sequence of *intra*-domain segments that cross the networks of different ASes. Therefore, providing information about inter-domain paths regarding different performance metrics requires ASes to share the relevant information about their networks. This is challenging, as ASes are not willing to share information about their topology; thus, the shared information must not leak any information about it. The second challenge is that some performance metrics can vary frequently and significantly over time. Thus, ensuring that the provided information is up-to-date requires updating the information frequently throughout the network, which can undermine the scalability of the system. The third challenge is that different performance metrics have different characteristics:

1 We refer to the source and destination ASes of traffic as end domains or end ASes.

They need to be measured in different ways, they have different units, they vary with different frequencies, and they have different extension operations. For example, to extend the latency of a path by the latency of a link, the latency of the link is added to the latency of the path, while to extend the bandwidth, the minimum of the bandwidths is computed. Thus, no unified design can be used to compute and distribute information on all different performance metrics. Instead, different systems are required, each tailored to the characteristics of different metrics.

Challenges of Detecting and Avoiding Under-Performing Paths The end-to-end performance of an inter-domain path depends on the performance of all its constituent *intra*-domain segments. Thus, the poor performance of even one AS on a path causes the whole path to be considered as under-performing. Therefore, not detecting the under-performing segments of a path has the following implications: (1) the problem can persist even after switching to another path because the new path can still contain the same or another AS with undesirable performance, and (2) switching to an arbitrary path instead of switching to a path that only avoids the specific faulty AS penalizes the well-performing ASes on the same path in the same way as the faulty AS, as they lose traffic and revenue just because they happened to be on the same path as a faulty AS. The second challenge in detecting the faulty ASes is that each AS can treat different classes of traffic, e.g., ICMP, UDP, and TCP, in different ways, e.g., by forwarding on different internal paths or (de-)prioritizing different classes of traffic. Therefore, the debugging packets that are, by convention, ICMP packets can experience performance different from that of the actual data packets. This further complicates the detection and localization of network faults.

1.2 CONTRIBUTIONS AND COVERED PUBLICATIONS

This thesis contributes to an open and competitive market for wide-area paths in SCION by (1) designing a SCION routing architecture that allows extensible filtering of paths according to the criteria of end domains (cf. Part I: ROUTING), (2) designing two systems to disseminate path information regarding different performance metrics (cf. Part II: DISSEMINATING PATH INFORMATION), and (3) designing a network debugging system that allows fine-grained fault localization in inter-domain path-aware networks (cf. Part III: NETWORK DEBUGGING). In the following sections, we elaborate on the main contributions of each part.

Part I: ROUTING

***I**nter-domain **R**outing with **E**xtensible **C**riteria* In Chapter 3, we propose IREC, a new control-plane architecture for SCION that enables extending route optimization criteria in real time, without standardization, vendor support, or human involvement. The cornerstone of IREC is parallel route computations: To select routes using independent selection operations in parallel containers, each optimizing routes on a different set of criteria. Therefore, each AS can independently participate in any route computation by deploying the corresponding selection operation in a new container.

However, this mechanism alone still needs standardization and human involvement in all ASes to ensure that all ASes participate in a specific route computation. To address this problem, we introduce destination- and source-defined route computations, where the destination AS or the source AS of traffic starts new route computations by originating routing messages that contain a desired selection operation. The selection operation is an Extended Berkeley Packet Filter (eBPF) [21] bytecode that selects a set of routes from a set of input routes. Every AS receiving such routing messages executes this code in a user-space sandboxed virtual machine (uBPF [22]). The input is only the set of routes associated with that selection operation. These mechanisms replace standardization, as ASes automatically agree on the same selection operation by executing the eBPF bytecode. Furthermore, because the execution is automatic, there is no need for human involvement at intermediate ASes. Human involvement is still needed at the AS that starts the route computation, i.e., the source or destination of the traffic. With these mechanisms, we achieve real-time extensibility of routing criteria: End ASes can start new route computations with new selection operations at any time.

The contributions of this chapter are partially published in the following article:

[1] **Inter-domain Routing with Extensible Criteria**
Seyedali Tabaeiaghdaei, Marc Wyss, Giacomo Giuliari, Jelte van Bommel, Ahad N. Zehmakan, Adrian Perrig
arXiv:2309.03551

Part II: DISSEMINATING PATH INFORMATION

We propose two systems to disseminate path performance information about a conventional metric, i.e., latency, and a new metric we propose, i.e., the carbon intensity of the paths. While these two metrics are both lengths, endpoints can combine them in various ways to select the best path, e.g., to use a linear combination of them

or to accept bounded latency inflation in exchange for a reduction in the carbon footprint. In both systems, SCION routing messages piggyback the performance information. This allows for route optimization and enhances scalability because of SCION's hierarchical routing[2] (cf. Section 2.1). In our proposals, each AS encodes the value of the metric for the segment of the path that crosses its network. This simplifies the design, as there is no need to incorporate the extension operation of different metrics. We refer to this approach as *path decoration*, where ASes on a path can decorate it with property information. While this can also be done in BGP, it is not actionable there since the endpoints do not have any control over the forwarding path.

Carbon-Aware Global Routing in Path-Aware Networks In Chapter 4, we propose the carbon intensity of inter-domain paths as a new performance metric. Having information about this metric in a path-aware architecture allows endpoints to send traffic over paths with lower carbon intensity to reduce their carbon footprint. To that end, we propose a model for the carbon intensity of an inter-domain path. In this model, the carbon intensity of the inter-domain path is the sum of the carbon intensities of its constituent *intra*-domain segments. We design a system that is deployed in each AS and uses our proposed model to compute the 24-hour prediction of the carbon footprint of *intra*-domain paths connecting the border routers of the AS. In this computation, the system uses the map of the network—including the location and the power efficiency of devices—, intra-domain paths, and the prediction of carbon intensity of electricity at each location. The system provides this information to the control plane, which encodes the predictions into the routing messages. Providing predictions is how we achieve scalability while ensuring that the disseminated information is accurate. Disseminating real-time information would hinder scalability as the carbon intensity can vary frequently, and disseminating an average would be inaccurate.

The contributions of this chapter are also published in the following article:

[2] **Carbon-Aware Global Routing in Path-Aware Networks**
Seyedali Tabaeiaghdaei, Simon Scherrer,
Jonghoon Kwon, and Adrian Perrig
ACM International Conference on Future Energy Systems (e-Energy),
Orlando, FL, USA, 2023 (BEST-PAPER CANDIDATE)

2 SCION's control plane computes routes hierarchically in two levels: (1) in the core of the network, and (2) between the core and the edges of the network. Therefore, SCION paths can consist of three segments: (1) a segment connecting the edge to the core, (2) a segment within the core, and (3) another segment connecting the core to another edge of the network.

Towards Global Latency Transparency In Chapter 5, we propose that each AS encodes the *propagation delay* of the segment of the inter-domain path that crosses its network in the routing messages. This is how we ensure the scalability of the system, as actual latency, which includes queuing delay, can vary in the order of seconds as the queue length changes over time. We show that propagation delay is useful for (1) first-packet latency estimation for short-lived connections, where probing paths can take longer than the connection itself, (2) efficient path probing to find the lowest-latency path using the algorithm we propose, and (3) providing more accurate propagation-delay estimation for delay-based congestion control algorithms (CCA), such as BBR, and improving their fairness with respect to the CUBIC algorithm [23, 24].

The contributions of this chapter are also published in the following article:

[3] **Toward Global Latency Transparency**
Cyrill Krähenbühl, Seyedali Tabaeiaghdaei, Simon Scherrer, and Adrian Perrig
IFIP Networking, Thessaloniki, Greece, 2024

Part III: NETWORK DEBUGGING

Programmable and Verifiable Inter-domain Network Telemetry In Chapter 6, we propose *Debuglet*, a debugging architecture for inter-domain path-aware networks. In designing *Debuglet*, we take advantage of the fact that SCION paths are specified at the granularity of inter-domain links. This means that endpoints are aware not only of the sequence of ASes on a path but also of the *intra*-AS segment within each AS. To take advantage of this finer granularity for debugging purposes, we propose to install debugging nodes at the borders of each AS. These debugging nodes accept measurement requests from endpoints. By running measurements at different segments of an end-to-end path, endpoints can identify the faulty segments. To allow reproducibility of application performance, the debugging nodes accept and execute debugging applications developed by endpoints. We refer to these applications as *Debuglet*s, which are WebAssembly [25] bytecodes. The role of these applications is to generate debugging traffic that is similar to application traffic in order to reproduce the same network behavior and potential faults. Using WebAssembly improves the extensibility of the system as it allows endpoints to write the debuglets in multiple languages and compile them as WebAssembly bytecodes. We leverage a blockchain-based smart contract as the control plane of the *Debuglet* system through which the code and the results are published and executions are scheduled. Publishing the results on the blockchain makes them

verifiable, i.e., it prevents ASes from altering the results that do not provide a favorable picture of their performance.

The contributions of this chapter are also published in the following article:

[4] **Debuglet: Programmable and Verifiable Inter-domain Network Telemetry**
Seyedali Tabaeiaghdaei, Filippo Costa, Jonghoon Kwon,
Patrick Bamert, Yih-Chun Hu, Adrian Perrig
IEEE International Conference on Distributed Computing Systems (ICDCS),
Jersey City, NJ, 2024

1.3 UNCOVERED PUBLICATIONS

[5] **Quality Competition Among Internet Service Providers**
Simon Scherrer, Seyedali Tabaeiaghdaei, and Adrian Perrig
PERFORMANCE, Evanston, IL, USA, 2023

My Contribution: Development of the ns-3 [26] SCION control-plane simulator used in experiments.

[6] **G-SINC: Global Synchronization Infrastructure for Network Clocks**
Marc Frei, Jonghoon Kwon, Seyedali Tabaeiaghdaei
Marc Wyss, Christoph Lenzen, Adrian Perrig
IEEE International Symposium on Reliable Distributed Systems (SRDS),
Vienna, Austria, 2022

My Contribution: Development of the ns-3 [26] SCION *data*-plane simulator, designing experiments in collaboration with other authors, and implementing and conducting experiments.

[7] **Data-Plane Energy Efficiency of a Next-Generation Internet Architecture**
Seyedali Tabaeiaghdaei, Adrian Perrig
IEEE Symposium on Computers and Communications (ISCC),
Rhodes Island, Greece, 2022

My Contribution: Analysis of SCION data plane's energy consumption compared to that of today's Internet.

[8] **Deployment and Scalability of an Inter-Domain Multi-Path Routing Infrastructure**
Cyrill Krähenbühl, Seyedali Tabaeiaghdaei, Christelle Gloor,
Jonghoon Kwon, Adrian Perrig, David Hausheer, Dominik Roos
ACM SIGCOMM International Conference on
Emerging Networking EXperiments and Technologies (CoNEXT),
Munich, Germany (Virtual Conference), 2021

Winner of the BEST-PAPER AWARD

My Contribution: Designing a new route selection and propagation algorithm for SCION core beaconing that significantly improves the scalability and discovers more disjoint paths, and designing and conducting experiments in the SCION ns-3-based control-plane simulator that I developed.

2

BACKGROUND

This chapter provides the background knowledge required for reading the present thesis, namely, background on the SCION PAN architecture (cf. Section 2.1), and background on routing algebra definitions (cf. Section 2.2).

2.1 SCION

As this thesis designs and builds systems for the SCION Internet architecture, we provide a brief background on SCION. Details can be found in *The Complete Guide to SCION* by Chuat et al. [10].

Architecture SCION groups ASes in *Isolation Domains* (ISDs). Each isolation domain is organized hierarchically into two levels: (1) Core ASes and (2) Non-core ASes. *Core AS*es, which constitute the *ISD core*, govern the ISD and provide the *non-core* ASes in their customer cones with connectivity to other ISDs. Neighboring ASes can have one of the three following relationships: (1) Core, between two neighboring core ASes, within or across ISDs, (2) Customer-provider, within an ISD, and (3) Peering, between two peer ASes within or across ISDs.

Data Plane SCION uses packet-carried forwarding state to forward inter-domain traffic. This means that endpoints encode AS-level inter-domain paths into packet headers based on which border router of every AS forwards packets. SCION paths are specified at the granularity of inter-domain (or inter-AS) interfaces representing links between neighboring ASes, providing endpoints with fine-grained control over paths. Border routers do not need to store inter-domain forwarding tables to make forwarding decisions, enabling scalable multi-path forwarding.

Because of the hierarchical organization of ASes in ISDs, SCION paths consist of up to three path segments: (1) an up-path segment, connecting the non-core source

AS to a core AS within the same ISD, (2) a core-path segment, connecting the core ASes in the source and destination ISDs, and (3) a down-path segment, connecting the core AS in the destination ISD to the non-core destination AS within the same ISD. A path can consist of just one or two of these path segments depending on the location of source and destination ASes in the network. Also, it is possible to use up- and down-path segments partially to reach another non-core AS on the same segment.

SCION paths are reversible: the destination of a packet can reverse the path in the header of the received packet and place the reverse path into the header of the response packet.

Control Plane Path segments are computed in the *beaconing* process. Core segments are computed by *core beaconing* in which all core ASes participate. Up- and down-segments within an ISD are computed by the *intra-ISD beaconing* in which only the ASes of the same ISD participate. Only core ASes can start a path-segment computation.

To start a core-segment computation, the *beacon service* of a core AS originates a path-construction beacon (PCB) and disseminates it only to the beacon service of its neighboring *core* ASes. The originator encodes information about its AS hop in the PCB by appending an `ASEntry` field to the PCB. An `ASEntry` is a PCB field that contains all the necessary information about an AS hop on a path segment. For an AS originating a PCB, this information includes the identifier of the inter-AS interface from which this PCB originated, i.e., the interface at which the path segment arrives at the AS. When the beacon service of a core AS receives PCBs pertaining to another core AS, it selects a subset of such PCBs to be registered at the *path service* of its AS so that the path segment can be looked up and used by endpoints. The beacon service periodically selects a subset of PCBs pertaining to any other core AS to propagate further to its core neighbors. To propagate a PCB, the beacon service appends an `ASEntry` to the PCB that contains information about its AS hop. For an AS propagating a PCB, this information includes the identifier of the inter-AS interface from which the PCB is received and the identifier of the inter-AS interface on which the PCB is propagated. This process computes core-path segments from any core AS to the originating core AS.

Intra-ISD beaconing computes down- and up-path segments in a similar process but with two differences: (1) PCBs travel only from providers to customers within the same ISD, and (2) The beacon service extracts the down segment corresponding to each PCB, reverses them to compute up-path segments, selects a subset of up-path and a subset of down-path segments, registers the up segments with its

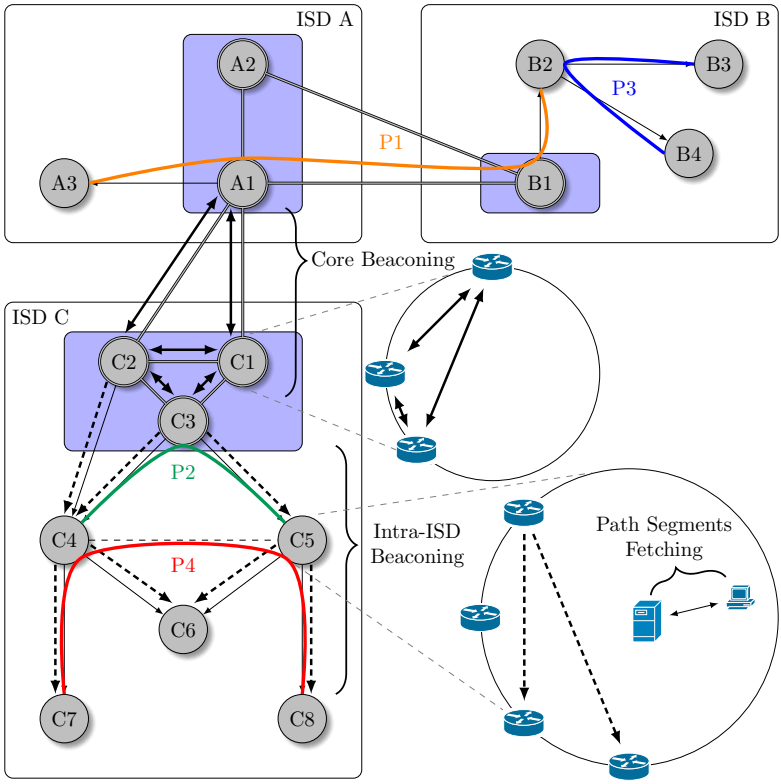


Figure 2.1: A SCION topology consisting of three ISDs and several possible end-to-end paths.

local path service, and registers the down segments with the path service of the originating core AS.

To construct an end-to-end forwarding path, an endpoint needs to request the necessary path segments from the path service at its AS in up to three steps. First, it requests the up-path segments to the core ASes of its ISD. Second, the endpoint requests core-path segments to the core ASes of the destination ISD. When the local path service receives this request from an endpoint, it queries the core path services of the core ASes of the local ISD for core-path segments from each of the core ASes to any core AS in the destination ISD. The local path service sends the responses back to the endpoint and caches them. Third, the endpoint requests its local path

service for down-path segments to the destination AS in the destination ISD. When the local path service receives such a request, it requests the core path services of the core ASes in the destination ISD for down-path segments to the destination AS. The local path service sends the responses back to the endpoint and caches them. To reach the core ASes of the destination ISD, the local path service uses the core-path segments it has received in the second step. Once the endpoint has received all the necessary path segments, it combines them into a path and encodes them into data packet headers.

Figure 2.1 illustrates a SCION topology consisting of three ISDs and several possible end-to-end paths.

2.2 REQUIRED DEFINITIONS FROM ROUTING ALGEBRA

Although this thesis does not propose new concepts and constructs in routing algebra, it uses routing algebraic terms to explain the problems and the proposed routing mechanisms. We briefly provide definitions of the used concepts that are borrowed from the work of Sobrinho et al. [27] on routing on multiple optimality criteria.

Attribute An attribute represents arbitrary performance metrics associated with a link or a path. Therefore, an attribute a can be written as a tuple of values of the performance metrics it represents. For example, $a = (w, l)$ is an attribute with width w and length l. We denote the attribute of the link that connects nodes u and v with $a[uv]$.

Extension Operation An extension operation $\oplus$ is a binary operation on attributes using which the attribute of a path can be calculated from the attributes of its constituent links. Given a path $P = u_0u_1 \dots u_{n-1}u_n$, P's attribute $a[P]$ is calculated as $a[P] = a[u_0u_1] \oplus \dots \oplus a[u_{n-1}u_n]$

Total Order A total order $\preceq$ on the set of attributes A is a binary relation on A that is antisymmetric

$$\forall a, b \in A,\ a \preceq b\ \wedge\ b \preceq a \Rightarrow a = b,$$

transitive

$$\forall a, b, c \in A,\ a \preceq b\ \wedge\ b \preceq c \Rightarrow a \preceq c,$$

and connex

$$\forall a, b \in A,\ a \preceq b\ \vee\ b \preceq a.$$

A total order expresses the relative preference between any two attributes in A. We write $a \prec b$ for $a \preceq b$ and $a \neq b$ and say that attribute a is preferred to attribute b according to the total order. A routing optimality criterion is modeled as a total order.

Partial Order A partial order $\preceq$ on the set of attributes A is a binary relation on A that is antisymmetric, transitive, and reflexive. Reflexive means that $\forall a \in A,\ a \preceq a$. Since connexity implies reflexivity, a total order is also a partial order. If $a \preceq b$ or $b \preceq a$ then a and b are comparable; otherwise, they are incomparable. We write $a \prec b$ for $a \preceq b$ and $a \neq b$ and say that attribute a is preferred to attribute b according to the partial order.

Selection Operation A selection operation $\sqcap$ is a unary operation on the set of attributes and is defined by a partial order. The set of selected attributes of A is defined as the subset of attributes of A that do not have a preferred attribute in A

$$\sqcap A = \{a \in A | \forall b \in A, \neg(b \prec a)\}.$$

Isotonicity The extension operation $\oplus$ is isotone for the partial order $\preceq$ if

$$\forall a, b, c \in A,\ b \preceq c \Rightarrow a \oplus b \preceq a \oplus c.$$

In other words, isotonicity means that the relative preference between the attributes of two paths is preserved when both are prefixed by a common third path.

Inflation The extension operation $\oplus$ is inflationary for the partial order $\preceq$ if

$$\forall a, b \in A,\ b \preceq a \oplus b.$$

In other words, the attribute of a path is not preferred to the attributes of any of its subpaths.

Isotonic Reduction An isotonic reduction of a partial order $\preceq$ on attributes A is a partial order $\preceq_R$ for which $\oplus$ is isotone such that $\forall a, b \in A,\ a \preceq_R b \Rightarrow a \preceq b$.

Largest Isotonic Reduction For every partial order $\preceq$ on attributes, there is a largest isotonic reduction $\preceq_R$ such that $\forall a, b, x \in A,\ a \preceq_R b \iff x \oplus a \preceq x \oplus b$.

Intersection of Criteria Let O be the collection of optimality criteria (total orders). The intersection of total orders $\preceq_i i \in O$ is written as $\preceq_O$ and is such that

$$\forall a, b \in A,\ a \preceq_O b \Rightarrow \forall i \in O,\ a \preceq_i b.$$

The intersection of total orders $\preceq_O$ is a partial order.

Dominant Attributes For a partial order $\preceq$, the set of dominant attributes $\mathcal{D}_{\preceq}(A)$ is the set of attributes in A such that there are no other attributes in A preferred to them. It is equivalent to $\sqcap A$ defined earlier in this section.

Routing on Multiple Optimality Criteria Sobrinho et al. propose a method for routing on multiple optimality criteria. This method consists of defining the intersection of criteria $\preceq_O$, finding the largest isotonic reduction of the intersection of criteria $\preceq_{O,R}$, and finally computing the set of dominant attributes $\mathcal{D}_{\preceq_{O,R}}(A)$.

Part I

ROUTING

PART OVERVIEW

In this part, we build the first foundational component for a competitive wide-area path market in SCION, i.e., a mechanism that allows endpoints and applications to find the most suitable inter-domain path for their communication criteria. This mechanism is the counterpart of filters in an online store, which allows customers to filter products based on various criteria. Because of the distributed nature of inter-domain networks and the infeasibility of computing all paths in the dense topology of the Internet, optimal paths should be discovered by the routing protocol through multi-criteria path optimization. Furthermore, since optimality criteria can change over time, the routing protocol should be able to extend the routing criteria accordingly.

Since finding optimal paths entails filtering paths based on some path performance metrics, complementary systems are required to provide this information about paths. We design such systems in Part II of the thesis.

3

INTER-DOMAIN ROUTING WITH EXTENSIBLE CRITERIA

3.1 INTRODUCTION

The wide variety of today's networked applications requires a diverse set of communication quality criteria that need to be satisfied by the network for an optimal quality of experience. For example, video conferencing, Voice over IP (VoIP), and online gaming applications, each have specific criteria for communication quality. Furthermore, applications are evolving at an unprecedented pace, and so are their communication quality criteria. For example, new criteria are expected to emerge with holographic communication [28], or the tactile Internet [29].

To meet these diverse criteria, the network must forward each application's traffic along paths optimized for its specific requirements, which necessitates discovering such paths within the network. Within the domain of wide-area network (WAN) providers, these optimizations have long been achieved through software-defined networking (SDN) and maintaining a complete, real-time view of the network. On the contrary, the public Internet, as an inter-domain network, has struggled to address the evolving needs of applications. This has driven the expansion of the private networks of hypergiants and the emergence of Network-as-a-Service (NaaS) providers. However, this trend has raised concerns about inheriting third-party policies and practices, vulnerability to outages, and industry consolidation [30].

The public Internet falls behind private networks primarily due to the stateful inter-domain forwarding and the limitations of the Border Gateway Protocol (BGP). Stateful forwarding restricts forwarding table sizes to the limited and expensive memory available on routers. Therefore, despite the existence of an exponentially large number of discoverable paths in the public Internet, routers usually only store a single path per destination in their forwarding tables. While this constraint limits the flexibility of route selection, the rudimentary and ossified route selection logic of BGP is also a consequence of operator choices, where paths are primarily optimized

for monetary cost and hop length. Proposed changes to BGP for computing multiple paths per destination [31, 37, 33, 38, 32, 36, 35, 34] will also eventually hit the barrier of limited forwarding state in routers. Overlay architectures such as TANGO [30] also cannot find optimal paths for diverse and evolving criteria as they can only discover a small number of paths and rely on BGP route computation.

Path-aware networking (PAN), on the other hand, takes a stateless approach to inter-domain packet forwarding by encoding the full inter-domain path in packet headers. This method eliminates the limitations imposed by stateful forwarding and BGP, with the control plane able to compute many paths between two Autonomous Systems (ASes). These paths are provided to endpoints (hosts within an AS), which can choose the most suitable one based on their criteria and the associated path metadata. The SCION [10] Internet architecture is a PAN commercially deployed by over 20 Internet service providers [16, 13, 14, 17, 12, 15]. In the current SCION network, though still relatively small yet expanding, we already observe a median of 100 usable paths between a pair of ASes, with some pairs observing up to 400 distinct paths. (These are not merely discoverable paths, but actually usable ones.) In the larger and denser topology of the Internet, the total number of potentially discoverable paths can be exponentially large. Lee et al. [30] count a median of around 5000 Gao-Rexford compliant paths between 1000 AS pairs randomly chosen from the CAIDA topology [39]. The number of paths can be even larger if we consider (1) the highest-degree ASes instead of random ones, (2) multiple connections between neighboring ASes, which are discoverable in SCION, and (3) the possibility of violating Gao-Rexford conditions in the core part of the SCION architecture.

The large number of paths will likely prompt the desire for even more optimality criteria, further increasing the importance of multi-criteria path optimization. Allowing endpoints to choose not only their local providers but also their wide-area provider [18], opens up the opportunity for network providers to convince more endpoints to steer their traffic through them to increase their revenue. Therefore, it is expected that providers introduce novel path metrics (e.g., the carbon intensity of paths [5, 2]) and advertise information regarding those metrics about segments of paths traversing their networks. Accordingly, the endpoints might update their optimality criteria to incorporate the novel metrics, accelerating the emergence of new optimality criteria.

While a promising opportunity, the exponential number of discoverable paths on the dense topology of the Internet poses considerable control-plane scalability challenges. Note that these scalability issues arise at a number of paths far exceeding those encountered in stateful forwarding. While PANs are capable of discovering

and using thousands of paths per destination, stateful forwarding cannot go beyond a few. However, even PANs cannot scale to an exponential number of paths.

Computing optimal paths for a specific criterion may seem as difficult as searching for a needle in a haystack. First, due to the exponential overhead, it is not scalable to compute all paths and provide them to endpoints for optimization. Second, unlike a WAN provider's network, where optimal paths can be centrally computed, inter-domain routing is distributed across independent, competing entities, lacking a centralized map of the network. Third, optimizing paths on a fixed and arbitrary set of criteria (e.g., shortest AS-path length), which is oblivious to the criteria of end domains, can hide the optimal paths for a variety of other criteria and result in suboptimal performance.

However, scalable computation of optimal paths for various criteria is achievable if the inter-domain routing protocol optimizes paths *during route computation*. This approach allows routing nodes at each AS to only disseminate the optimal paths to neighboring ASes, avoiding the exponential overhead of sharing all possible paths. Given the wide variety of criteria, the routing protocol needs to *route on multiple optimality criteria*. Sobrinho et al. [40] lay the algebraic foundations for routing on multiple optimality criteria by proposing to route on the largest isotonic reduction of intersections of criteria (cf. Section 2.2). While being a major breakthrough in routing algebra, practical implementation of their principles remains missing, which is mainly due to the limitations of BGP. Furthermore, by focusing on the algebraic foundations, Sobrinho et al. do not answer the following operational questions that will appear in real-world deployments:

- *How to make inter-domain routing nodes distributed in different domains aware of the routing criteria on which they need to optimize paths?*
- *How to extend the routing criteria, i.e., how can new criteria efficiently be added, or old ones removed across the network?*

In this thesis, we answer these questions by designing and implementing a new routing architecture for the deployed SCION PAN. Essentially, our architecture instantiates the principles by Sobrinho et al. [40] in an extensible manner from the operational viewpoint, minimizing the effort for introducing new routing criteria.

3.1.1 *Obstacles to Criteria Extensibility*

Our goal is to overcome the operational obstacles to the extensibility of inter-domain routing criteria. We thus highlight the major challenges.

Standardization The main obstacle to routing extensibility is that the Internet Engineering Task Force (IETF) does not adopt most changes proposed by network operators, preventing routing on the desired criteria of many domains. Even in cases where changes are adopted, it takes years for a new feature to be standardized. Only then do vendors implement these new features. Wirtgen et al. [20] show that the median time to publish BGP extension RFCs is 3.5 years, while it can take up to 10 years for some extensions. With this slow pace, standardization lags significantly behind the evolution of the optimality criteria desired by operators and applications.

Development and Deployment Cycle Once a new inter-domain routing feature is standardized, it needs to be implemented by vendors and deployed by many ASes in the network. However, it is not guaranteed that the vendors implement every published standard. Moreover, similar to standardization, implementing a new feature in commercial routers is a long and expensive process, requiring many iterations of design, review, and comprehensive testing. The expenses and duration of this process can be significantly longer for features that require a change in the hardware, especially if it is a change in chip design.

Likewise, network operators can find it cumbersome to deploy new routing features, as updating the software of all routers in the network requires tremendous time and effort. Hardware updates further intensify this problem. Furthermore, unlike vendors, not all network operators are interested in new features. This results in partial deployment of new routing criteria and, thus, suboptimal discovered paths.

xBGP [20] is a vendor-neutral architecture for BGP implementation that allows network operators to deploy new BGP extensions in their multi-vendor network. xBGP exposes an API to the BGP implementation and executes extensions on an eBPF [21] virtual machine. xBGP is a significant step towards inter-domain routing extensibility. The major drawback of xBGP is that it only allows for local decisions about BGP extensions: Operators of one domain can only decide about which extensions to deploy in their domain. However, deploying new inter-domain routing criteria requires a mechanism to orchestrate the addition of the new criteria across domains, which is not provided in xBGP. Consequently, xBGP does not provide any mechanism using which end domains could express their routing criteria to the inter-domain control plane. Therefore, xBGP is not extensible from the perspective of end-domain routing criteria.

Inconsistent Implementation and Deployment across the Internet Different vendors can implement the same standard in different ways. For example, they use different algorithms for path selection based on the BGP MED attribute [41]. Even routers from the same vendor can have different software versions both within and across domains, resulting in inconsistent routing behavior and, thus, potential suboptimality or non-determinism of selected paths. For example, consider that a router has a new version of routing software that optimizes paths for latency and environmental impact, while an adjacent router has an older software version that optimizes paths only for latency. In this scenario, the paths discovered by these two routers can be suboptimal for environmental impact. xBGP exacerbates this problem, as instead of a few possible implementations provided by a few vendors, every AS can develop and deploy its own BGP extension, which can be different from other ASes' implementations, resulting in more variants of implementations.

Control-plane Interruption Updating a router's software or hardware entails stopping and restarting its routing sessions with its neighboring routers, resulting in a re-exchange of routes with neighboring routers and potentially causing a flood of new routes in the network. Thus, frequent router updates—due to updates of optimality criteria—across the Internet can lead to an excessive number of routing messages in the network that can overwhelm some routers.

Data-plane Interruption Updating routers can interrupt the data plane mainly in two ways: (1) Forwarding loops and blackholes can emerge during the control-plane convergence due to an inconsistent view of the network among routers after each update, and (2) Heavy control-plane processing due to frequent router updates can overwhelm some routers, which may also disrupt their forwarding capabilities.

3.1.2 *Contributions*

We design IREC, i.e., **I**nter-domain **R**outing with **E**xtensible **C**riteria, a new routing architecture for the SCION PAN that enables (1) multi-criteria inter-domain routing, and (2) real-time and automated addition and removal of criteria. We build IREC on two key intuitions. First, separately optimizing paths for different sets of criteria in parallel, allowing the design of a flexible system that adds or removes criteria without interrupting other optimization processes. Second, allowing end ASes (destination or source of traffic) to communicate their routing criteria using the routing messages they originate, while other ASes—incentivized by receiving more traffic and increasing revenue—optimize routes according to the specified criteria.

To realize these intuitions, we design a system to be deployed in participating ASes. The system uses eBPF technology [21] to automatically deploy and execute an arbitrary number of route selection programs *in parallel*. Every route selection program is an eBPF program that selects paths for a separate set of criteria. Each AS can independently deploy their desired selection programs. The second intuition is realized by using routing messages as a vessel to deliver the bytecodes of programs desired by end ASes and *automatically* deploy and execute them in all other ASes for the routes pertaining to the end AS.

We show IREC's viability through implementation and emulation on a realistic topology, where it achieves reductions in median CPU and memory usage compared to the existing monolithic implementation of the SCION control plane. Furthermore, using simulations on a much larger Internet topology, we examine the impact of routing with separate but smaller sets of criteria and compare it with that of routing with one larger set of criteria. We observe similar message complexity and convergence time, suggesting that IREC can achieve extensibility with negligible cost.

In summary, we contribute to a performant and extensible Internet by:

- designing IREC, a control-plane architecture for the SCION PAN, enabling extensible and multi-criteria path optimization (cf. Sections 3.3 and 3.6);
- implementing IREC in the open-source codebase [42] of the SCION PAN architecture [10] using user-space eBPF (uBPF) [22] (Section 3.9), and emulating it using the Kathará emulator [43]; and
- evaluating IREC on a realistic Internet topology by developing and using a simulator based on ns-3 [26] (cf. Section 3.10).

The contributions of this part are based on an article available on arXiv [1].

3.2 EXAMPLES

his section elaborates on our goals of routing on multiple optimality criteria and the extensibility of routing criteria by means of two examples that we re-use throughout the chapter.

3.2.1 *Multi-Criteria Path Optimization*

Figure 3.1 illustrates an inter-domain topology in which each node represents an AS and each edge represents an inter-domain link with *attribute* (w, l) where

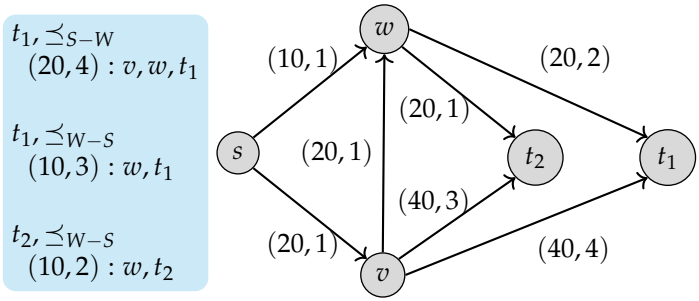


Figure 3.1: Example of multi-criteria path optimization, where routes to t_1 should be optimized using the shortest-widest and widest-shortest orders, while routes to t_2 should be optimized using the widest-shortest order. Link labels demonstrate their associated (*width*, *length*) attributes.

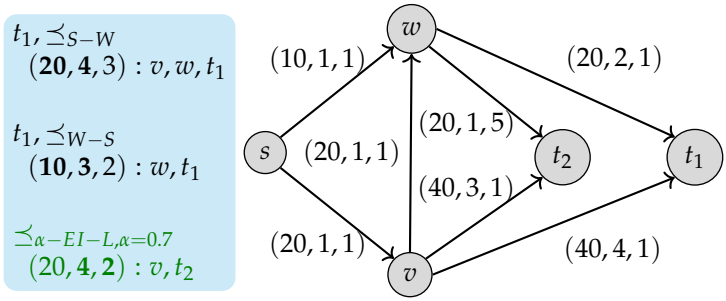


Figure 3.2: Example of the need for extensibility of routing criteria. In this case, the optimality criterion for paths to t_2 is changed, compared to Figure 3.1, to take into account both the length and the environmental impact (an additive metric) of routes. Link labels demonstrate their associated (*width*, *length*, *environmental impact*) attributes.

w is its width and l is its length l. Using this topology, we provide an example of the need for path optimization on different criteria for different applications and end domains. Suppose AS t_1 hosts two services: a video-call service to which customers should reach via paths primarily with high bandwidth and secondarily with low latency, and a VoIP service to which customers should reach via paths primarily with low latency and secondarily with high bandwidth. Concurrently, suppose that AS t_2 is hosting a VoIP application that also needs to be reached through primarily low-latency and secondarily high-bandwidth paths. The video

call's optimality criterion is the shortest-widest order, $\preceq_{S-W}$, in which width-length (w, l) is preferred to width-length (w', l') if its width is greater, $w > w'$, or the widths are equal, and its length is smaller, $w = w'$ and $l < l'$. The VoIP's optimality criterion is the widest-shortest order, $\preceq_{W-S}$ in which width-length (w, l) is preferred to width-length (w', l') if its length is smaller, $l < l'$, or the lengths are equal and its width is greater, $l = l'$ and $w > w'$. The blue box at the left of node s in Figure 3.1 shows the desired final state of routing protocol at s for each destination t_1 and t_2. To reach this final state, the control plane needs to be aware of different routing criteria for t_1 and t_2 and optimize paths accordingly.

3.2.2 *Criteria Extensibility*

Suppose that link attributes in the topology can be augmented with a new quality metric, e.g., the environmental impact of paths in terms of carbon intensity of data transmission (ISPs trying to attract more traffic from eco-conscientious users). Figure 3.2 illustrates the updated topology, where the link attribute (w, l, EI) represents width w, length l, and environmental impact EI, which is an additive metric. With such a new metric, new optimality criteria are expected to emerge. For example, suppose the AS s wants to incorporate greenness into its path selection criterion to AS t_2 by defining a total order $\preceq_{\alpha-EI-L}$ on the linear combination of non-dimensionalized length and non-dimensionalized environmental impact, i.e., $\alpha EI_{nd} + (1-\alpha) l_{nd}$, where (w, l, EI) is preferred to (w', l', EI') if $\alpha EI_{nd} + (1-\alpha) l_{nd} < \alpha EI'_{nd} + (1-\alpha) l'_{nd}$. An α value of 0.7 results in the change of the desired path to t_2 for s compared to the previous widest-shortest criteria. For this path to be advertised to s, v needs to be aware of this change of routing criteria (and possibly the value or the range of α at s) and update the routing logic for destination t_2 and source s accordingly. Importantly, the value of α can differ for different source and detonation AS pairs, resulting in distinct criteria, all of which should be accommodated by the control plane.

3.3 IREC'S ROUTING MECHANISMS

In this section, we define our design goals and introduce IREC's routing mechanisms to achieve these goals: (1) Parallel routing, (2) Destination-defined route computations (RCs), and (3) Reverse routing: Source-defined RCs, which together achieve all four goals of extensibility of routing criteria, as we describe next.

3.3.1 *Design Goals*

To realize the extensibility of inter-domain routing criteria in SCION, our design must overcome the challenges mentioned in Section 3.1.1. Therefore, we specify the following concrete goals, the achievement of which is essential for a design to overcome those challenges:

G1 Enabling the addition and removal of routing criteria without interrupting the routing on other criteria.

G2 Allowing each domain to develop and deploy routing criteria of their choice without waiting for standardization or vendors.

G3 Providing a mechanism for automated, real-time, and consistent deployment of new routing criteria in multiple domains without standardization, vendor support, or involvement of their network operators.

G4 Providing end domains, i.e., any domain that can be the source or destination of traffic, with a real-time mechanism to express their desired routing criteria with immediate effect, i.e., a new RC optimizing routes according to the expressed criteria is instantly started.

3.3.1.1 *Scope of the Goals: Core Beaconing in SCION*

In the context of SCION, which has a hierarchical routing architecture, i.e., core and intra-ISD beaconing, we only aim to achieve the above goals in *core beaconing*, and thus, propose our design for core beaconing. This is due to the fact that in the core of the network, the number of core-path segments is exponentially large because of the dense interconnection of ASes and the possibility of valley-full routing. In such circumstances, discovering all possible core-path segments is practically impossible due to the overwhelming communication and computation costs. Therefore, to discover core-path segments that are optimal according to specific criteria, it is necessary to take the optimality criteria into account for core-segment computation during core beaconing. In contrast, sparse intra-ISD topologies, together with the unidirectional intra-ISD beaconing from core to leaf ASes that builds a directed acyclic graph (DAG), make it possible to discover *all* possible up- and down-path segments with negligible cost. This is empirically shown by Krähenbühl et al. [8]. Hence, taking into account optimality criteria for intra-ISD beaconing does not provide any benefit and is therefore unnecessary.

3.3.2 *Overview*

IREC has three routing mechanisms. First is *parallel routing*, which means running multiple RCs in parallel, each optimizing routes according to a different set of criteria. To participate in each RC, each AS needs to deploy the corresponding route selection program (SP). This mechanism allows for interruption-free addition or removal of criteria. IREC provides a flexible framework for deploying these SPs in parallel using eBPF [21] technology (cf. Section 3.6). The other two mechanisms build on the foundation provided by the first mechanism to provide an interactive interface for end domains, i.e., core ASes at the ends of core-path segments, to express their desired routing criteria. Second is *destination-defined RCs* whereby any core AS can influence the optimization of routes (i.e., core-path segment) *to itself*, meaning that the destination of traffic is in the customer cone of the originating core AS, and thus, it is called a *destination-defined RC*. To use this mechanism, an AS uses the PCBs it originates as a vessel to also deliver the eBPF bytecode of its desired SP. The SP optimizes the routes according to the criteria of the originating AS. Motivated by receiving more traffic and increasing revenue (cf. Section 3.11), core ASes other than the originating AS deploy and execute these SPs in real time only for the routes pertaining to the originating AS. This way, ASes reach an agreement on the SP without standardization, vendor support, or human involvement. Third is *source-defined SPs* whereby any core AS can request computation of routes (i.e., core-path segments) *in reverse from itself to all other core ASes*, meaning that the source of traffic is in the customer cone of the originating core AS, and thus it is called *source-defined SPs*. Using this mechanism, the originating AS influences the optimization of routes similar to the previous mechanism.

Although we design IREC for the core beaconing in SCION, its fundamental principles are more general than SCION core beaconing. Therefore, we use general routing terms and concepts in explaining those principles, making it easier for a broad audience to understand the contributions of IREC. For audience familiar with SCION, we clarify the meaning of the following terms used in the text in the context of SCION core beaconing: (1) by AS, we mean SCION core AS, (2) by end domain, we mean a core AS at the end of a core-path segment, (3) by destination AS, we mean a core AS whose customer cone (including itself) includes the destination AS of the traffic, and (4) by source AS, we mean a core AS whose customer cone (including itself) includes the source AS of the traffic. Consequently, a destination-defined RC is defined by a core AS whose customer cone includes the destination AS of the traffic and a source-defined RC is defined by a core AS whose customer cone includes the source AS of the traffic.

3.3.3 *Parallel Routing*

Parallel routing means running parallel RCs, each with a different selection operation, such that they do not interfere with each other. This contrasts with today's inter-domain routing, where routes are computed in one single RC with a unified selection operation.

The selection operation of an RC selects paths using *either* of the following methods:

- It sorts path attributes based on a total order that satisfies inflation and isotonicity properties and selects the best path attributes or
- It computes the set of dominant path attributes according to the largest isotonic reduction of the intersection of one or multiple total orders that satisfy inflation [27].

We refer to the executable implementation of a selection operation as a *SP*, which is executed by ASes participating in the corresponding RC.

Each SP takes into account a subset of performance metrics present in the path attributes. As an example, assume that the path attributes are in (w, l, EI), where w, l, and EI represent the width, length, and environmental impact of a path, respectively. There can exist two parallel RCs: one with a selection operation that selects paths based on their width and length (w, l), and another with a selection operation that only considers the length and environmental impact (l, EI).

IREC achieves multi-criteria path optimization with a different approach than the one proposed by Sobrinho et al. [27]: Instead of routing on a *single* intersection of *all* criteria in *one* RC, IREC routes on different *subsets of criteria* in parallel RCs. This allows for the extensibility of routing criteria from an operational perspective. Nevertheless, IREC does not introduce new algebraic concepts but re-uses the ones introduced by Sobrinho et al. [27] in a new way.

3.3.3.1 *Examples Revisited*

Figure 3.3 re-uses the example in Figure 3.2 to show how parallel RCs work and optimize paths for multiple criteria in practice. In this example, nodes v, w, and s all deploy the same two SPs. The first SP computes the set of dominant attributes on the product order[1] of width and length [44] ($\mathcal{D}_{\preceq_{W\times L}}$). The second SP sorts attributes based on the linear combination of non-dimensionalized length and

1 In the product order (w, l) is preferred to (w', l') if they are different and $w \geq w'$ and $l \leq l'$

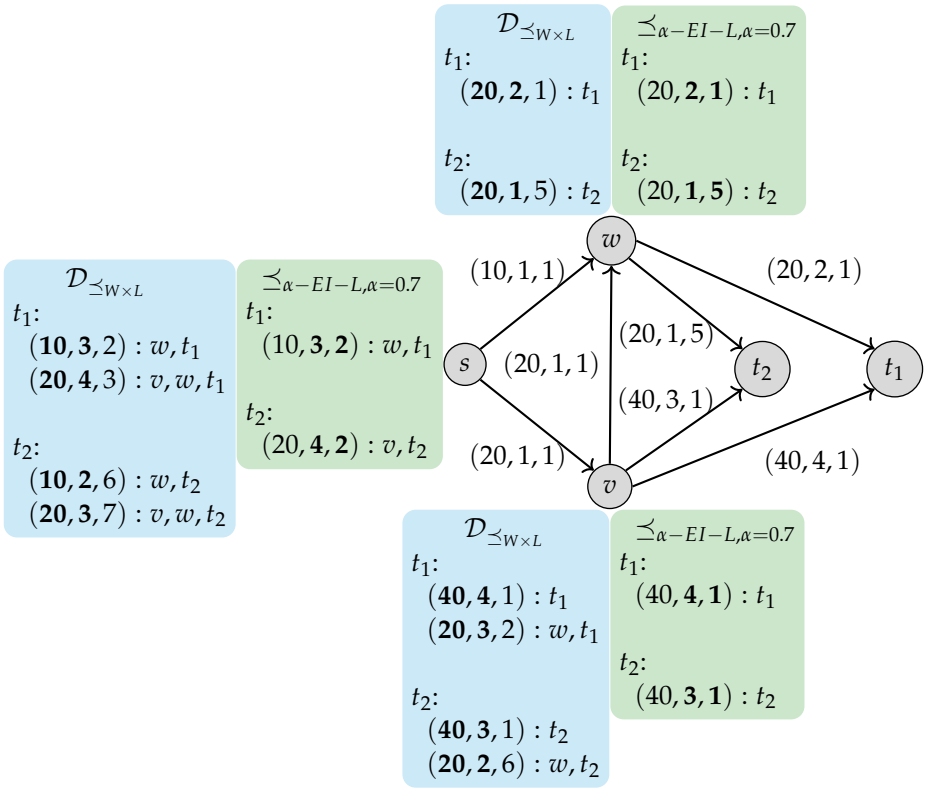


Figure 3.3: An example of parallel route computations. There are two computations: $\mathcal{D}_{\preceq_{W\times L}}$ that computes dominant attributes on the width length product order, and $\preceq_{\alpha-EI-L,\alpha=0.7}$, which selects attributes with the lowest linear combination of length and environmental impact with a weight of 0.7 for environmental impact. Attributes are in (w, l, EI) form representing width, length and environmental impact, respectively.

non-dimensionalized environmental impact, i.e., $\alpha \times EI_{nd} + (1-\alpha) \times l_{nd}$, where α is 0.7. As the example shows, the SPs that run in parallel may select the same attributes, which means redundant computation. This computational overhead is the inevitable price that needs to be paid for extensibility. However, bandwidth-use overhead can be avoided by filtering duplicate paths after selection by all parallel SPs (cf. Section 3.6).

3.3.3.2 *Extensibility*

Parallel routing achieves G1 and G2 (cf. Section 3.3.1) in enhancing the extensibility of routing criteria through:

- Interruption-free Deployment: When RCs run in parallel, introducing a new RC or removing one does not affect other RCs. This solves the problem of *Control-plane Interruption* when extending the criteria (cf. Section 3.1.1).
- Easier and Faster Development: With separate RCs, adding new criteria requires implementing a new SP. We argue that this is easier and faster than modifying one single SP every time new criteria are introduced. First, designing an order that satisfies a small subset of criteria is easier than designing an order that satisfies all criteria, especially when the number and variety of criteria grow over time. Second, the SP is simpler when it needs to satisfy a small subset of criteria, resulting in faster development and testing cycles. Third, extending one single SP means an ever-complicating and expanding program. This makes the maintenance and evolution of the SP more expensive over time, hindering extensibility. In contrast, each of the parallel SPs in IREC does not grow and change as new criteria are added by developing new SPs. Therefore, parallel routing makes the *Development and Deployment Cycle* (cf. Section 3.1.1) easier and faster.
- Allowing for non-Standardized RCs: Running RCs in parallel allows ASes to participate in non-standardized RCs while respecting the standardized ones as well, providing a solution to the *Standardization* problem in Section 3.1.1. In contrast, with a single standardized RC, all ASes need to follow the standard, and adding or removing criteria means deviating from the standard.

3.3.3.3 *Participating in RCs*

By design, IREC does not mandate the deployment of any specific SP by ASes. This gives every AS the freedom to deploy SPs independently, hence enhancing extensibility.

However, computing optimal paths on specific criteria requires the deployment of the corresponding SP by all ASes on the optimal path(s). As optimal paths are not known in advance in a distributed and federated network, all ASes need to participate in an RC to guarantee the optimality of paths. Therefore, independent choice of SPs, which can be inconsistent across ASes, can lead to suboptimal paths.

This is an instance of the *Inconsistent Implementation and Deployment across the Internet* obstacle to extensibility (cf. Section 3.1.1), which also exists in xBGP [20].

To increase the chance of the participation of ASes in certain RCs, we explore two complementary ways of specifying the RCs in which they need to participate without preventing them from participating in the RCs they desire. The first incorporates two built-in mechanisms of IREC we propose next in Section 3.3.4 and Section 3.3.5, allowing the deployment of new SPs in *all* participant ASes in real time. These mechanisms replace standardization and eliminate human involvement. The other more conventional way is a standardization and deployment model for SPs we discuss in Section 3.7.

3.3.4 *Destination-defined RCs*

With this mechanism, any traffic destination can express the optimization criteria for paths pertaining to its AS and ensure that other ASes apply these criteria without waiting for standardization, vendor support, or deployment efforts in other ASes.

This mechanism leverages the potential of adding new RCs provided by the parallel routing mechanism to further enhance the extensibility of IREC.

Every destination-defined RC is a new RC started by a destination AS that computes the routes pertaining to that destination using an SP specified by that same destination. This customized SP is communicated via an extension to PCBs (i.e., routing messages in SCION) that is populated by the destination. A destination AS can start multiple independent RCs with different SPs. When receiving such PCBs, every other AS participating in this mechanism executes these destination-defined SPs to select and propagate PCBs. With parallel routing, these SPs are executed in parallel to other SPs, thus not interrupting them.

A destination-defined RC consists of the following steps:

1. The destination AS that wants to instantiate such an RC develops the corresponding SP.

2. The destination AS compiles the SP into an executable format that is standardized for this purpose. eBPF [21] and WebAssembly [25] bytecodes are both suitable examples of such formats.

3. The destination AS makes this executable publicly accessible as a web resource served within its AS. The server serving these resources uses the SCION control-plane public-key infrastructure (CP-PKI) to establish the TLS

connection and authenticate itself. Therefore, circular dependency on the web PKI is avoided.

4. The destination AS originates new PCB(s) with (1) the unique resource locator (URL) of the SP encoded in an extension and (2) the performance metric(s) considered in the SP in other extensions. To avoid circular dependency on the domain name system, the URL must contain the SCION address of the server on which the executable is located.

5. Every AS receiving PCBs with SPs groups them so that all PCBs with the same executable URL and pertaining to the same destination—i.e., belonging to the same RC—fall into the same group.

6. The receiving AS requests the SP associated with each group from the corresponding destination AS. To reach each destination AS, the AS can extract the path specified by any received PCBs pertaining to that destination.

7. The receiving AS performs security checks and verification on the received executable to avoid malicious behavior. Some of such checks are embedded in eBPF and WebAssembly.

8. The receiving AS executes all SPs it has received in a round-robin way. The program argument is the set of PCBs specifying that SP in their extension. The AS can remove PCBs violating its local policy from the candidate set *before* executing the SP.

9. The SP returns the set of selected PCBs.

10. The AS extends the selected PCBs and their path attributes and advertises them to its neighbors.

3.3.4.1 *Examples Revisited*

Figure 3.4 provides an example of how destination-defined RCs can satisfy the criteria presented in Figure 3.1 and Figure 3.2 (cf. Section 3.2). In this example, we assume that there is no standardized generic RC and that all RCs are defined by destinations.

Some applications in destination t_1 prefer the widest-shortest paths, and some prefer the shortest-widest paths. However, the shortest-widest order is not isotone [27]. Therefore, destination t_1 starts an RC that computes the set of dominant attributes to t_1 on the product order of width and length (t_1, $\mathcal{D}_{\preceq_{W\times L}}$), which is the

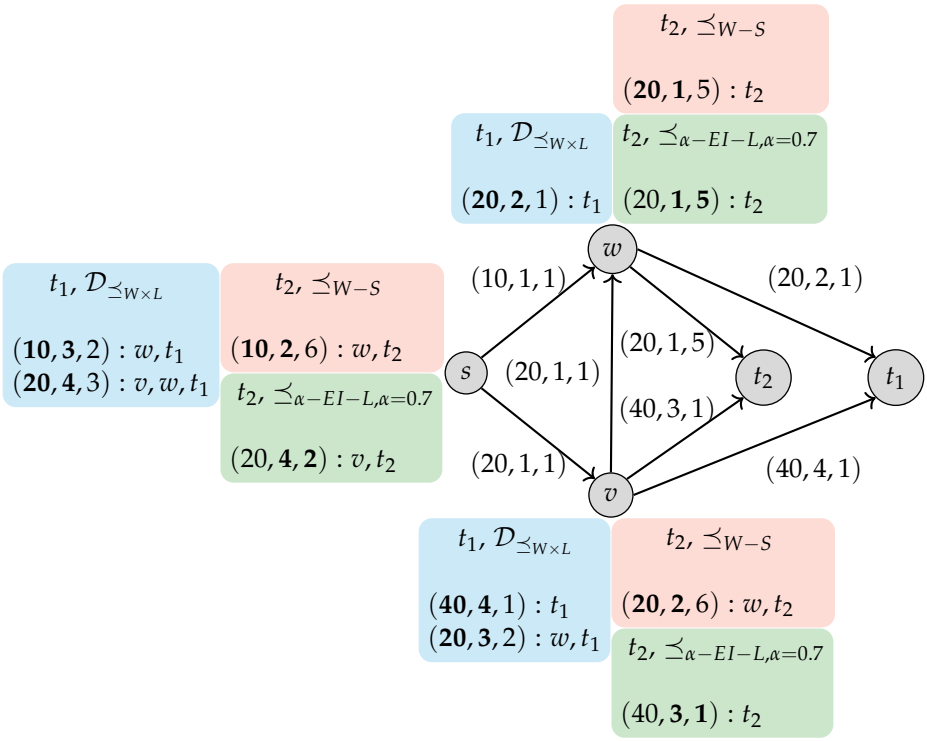


Figure 3.4: Example of destination-defined route computations. Destination t_1 starts a computation that computes dominant attributes pertaining to t_1 on the width length product order (t_1, $\mathcal{D}_{\preceq_{W\times L}}$). Destination t_2 starts two independent route computations: (1) One that selects the best attribute to t_2 according to the widest-shortest order (t_2, $\preceq_{W-S}$), which is a an isotone total order, and (2) One that selects the attribute to t_2 with the lowest linear combination of length and environmental impact with a weight of 0.7 for environmental impact (t_2, $\preceq_{\alpha-EI-L,\alpha=0.7}$).

largest isotonic reduction of the shortest-widest order. This order also allows for routing on the widest-shortest order.

Destination t_2, on the other hand, initially prefers the widest shortest paths. Since the widest-shortest order is an isotone total order, destination t_2 starts an RC that sorts path attributes to t_2 using the widest shortest order (t_2, $\preceq_{W-S}$) and selects the best one. Later, when the environmental impact of the paths is introduced as a valid performance metric, the destination t_2 starts another RC that selects the attribute

to t_2 with the lowest linear combination of non-dimensionalized length and non-dimensionalized environmental impact with a weight of 0.7 for environmental impact (t_2, $\preceq_{\alpha-EI-L,\alpha=0.7}$). If destination t_2 wants to completely discontinue the RC with the widest-shortest order, it just needs to stop originating such PCBs from its AS. Since SCION PCBs have a limited lifespan, such PCBs are automatically removed from the control plane.

3.3.4.2 *Extensibility*

Destination-defined RCs achieve G3 and G4 (only for the destination) goals of extensibility of routing criteria (cf. Section 3.3.1) through:

- Standardization Substitution: A destination-defined RC has the same functionality as a standard: It ensures all ASes conform to the same route selection operation. The difference is that standardization takes years, while agreement among ASes using destination-defined RCs is reached instantly. This solves the *Standardization* problem in achieving criteria extensibility (cf. Section 3.1.1).
- Automated and Instant Deployment: Unlike the deployment of generic SPs, which requires human involvement in all participating ASes, destination-defined SPs are executed automatically and instantly without human involvement in participating ASes other than the destination AS. This removes the *Deployment Cycle* obstacle to extensibility of routing criteria (cf. Section 3.1.1).
- Consistent Implementation and Deployment: A destination-defined SP not only acts as a standard but also guarantees that all participating ASes execute the exact same program. A standard cannot provide such a guarantee. Thus, destination-defined RCs are superior to standardized generic RCs regarding implementation consistency. Furthermore, the adoption rate of a destination-defined RC is higher than that of a standardized generic one because of the automated and instant deployment of destination-defined ones, as opposed to the generic ones. Therefore, destination-defined RCs overcome the *Inconsistent Implementation and Deployment across the Internet* obstacle to the extensibility of routing criteria (cf. Section 3.1.1).

3.3.5 *Reverse Routing: Source-defined RCs*

This mechanism allows the source AS of traffic to initiate an RC that computes the paths in reverse *from the source to any destination* using the SP defined by the

source. The novelty of this mechanism is the computation of paths in reverse: PCBs (routing messages) flow in the *same direction* as data packets, i.e., from source to destination, to compute paths *from a specific source to any destination*. This is in contrast to conventional routing protocols, where routing messages flow in the *opposite direction* of data packets to compute paths *from any source to a specific destination*.

A source-defined RC works almost the same as a destination-defined one, but with the following differences:

- The source AS starts the RC by originating PCBs with a reverse flag in a PCB extension.
- Every AS receiving such a PCB extends its path attribute in the direction of PCB traversal. This is in contrast to conventional routing, where the path attributes of the PCBs are extended in the opposite direction of PCB flow.
- Every AS receiving such a PCB, in addition to storing and advertising it if elected, sends one copy of the PCB back to the source AS. This is because any AS is a destination to which the routes are being computed.
- The encoded path in the PCB is the optimized path from the source to a destination and should be used as is. In conventional routing, the optimized path is the reverse of what is encoded in the PCB and should be reversed.

Figure 3.5 provides an example of source-defined RCs.

Note that reverse routing is only possible in a PAN architecture that benefits from stateless forwarding. This is because, in *stateful* forwarding, the forwarding table contains entries per specific destination; thus, routes can only be computed *to a specific destination*, and not from *a specific source to any destination*.

This mechanism completes the puzzle of routing criteria goals (cf. Section 3.3.1) IREC can achieve by allowing the source of traffic to express its routing criteria, i.e., *G4*.

3.4 INCORPORATING INTRA-AS TOPOLOGIES FOR OPTIMALITY

Every AS on an inter-domain path is a network of routers distributed across different locations. Consequently, an inter-domain path is a sequence of *intra-AS* paths connected by inter-domain links. Therefore, the attributes of intra-AS paths contribute to the attributes of inter-domain paths. However, the SCION control plane abstracts away the internal topology of ASes by modeling them as abstract

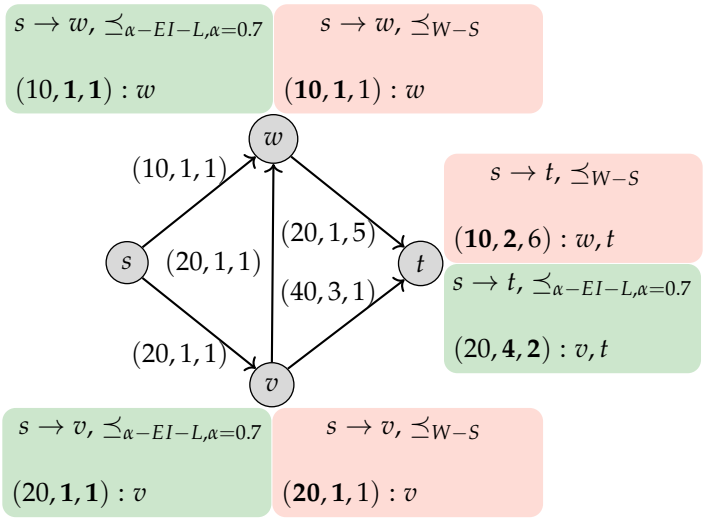


Figure 3.5: Example of source-defined route computations. Source s starts two independent route computations: (1) One that selects the best attribute from s to any destination according to the widest-shortest order ($s \to *, \preceq_{W-S}$), which is a an isotone total order, and (2) One that selects the attribute from s to any destination with the lowest linear combination of length and environmental impact with a weight of 0.7 for environmental impact ($s \to *, \preceq_{\alpha-EI-L,\alpha=0.7}$).

nodes. This means that the attributes of *intra-AS* paths within ASes are not taken into account in computing the attributes of inter-domain paths, potentially resulting in the selection of suboptimal inter-domain paths. In the following two sections, we elaborate on how this modeling can cause suboptimality and how we address these problems.

3.4.1 *Last-Mile Path Matters*

3.4.1.1 *Problem*

Modeling ASes as abstract nodes ignores the last-mile intra-AS path from the last border router to the traffic destination within the end AS. Such modeling falsely assumes that any inter-domain path that is optimal for reaching a border router of an AS is also optimal for reaching any endpoint within that AS. SCION beaconing, until the time of writing this thesis, works based on this assumption: It computes

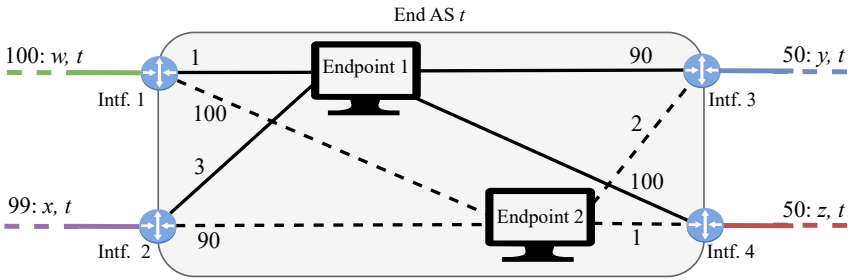


Figure 3.6: An example of per-AS route computation resulting in suboptimal end-to-end paths. Each line represents a path. Lines within the AS represent intra-AS paths to endpoints, and lines out of the AS represent inter-domain paths to AS t received by an AS v (not shown). While the shortest path to endpoint 1 enters AS t at interface 1, a per-AS route computation according to the shortest order $\preceq_S$ results in selecting the two paths reaching interfaces 3 and 4.

routes on a per-AS basis, i.e., it selects optimal paths to reach *any entry point* of an AS, which can result in suboptimal paths to endpoints.

Figure 3.6 illustrates an example of per-AS route computation resulting in suboptimal paths to endpoints. A per-AS route computation according to the shortest order $\preceq_S$ selects the shortest inter-domain paths to AS t, i.e., the ones with length 50 to interfaces 3 and 4. Using these paths to reach endpoint 1 results in data packets being forwarded on intra-AS paths with lengths 90 or 100 from interfaces 3 or 4 to endpoint 1. This results in a total length of 140 or 150 to endpoint 1. On the other hand, selecting and using the inter-domain path to interface 1 results in a total length of 101 to endpoint 1, i.e., the shortest possible length to that endpoint. However, the per-AS route computation never selects the path to interface 1, resulting in suboptimal end-to-end performance.

For core-path segments combined with up- or down-path segments, the endpoint in the above example is the border router where the up or down segment ends or starts, respectively.

3.4.1.2 *Solution*

To better account for the last-mile intra-AS paths to endpoints, we propose computing routes pertaining to groups of interfaces of each AS. This means selecting and advertising optimal attributes to reach every group of AS interfaces. This approach

is equivalent to considering every group of interfaces as a different AS to which routes are computed without creating new ASes.

The intuition behind this approach is that the interfaces of an AS can be grouped together based on the attributes of the paths connecting them to different services hosted in the AS. Interfaces in the same group offer similar performance, with regard to a subset of metrics, to reach all services hosted in the AS. Therefore, this approach can provide benefits similar to announcing different prefixes from different subsets of routers with BGP in today's Internet.

RC pertaining to interface groups requires the following actions:

ASes Starting RCs (End ASes)

- *Defining interface groups and assigning an identifier to them.* Different ways of grouping can co-exist according to different performance metrics. For example, location-based groups and bandwidth-based groups can co-exist for latency and bandwidth optimizations, respectively. Therefore, the interface groups are not mutually exclusive. An interface in the central Europe location-based group can be part of the high-bandwidth group together with other high-bandwidth interfaces around the world.

- *Starting at least one RC for every interface group.* This is done by originating PCBs, each containing the interface group identifier, from all interfaces that are part of that interface group. The inclusion of an interface-group identifier is independent of whether the RC is a generic one or a source- or destination-defined one, depending on whether the AS includes an SP in the PCBs as well. Multiple RCs with different SPs can be started for the same interface group.

Note that an AS can modify its interface groups. In that case, it needs to start new RCs accordingly.

ASes Receiving PCBs with Interface-Group Identifiers

- *Performing the selection operation separately on PCBs pertaining to every interface group of every AS.* In doing so, they group PCBs based on their interface group identifier (and ISD-AS number, to avoid collision of interface group identifiers in different ASes).

- *Advertising selected PCBs pertaining to every interface group to neighboring ASes.*

Figure 3.7 shows how interface groups can be defined in the same topology as Figure 3.6 to enhance the optimality of paths to endpoint 1 with regard to shortest

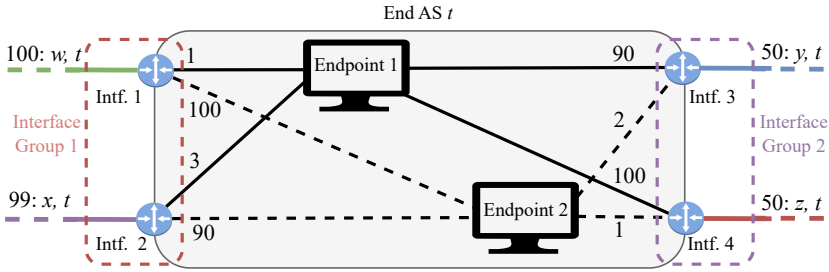


Figure 3.7: An example of interface groups enhancing the optimality of paths to endpoint 1 with regard to shortest order $\preceq_S$. The shortest path to interface group 1 is 99: x, t to interface 2, resulting in a length of 102 to endpoint 1. Without interface groups, the shortest paths to AS t are 50: y, t and 50: z, t, resulting in a length of 140 to endpoint 1.

order $\preceq_S$. AS t defines two interface groups: group 1, including interfaces 1 and 2, and group 2, including interfaces 3 and 4. AS v (not shown in the figure) receives paths starting from these four interfaces and groups these paths according to their interface groups: Two paths pertaining to group 1, i.e., 99: x, t and 100: w, t, and two paths pertaining to group 2, i.e., 50: y, t and 50: z, t. The selection operation is performed for every interface group separately: Path 99: x, t is elected for reaching interface group 1 of AS t, and both 50: y, t and 50: z, t are selected to reach interface group 2 of AS t. Using 99: x, t to reach endpoint 1 results in a near-optimal length to the endpoint, i.e., 102, while a per-AS path computation only selects 50: y, t and 50: z, t, resulting in at least a length of 140 to endpoint 1.

Note that the selected path to interface group 1, i.e., 99: x, t, still does not result in the optimal length to endpoint 1. The optimal length to endpoint 1, i.e., 101, only results from using 100: w, t, which is not selected as it is longer than 99: x, t to the same interface group 1. To address this issue, AS t can define more granular interface groups by creating an interface group that only contains interface 1 and another interface group that only contains interface 2. In that case, 100: w, t, is also selected, resulting in optimal length to endpoint 1.

The remaining research question is how to realize that endpoint 1 should be reached over interface group 1. Unless this question is answered, the proposed mechanism cannot provide a significant benefit. This problem is also due to per-AS instead of per-prefix routing in SCION. One way to solve this problem is to use DNS to provide information on the optimal interface groups to reach a service.

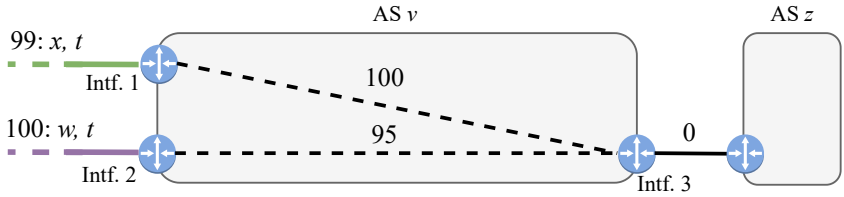


Figure 3.8: An example of how abstracting internal AS topology results in advertising suboptimal attributes. Attribute b = 99: x, t is preferred to attribute c = 100: w, t according to the shortest order $\preceq_S$. By abstracting the internal topology of AS v, attribute b is chosen to be advertised to AS z. This results in a length of 199, while selecting c would result in a length of 195.

However, providing a concrete solution to this problem is out of the scope of this thesis.

3.4.2 *A False Notion of Non-Isotonicity*

3.4.2.1 *Problem*

Modeling ASes as abstract nodes can result in advertising suboptimal paths by selecting attributes that do not remain optimal after extension by intra-AS attributes. Figure 3.8 provides an example of this phenomenon for a route computation according to the shortest order $\preceq_S$, selecting the attribute with the shortest length. There are two received attributes, i.e., $b = 99: x, t$ and $c = 100: w, t$. Clearly, $b \preceq_S c$ because of the shorter length. By abstracting the internal topology of AS v, attribute b is chosen to be advertised to AS z. However, $a[zv] \oplus a[Intf.\ 3, Intf.\ 2] \oplus c \preceq_S a[zv] \oplus a[Intf.\ 3, Intf.\ 1] \oplus b$, because $a[zv] \oplus a[Intf.\ 3, Intf.\ 2] \oplus c = 195 : v, w, t$, while $a[zv] \oplus a[Intf.\ 3, Intf.\ 1] \oplus b = 199 : v, x, t$. In other words, the relative preference between b and c is not preserved when they are advertised to the next AS despite the fact that $\preceq_S$ is isotone. This is because the attributes of the two paths from interface 3 to t are in fact not b and c but are $a[Intf.\ 3, Intf.\ 1] \oplus b$ and $a[Intf.\ 3, Intf.\ 2] \oplus c$.

Note that this is not a problem in routing protocols in which every router computes routes from itself to the destination: Routes received from a peer router in the same AS are already extended by the attributes of the intra-AS path that connects them. Therefore, the differences in intra-AS paths are visible in inter-

domain paths. In SCION, however, it is the logically centralized control service of each AS that participates in inter-domain routing, not the border routers.

3.4.2.2 *Solution*

To address this problem, we propose that the control service of each AS selects and extends routes from the perspective of every border router such that it computes the same set of paths that the routers themselves would select. Therefore, for every RC, the control service performs the following steps *for every interface i* in the set of interfaces I:

1. Extend every inter-domain attribute pertaining to the RC that is received from any interface o other than i by $a[io]$, i.e., the attribute of the intra-AS path connecting interfaces i and o. This step builds set $E_i := \{a[io] \oplus c | c \in C[o], o \in I \setminus i\}$, where $C[o]$ represents inter-domain attributes pertaining to the RC that are received from interface o.
2. Perform selection operation on E_i.

In Figure 3.8, when selection is performed on $E_3 = \{a[Intf.3, Intf.1] \oplus b = 199 : v, x, t, a[Intf.3, Intf.2] \oplus c = 195 : v, w, t\}$, the optimal attribute $a[Intf.3, Intf.2] \oplus c = 195 : v, w, t$, is selected.

3.5 PCB EXTENSION

We introduce the IREC PCB extension that provides information to ASes participating in IREC. Each PCB can have at most one IREC extension included in its first AS entry. Only the AS starting an RC, i.e., the AS originating the PCB, can include this extension. ASes receiving such a PCB must not modify this extension. ASes not supporting IREC must ignore the extension, meaning that they must *neither* discard the PCB *nor* remove the extension. This ensures connectivity and allows for partial optimization by IREC-enabled ASes.

This extension contains the following fields:

- SP Identifier (`SP`): Specifying the URL using which the SP can be retrieved from the originating AS. This field is used for source- and destination-defined RCs (cf. Sections 3.3.4 and 3.3.5). If not specified, the PCB is considered for generic RCs.
- Reverse Computation Flag (`Rev`): This flag determines whether the PCB belongs to a source-defined reverse-RC (cf. Section 3.3.5). This flag can only

be set if the SP is not null. If set, an AS extends attributes in reverse and returns a copy of the PCB to the originating AS.

- Interface Group Identifier (IG): An integer specifying the interface group for which the RC computes the routes (cf. Section 3.4.1.2). This field can be set independently of the other two fields.

An RC can be uniquely identified by (Originating ISD, Originating AS, SP, IG, Rev). We call this tuple the *RC Identifier* ($\texttt{ID}_{\texttt{RC}}$).

Note that IREC can also work on PCBs without such an extension. PCBs without the extension are considered for all parallel generic RCs. This is because generic RCs are oblivious to SP and Rev fields, and IG is optional.

3.6 CONTROL SERVICE DESIGN

This section describes a new design of the SCION control service that supports IREC routing mechanisms (cf. Section 3.3). To support IREC, a SCION AS deploys this new version of the control service. This new version is backward compatible and can be incrementally deployed.

3.6.1 *Overview*

The IREC-enabled control service has three main components: (1) The *ingress gateway*, receiving PCBs, storing them, and filtering them based on the AS-local policies, (2) *SP container*s (SPCs), executing SPs in isolation, and (3) The *egress gateway*, collecting the selected PCBs from the SPCs, filtering duplicate ones, and advertising them further to neighboring ASes. The egress gateway is also responsible for starting new RCs through originating PCBs. Figure 3.9 provides an overview of this design of the control service.

3.6.2 *Ingress Gateway*

When receiving a PCB from a neighboring AS, the ingress gateway verifies the included signatures and whether the path constructed by the PCB complies with the local AS' policies. The ingress gateway then stores the PCB in its *ingress database*. The ingress database stores PCBs indexed by their RC identifiers $\texttt{ID}_{\texttt{RC}}$, i.e., the tuple (Originating ISD, Originating AS, SP, IG, Rev) defined in Section 3.5. The ingress gateway retrieves the SP from the originating AS if it has not yet cached the SP

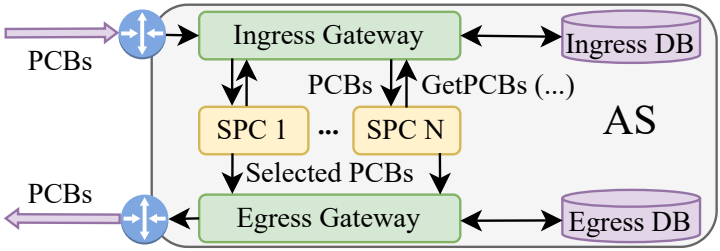


Figure 3.9: Components of the new SCION control service design to support IREC.

and caches it as long as there are PCBs specifying that SP. Note that there is no cyclic dependency in SP retrieval: The originating AS can always be reached via the path contained in one of the PCBs. The gateway periodically removes expired PCBs from the database.

3.6.3 *SP Containers*

A SPC is a sandboxed environment for executing SPs. We design two types of SPCs: (1) Static, which executes one generic SP configured by the local AS, and (2) Dynamic, which, in a round-robin way, executes various destination- or source-defined SPs.

Instances of both SPC types request PCBs from the ingress gateway in a round-robin way. The ingress gateway serves requests of these two types of SPCs differently. A static SPC is provided with all received PCBs. A dynamic SPC is provided with the set of PCBs associated with an unprocessed $\texttt{ID}_{\texttt{RC}}$ together with the corresponding SP. An unprocessed $\texttt{ID}_{\texttt{RC}}$ is one with at least one new received PCB that is not processed by a dynamic SPC. After receiving the PCBs from the ingress gateway, a static SPC executes its statically configured SP per triplet of (Originating ISD-AS, Interface group at the originating AS, Interface of the local AS). A dynamic SPC, on the other hand, executes the received SP per interface of the local AS since the received PCBs in each round all belong to the same $\texttt{ID}_{\texttt{RC}}$. Before every execution, a SPC excludes the PCBs that are not allowed to be propagated on the considered interface according to the AS-local policies. Upon execution, a SPC provides the SP with (1) the set of PCBs allowed to be propagated on the considered interface according to AS-local policies, and (2) available performance metric information about the intra-AS paths connecting all AS' interfaces to the considered interface,

allowing for extending PCB attributes before selection (cf. Section 3.4.2.2). The SP returns the set of selected PCBs for that execution. SPC and SPs communicate inputs and outputs using a standardized interface (cf.Section 3.7). Once all executions are performed, the SPC communicates the selected PCBs for each interface to the egress gateway. Then, it requests the ingress gateway for PCBs again.

3.6.4 *Egress Gateway*

3.6.4.1 *PCB Origination*

When creating new PCBs, the egress gateway of the originating AS adds the attribute of the inter-domain link from which the PCB is being emitted. To benefit from a destination- or source-defined RC or optimization per interface groups, the egress gateway must add an IREC extension with the respective SP URL (`SP`), the reverse computation flag (`Rev`), or the interface group identifier (`IG`, cf. Section 3.5). The egress gateway then signs the PCB and sends it to the ingress gateway of the neighboring AS connected to the considered interface.

3.6.4.2 *PCB Propagation*

When receiving selected PCBs for each interface from the SPCs, the egress gateway filters them by consulting its *egress database*. If a received PCB is not yet present in this database, the PCB is inserted together with the egress interface IDs for which it is selected. The egress gateway then performs the extension operation for each PCB selected for each interface: It appends an AS entry that, in addition to other necessary information, contains (1) the attributes of the intra-AS path connecting the interfaces where the PCB entered the AS and is going to exist the AS, and (2) the attributes of the inter-AS links connected to those two interfaces. The egress gateway sends the extended selected PCB over the selected inter-domain interface to the ingress gateway of the neighboring AS on the other end of the interface. If the `Rev` flag is set in the PCB, a copy of the PCB is also sent to its originating AS. If a received PCB is already present in the egress database, which can happen for PCBs selected by multiple SPCs or in different rounds of selection, the PCB is only propagated over those interfaces whose IDs are newly added to the database. Similarly to the ingress database, (soon-to-be) expired PCBs are removed from the egress database.

3.6.4.3 *Path-segment Registration*

To make the computed path segments available to endpoints, the egress gateway registers them at the path service of the AS. To allow usability, it tags each segment with the identifiers of the SPs by which it is computed.

3.7 STANDARDIZATION AND DEPLOYMENT

The IREC architecture, as presented so far, gives individual ASes almost unbounded freedom and flexibility in deploying multiple parallel SPs. However, routing and path optimization are collaborative efforts, and their outcomes greatly benefit from having as many ASes as possible participating in the process. Therefore, the architecture proposed in this chapter would not be complete without a *standardization model* to guide ASes toward shared SPs and implementation choices such that globally desirable goals are always achieved.

Borrowing from software engineering terminology, we thus propose a tiered standardization model where architectural features are categorized by their criticality to Internet-wide connectivity and their expected stability over time.

Feature Standardization Model We identify three standardization tiers for IREC's features: (1) *stable* features, which are essential to global connectivity and should be changed infrequently, (2) *beta* features, comprising additional functionality—e.g., optimizing paths on elementary optimality criteria—which are not critical to connectivity, but can greatly benefit from widespread deployment, and finally (3) *nightly* features, which can change extremely frequently—e.g., optimizing paths for arbitrary criteria of specific applications.

We now provide more details on each of these tiers.

Stable Features These features are the foundation of the IREC architecture and are ideally standardized once, with minimal future updates to maximize their global adoption. The stable features are:

- The basic format of PCBs, compatible with the SCION's legacy PCB format [10], containing the necessary information to specify paths.
- The PCB extension (cf. Section 3.5) for destination- and source-defined RCs (cf. Sections 3.3.4 and 3.3.5) and interface groups (cf. Section 3.4.1.2).
- The interface between SPCs (cf. Section 3.6.3) and SPs, which needs to be standardized to allow low-effort deployment of new SPs and to enable destination-

and source-defined RCs. With a standardized interface, SPs can be deployed ubiquitously, favoring adoption and extensibility.

- An RC to guarantee global connectivity. Although designing an RC for fast and reliable global connectivity is beyond the scope of this thesis, the default choice can be the current SCION RC.

Any AS can participate in IREC as long as it implements these features correctly.

Beta Features Elementary optimality criteria—for common path performance metrics such as latency and bandwidth—are likely to be widely adopted and used by many different applications. Therefore, to promote interoperability and consistent optimization, it is essential to have a standard definition of (1) which metrics are used, (2) how the metrics are computed and extended, (3) how each metric is encoded in PCBs, and (4) which SPs are globally preferred for each set of elementary criteria. Finally, note that extending the list of metrics must be a straightforward task, not hindering the development of new SPs. These rapid updates could be supported, for example, by publishing new metrics and SPs to public append-only lists. ASes can choose whether or not to participate in path optimization on elementary criteria and, if so, which SP to use.

Nightly Features In this tier, destination- and source-defined RCs (cf. Sections 3.3.4 and 3.3.5) replace standardization: These mechanisms ensure that all ASes run the same SP, which is specified within PCBs by the end AS. SPs disseminated in this way are not standardized, allowing end ASes to freely choose any SP they want to deploy. These SPs use the publicly available *beta* metrics.

3.8 CASE STUDY: OPTIMIZING PATHS FOR CLIENT-SERVER APPLICATIONS

We study how the IREC architecture can be used to enhance the quality of experience for client-server applications by allowing optimization of SCION core-path segments for the criteria desired by each application. In this study, we assume a common case in modern applications, where the client application is developed by the same entity that develops and operates the server side. For example, the YouTube app is developed by Google, and the content is served by Google-developed servers on the cloud run by Google. In such cases, the entity (e.g., Google) knows which path optimality criteria suit either direction of the communication (i.e., client-to-server and server-to-client) the best. This also applies to tenants of cloud providers.

Which Side Should Start the RCs? We argue that the server side of such applications should leverage the mechanisms provided by IREC to optimize paths in both directions between clients and servers. The reasons are fourfold:

- It is significantly more scalable to start one RC from the server side (or a handful if the server is in the customer cones of multiple core ASes) than to start thousands of RCs from all core ASes with clients in their customer cones.
- A cloud provider that hosts a service can operate core AS(es) and have full control over its core-segment computations or have an influence on its providers' core-segment computations. This control can be offered as a service to cloud tenants. On the other hand, client applications are not expected to have a significant, if any, influence on which RCs their ISP starts.
- Starting RCs from the client side would require either the involvement of end users or the standardization and development of an interface between client applications and ISPs to automate the expression of the SP to ISPs. None of these are required on the server side. The cloud provider offers a user interface (UI) or an infrastructure as code (IaC) model to tenants, using which they can define their SPs.
- Changing SPs on the server side is dramatically easier, in terms of operational effort, than on the client side. On the server side, the defined SP on the cloud UI or IaC needs to be updated. On the client side, the application developer would need to release a patch for (up to) billions of client applications.

Required RCs To optimize core-path segments between clients and a server of an application, the core AS whose customer cone includes the server's AS needs to (1) start a destination-defined optimization that computes paths from clients to the server according to the criteria for reaching the server, and (2) start a source-defined optimization that computes paths from the server to clients according to the criteria for reaching the clients. Note that the SPs of these RCs can use different orders. For example, for a streaming service, a high-bandwidth path may be required from the server to the clients, while a low-latency path may be required for acknowledgment packets from the clients to the server.

Path-segment Lookup Once these core segments are computed, they need to be looked up by the endpoints. We argue that the segments for both directions of the communication should be looked up on the client side. This is because path

lookup is an expensive operation and can overwhelm a server for a large number of clients. Furthermore, the core AS on the client side receives and stores optimal segments for both directions of the communication. On the contrary, the core AS on the server side receives the optimal paths only for the server-to-client direction of communication through source-defined RCs.

IREC-aware Core Segment Selection at Endpoints The core-segment lookup operation can result in multiple segments for each direction of communication. If they are computed by a single RC, a random one can be returned to the client. However, it can be the case that the segments are computed by different RCs, and each is optimal for different sets of criteria. In such cases, the client application should know which RC is tailored to its communication criteria. To solve this problem, we propose the IREC-aware core-path segment selection procedure. This means that endpoints choose the segments that are computed by their desired RC instead of performing segment optimization themselves. This saves endpoints from repeating the same computation already performed by RCs. In addition, it significantly reduces the number of segments returned to clients from the path service, enhancing the scalability of segment lookup. Most importantly, client applications remain much simpler, as changing the optimality criteria does not require modifying the client application code.

We propose the following IREC-aware segment selection procedure:

1. The client endpoint looks up for core-path segments to and from the destination ISD where the server AS is located.

2. For each direction, if there is only one RC, the path service returns the optimal segments computed by the RC. If there is more than one RC, the path service returns *a* segment together with the list of RC identifiers ($\texttt{ID}_{\texttt{RC}}$, cf. Section 3.5) existing for each direction of communication.

3. If the result of the lookup contains a list of $\texttt{ID}_{\texttt{RC}}$s, the client contacts the application server to ask for the desired $\texttt{ID}_{\texttt{RC}}$ out of the provided list. Note that the application developer who controls the server is aware of both the desired criteria for the application and the corresponding RCs.

4. The client performs another segment lookup for each direction of the communication, providing the path service with the $\texttt{ID}_{\texttt{RC}}$ chosen by the server. The path service returns the optimal segments computed by the specified RC.

3.9 IMPLEMENTATION AND EVALUATION

We implement a proof-of-concept of IREC in a fork of the SCION official open-source codebase [42] and evaluate it through emulation using the Kathará network emulator [43, 45].

3.9.1 *Implementation*

We implement the control service proposed in Section 3.6 in the Go programming language as two separate applications bundled in two Docker [46] images: One application that implements the aggregated functionalities of ingress and egress gateways (GWs), and another that implements SPCs. An instance of the SPC application can be configured to be a static or a dynamic SPC. SPCs communicate with gateways using remote procedure (gRPC) [47] calls. We use SQLite [48] for the ingress and egress databases. To ensure interoperability with neighboring ASes not supporting IREC, the ingress gateway listens on the same service address as the globally standardized service address of the legacy control service [49].

SPCs To reduce maintenance effort and cost, we develop a unified implementation for static and dynamic SPCs. The type of SPC is configured upon initialization. SPCs use the extended Berkeley Packet Filter (eBPF) [21] technology to execute SPs. This allows for (1) extensibility, as SPs can be written in various languages and compiled into eBPF bytecodes, and (2) safety, as these SPs run within isolated memory regions. We use the user-space variant of eBPF, uBPF [22], as we do not want to run alien code from other ASes in the kernel. uBPF runs the eBPF bytecode within a virtual machine (VM) in the user space. As uBPF has limited helper functions compared to eBPF, we use the libxbpg [50] library, which enhances the uBPF VM with more helper functions. Since uBPF is a library written in C and SPCs are written in Go, we define a C interface for calls to the library, export it, and invoke it from Go using Cgo [51]. A dynamic SPC creates a new VM for every execution of an SP and destroys it afterward. A static SPC creates the VM only once upon initialization. To enhance performance, we use the just-in-time compiler to translate eBPF bytecode to x86 instructions upon creating the VM. Before every execution, a SPC prepares the memory region for the SP. The memory region is used to provide PCBs to the SP as well as a working memory. Extended attributes are provided to the SP using FlatBuffers [52], allowing for fast access to variable-sized constructs without parsing or unpacking. We develop one SP in C that implements the same selection operation as the one in the master branch of

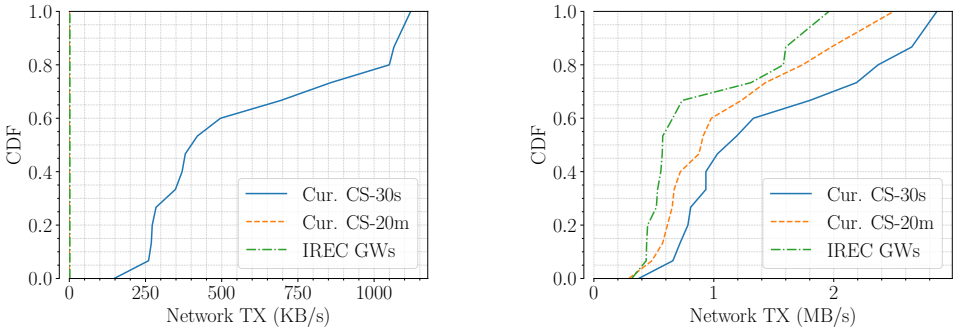
the SCION codebase at the time of writing this thesis. The selection operation sorts paths according to their AS-hop lengths and selects the N-shortest paths with a configurable N[2].

3.9.2 *Evaluation*

We evaluate our implementation by emulating a SCION network comprising 15 core ASes and a total of 581 inter-AS links. This topology is extracted from the CAIDA AS relationships with geographic annotations [53] data set, containing relationships and the locations of links between more than 12 000 ASes across the Internet. The extracted ASes are the ones with the highest degree in the data set. The distance between all pairs of extracted ASes, except one pair, is one. The distance of that single pair is two.

Due to the large size of the topology, we use a variant of the Kathará emulator, Megalos [54], that performs network emulations in a Kubernetes [55] cluster. Megalos automates the creation of the cluster, deployment of containers of emulated network devices, and the connections between them. For our emulations, we instantiate 10 compute nodes in a data center of a cloud provider, all of the same instance type, with 8 vCPUs and 16 GB main memory. We carry out three experiments, each for one hour: (1) an IREC experiment in which all ASes deploy IREC with only one *dynamic* SPC and each AS starts one destination-defined RC (cf. Section 3.3.4), (2) Cur. CS-30s, an experiment in which ASes deploy the current SCION control service (CS) with a beacon propagation interval of 30 seconds, and (3) Cur. CS-20m, an experiment with the current control service with a beacon propagation interval of 20 minutes. In all experiments, PCBs are originated only once at the beginning of the experiment. In SCION experiments, we have a total of 83 virtual devices (containers), from which 53 are border routers, 15 are control services, one per AS, and another 15 are end hosts, one per AS. In the IREC experiment, there are 98 devices, where the 15 additional devices are the SPCs, one per AS. In all experiments, we disable the signature verification of PCBs as it is an expensive operation requiring more resources for emulations. Note that signature verification is a common operation for all PCBs and requires the same per-PCB effort in the current control service and IREC. The additional effort for signature verification would have been proportional to the number of PCBs received by the control service.

2 The SP selects $N-1$ shortest paths and one path that is most disjoint to the other $N-1$s. For simplicity, we say it selects the N-shortest paths.



(a) Median

(b) Peak

Figure 3.10: Bandwidth usage of containers.

We sample CPU, memory, and bandwidth usage of every container every 5 s during each experiment and collect them using Prometheus [56].

Bandwidth Figure 3.10 illustrates the network transmit bandwidth use of selected containers in the three experiments. Figure 3.10a shows three distributions of the median bandwidth usages of (1) the current CS containers in the Cur. CS-30s experiment, (2) the current CS containers in the Cur. CS-20m experiment, and (3) the gateway containers (GWs) in the IREC experiment. Similarly, Figure 3.10b demonstrates the distributions of the peak bandwidth usages. The median and peak usages for each container are computed over the time series collected over the course of each experiment.

Among the three experiments, IREC and Cur. CS-20m both have a median TX rate of 0. This is because both do not propagate PCBs most of the time but for different reasons. Cur. CS-20m propagates PCBs only every 20 min, meaning that it does not propagate anything in 19 min out of every 20 min. IREC, on the other hand, keeps track of propagated PCBs in the egress gateway. PCBs selected by the SPC are not propagated if they have already been propagated (cf. Section 3.6.4). Since there is only one origination of PCBs, and the SP selects the same set of PCBs repeatedly, PCB propagation occurs only at the very beginning of the experiment. The peak network transmit rate, however, is similar in all experiments. This is because the peak rate represents the peak PCB propagation rate in all experiments, which is equal across experiments because the selection operation is the same across all of them.

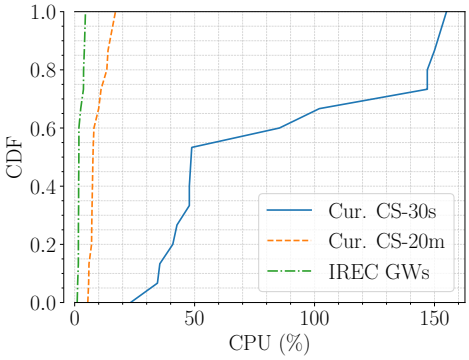
CPU Figure 3.11 illustrates the CPU use, in the percentage of one CPU core, of selected containers in the three experiments. Figure 3.11a shows three distributions of the median CPU usages of (1) the current CS containers in the Cur. CS-30s experiment, (2) the current CS containers in the Cur. CS-20m experiment, and (3) the gateway containers (GWs) in the IREC experiment. Similarly, Figure 3.11b demonstrates the distributions of the peak CPU usages. Lastly, Figure 3.11c shows the distributions of median and peak CPU usages of the SPC containers in the IREC experiment. The median and peak usages for each container are computed over the time series collected over the course of each experiment.

Among the three experiments, IREC has the lowest median CPU usage of all because of a very low rate of PCB propagation as a result of filtering and not propagating duplicate PCBs at the egress gateway. Cur. CS-20m also has much lower CPU usage than Cur. CS-30s because of a much lower PCB propagation rate, i.e., 40× lower. The peak CPU usages, however, are similar in all experiments because of a similar peak PCB propagation rate.

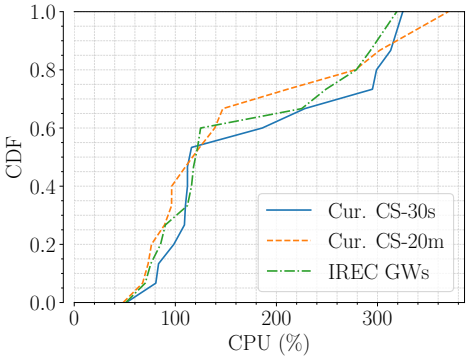
The most important result is the very low CPU usage of SPCs, both at the peak and the median, compared to the current control service and gateways. This suggests the feasibility and scalability of SPs in uBPF VMs. This allows for running many SPCs in parallel with modest compute resources, allowing for the deployment of many SPs and the extensibility of routing criteria. Note that SPCs are always running and are executing SPs every 5 s even though they are filtered by the egress gateway.

Memory Figure 3.12 illustrates the memory use, in MB, of selected containers in the three experiments. Figure 3.12a shows three distributions of the median memory usages of: (1) the current CS containers in the Cur. CS-30s experiment, (2) the current CS containers in the Cur. CS-20m experiment, and (3) the gateway containers (GWs) in the IREC experiment. Similarly, Figure 3.12b demonstrates the distributions of the peak memory usages. Lastly, Figure 3.12c shows the distributions of median and peak memory usages of the SPC containers in the IREC experiment. The median and peak usages for each container are computed over the time series collected over the course of each experiment.

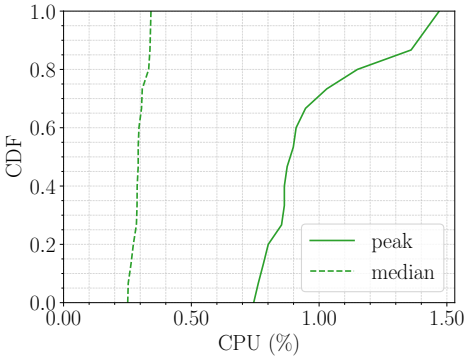
We see similar trends for memory usage as we see for CPU usage, suggesting the superiority of IREC and the feasibility and scalability of SPCs (less than 13 MB peak memory use for any SPC), enabling the extensibility of routing criteria.



(a) Cur. CS vs.IREC GWs - Median



(b) Cur. CS vs. IREC GWs - Peak

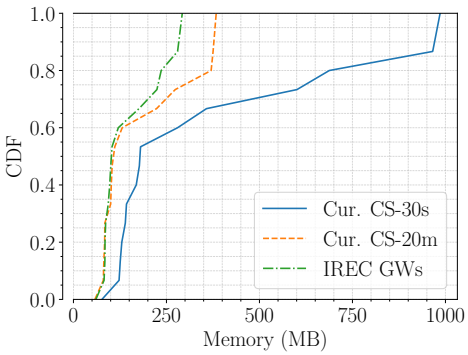


(c) SPC

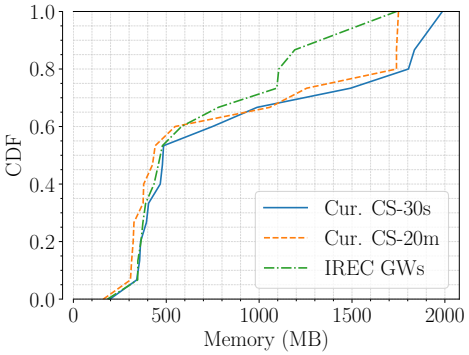
Figure 3.11: CPU use of containers in percentage of one CPU core.

3.10 LARGE-SCALE SIMULATIONS

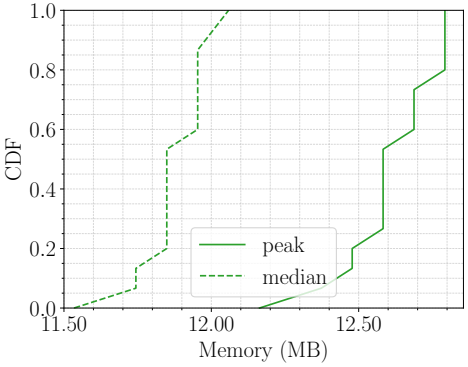
In this section, we study the inter-domain behavior of parallel routing in a large network using simulations. By abstracting the implementations of IREC and the current control service, large-scale simulations are significantly less resource-intensive than large-scale emulations (cf. Section 3.9.2) for the same large topology.



(a) Cur. CS vs. IREC GWs - Median



(b) Cur. CS vs. IREC GWs - Peak



(c) SPC

Figure 3.12: Memory use of containers in MB.

3.10.1 *Simulation Setup and Topology*

We implement IREC in the ns-3-based [26] SCION simulator we develop. The topology consists of 500 ASes and more than 90 000 inter-AS links. This topology is derived from the CAIDA AS relationships with geographic annotations [53], which we also used for emulations (cf. Section 3.9.2). The extracted 500 ASes are the highest-degree ones in the original topology. To extract them, we iteratively remove the lowest-degree ASes from the CAIDA dataset. The locations of inter-AS links in this topology allow for the estimation of the propagation delay between border routers of each AS using the great circle distance. We also annotate inter-AS links

with bandwidths using degree gravity, which sets the link capacity proportionally to the product of the degrees of ASes at the two ends of the link. The width of a link is chosen from the set $\{1, 10, 25, 40, 100, 400\}$.

3.10.2 *Experiments*

We perform the following experiments:

Parallel RC Experiments

- All ASes advertise the *union* of optimal attributes according to *two* total orders $\preceq_{\alpha-EI-L}$, one with $\alpha = 1$ and the other with $\alpha = \frac{1}{2}$. Each total order selects the attribute with the lowest linear combination of non-dimensionalized environmental impact and non-dimensionalized length (latency), i.e., $\alpha \times EI_{nd} + (1-\alpha) \times l_{nd}$. To non-dimensionalize length (latency), we divide latency values by the propagation delay of a fiber of 100 km length, i.e., 0.5 ms. To non-dimensionalize environmental impact, we divide the values by the environmental impact of one core router powered by coal-fired power plants, i.e., 0.001 94 gCO_2/Gbit (cf. Chapter 4). We refer to this experiment as $\bigcup_{\alpha \in \{\frac{1}{2}, 1\}} \sqcap_{\preceq_{\alpha-EI-L}} A$, where A represents the set of attributes and $\sqcap_{\preceq_{\alpha-EI-L}} A$ represents the set of selected attributes from A according to order $\preceq_{\alpha-EI-L}$.
- Similar experiment but with three $\alpha \in \{\frac{1}{2}, \frac{2}{3}, 1\}$. We refer to this experiment as $\bigcup_{\alpha \in \{\frac{1}{2}, \frac{2}{3}, 1\}} \sqcap_{\preceq_{\alpha-EI-L}} A$.

Single RC Experiments

- All ASes advertise the dominant attributes on the intersection of the two total orders $\preceq_{\alpha-EI-L}$ with $\alpha \in \{\frac{1}{2}, 1\}$. We refer to this experiment as $\mathcal{D}_{\bigcap_{\alpha \in \{\frac{1}{2}, 1\}} \preceq_{\alpha-EI-L}}(A)$.
- All ASes advertise the dominant attributes on the intersection of the three total orders $\preceq_{\alpha-EI-L}$ with $\alpha \in \{\frac{1}{2}, \frac{2}{3}, 1\}$. We refer to this experiment as $\mathcal{D}_{\bigcap_{\alpha \in \{\frac{1}{2}, \frac{2}{3}, 1\}} \preceq_{\alpha-EI-L}}(A)$.
- All ASes advertise the dominant attributes on the product order of the width and length (latency). In the product order, (w, l) is preferred to (w', l') if they are different and $w \geq w'$ and $l \leq l'$ [44]. We refer to this experiment as $\mathcal{D}_{\preceq_{W \times L}}(A)$.

In all experiments, there is only one PCB origination event at the beginning of the simulation. PCB selection and propagation at each AS happens periodically with the same period across all ASes. ASes are assumed to be perfectly time-synchronized. At every interval i, only PCBs received up to the previous interval $i-1$ are considered as candidates for the selection operation. In all experiments, if there are multiple paths corresponding to the selected attribute, only one of the paths is selected. To ensure determinism, with regard to selection and not arrival order, we use the following tie-breaker: (1) The path with the fewest hops is preferred; (2) Among paths with the same number of AS hops, the one whose sequence of AS numbers and interface identifiers is lexicographically the smallest is selected. This tie-breaker ensures determinism *only* in the *simulated periodic beaconing* explained above, where non-determinism in the arrival order of PCBs cannot impact path selection. In such a setup, it is guaranteed that selected paths with the same AS-hop length l are propagated and received by the next AS at the $l-1^{th}$ interval of the simulation (starting from 0) and are considered as candidates only at the next interval, i.e., l^{th} interval.

3.10.3 *Results*

Figure 3.13 compares the results of different experiments.

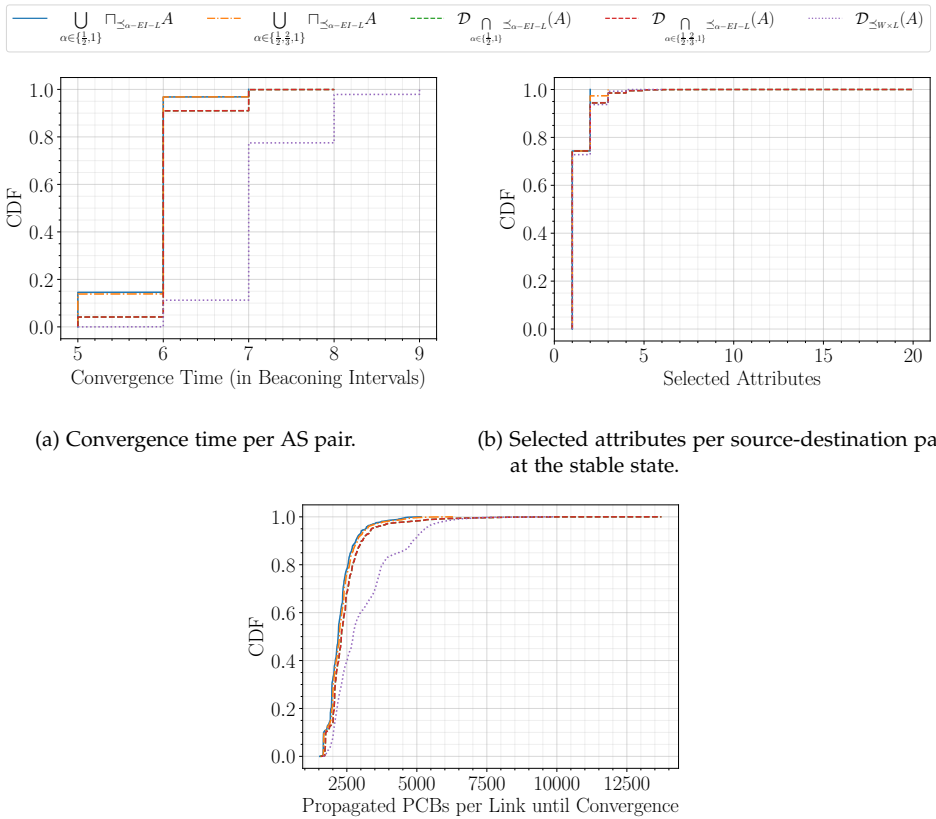
Convergence Figure 3.13a illustrates the distributions of the first interval since when an AS s does not receive any new PCBs pertaining to a destination AS t in different scenarios. In other words, it shows the convergence time, in the number of intervals, for all pairs of ASes. Although the diameter of the graph, i.e., the number of links in the longest shortest path, is 3, computing the optimal paths can take at least 5 and at most 9 beaconing intervals, depending on the experiment. The two experiments with parallel RCs have the quickest convergence time, with more than 95% convergence in fewer than 7 intervals. For routing on the dominant attributes of the intersection of linear combinations, more than 90% of convergence happens in fewer than 7 intervals. The longest convergence time happens when routing on the dominant attributes of the product order of width and length. For that RC, around 10% of convergence happens in fewer than 7 intervals, around 75% in fewer than 8, and around 97% in fewer than 9 intervals. This RC has the maximum full convergence of all, which is after 9 intervals, while it is 7 for parallel scenarios and 8 for dominant attributes of linear combinations.

Selected Attributes at the Stable State Figure 3.13b shows the distributions of the number of attributes selected by any AS s from any of its interfaces pertaining to any destination AS t in the stable state (after convergence) in different scenarios. The number of data points for each experiment is more than 95 million. In all scenarios, only one attribute is selected for more than 70% of pairs of interface-AS pairs. Since the selected attributes for parallel RCs are the unions of selected ones by each of the RCs, having one selected attributes implies that the RCs running in parallel select the same attribute for more than 70% of interface-AS pairs. The maximum number of selected attributes for two and three parallel RCs is two and three, respectively, which is also due to taking the union of selected attributes by two and three RCs. According to the figure, dominant attributes on the intersection of linear combinations are less than 6 for 99% of the interface-AS pairs but can be as many as 20 for a small percent of interface-AS pairs. The distribution of the number of dominant attributes on the product order of width and length is very similar to that of dominant attributes on the intersections of linear combinations except for a much shorter tail ending at 6.

Message Complexity Figure 3.13c shows the distributions of the number of PCBs propagated on every inter-domain interface throughout experiments. The number of data points for each experiment is more than 192 000 (corresponding to 96 000 links, each connecting two interfaces). We can clearly see that the message complexity of parallel routing with two linear combinations is lower than that of parallel routing with three linear combinations, which is lower than that of routing on dominant attributes of intersections of linear combinations, which is lower than that of routing on the dominant attributes on the product order of width and length. It can also be seen that routing on the dominant attributes on the product order of width and length has a noticeably higher message complexity than all other scenarios. Lastly, it is observable that routing on dominant attributes (of different orders) has long tails for message complexity. This long tail ends at above 12 500 PCBs per interface for routing on dominant attributes on intersections of linear combinations. This is more than two times the maximum PCBs per interface for parallel routing with three linear combinations.

3.11 DISCUSSION

Executing Untrusted Code Executing code from other ASes in destination- or source-defined computations can be controversial from the security viewpoint. However, we argue that authenticated PCBs and our design and implementation,



(a) Convergence time per AS pair.

(b) Selected attributes per source-destination pair at the stable state.

(c) Total propagated PCBs on each interface.

Figure 3.13: Comparing parallel routing to routing on dominant attributes.

together with some best practices, eliminate the potential of the most important threats. Two important malicious uses of an SP are to use it to access confidential information from other ASes or to launch a denial of service (DoS) attack on the control plane of the Internet by consuming excessive resources and overwhelming SPCs and eventually causing disconnectivity. First, with the signed and authenticated PCBs and SPs in SCION, the identity of an AS that uses malicious SPs is known to the whole network. This discourages them from such behavior as it hurts their reputation and business. Second, the design of IREC (cf. Section 3.6) completely isolates SPCs from each other. Therefore, even if a SP is overwhelming

for one SPC, other SPCs can continue their operation, preventing a denial of service on other route computations. To guarantee connectivity, ASes need to have at least one static SPC that runs only one *generic* SP. Third, the implementation of SPs as uBPF bytecodes ensures that they are run in isolated VMs inside each SPC (two levels of isolation, SPCs and uBPF VMs). uBPF programs have access to an isolated memory region; thus, they cannot access confidential information. In addition, we do not allow them to interact with the network stack. They return the selected PCBs to the SPC itself. As a security best practice, we also advise against storing confidential information on hosts running SPCs.

Ensuring that ASes Execute Destination- and Source-Defined SPs Destination- and source-defined route computations are only useful if ASes are committed to executing their associated SPs. We argue that ASes willingly execute the SPs as it is to their best benefit; thus, there is no need for an enforcement mechanism. The reason is that end ASes specifying such SPs want to later use the optimal paths computed by those SPs for sending traffic. Therefore, if an AS on the optimal path executes the SP, the traffic goes through that AS, increasing its revenue. Thus, they have a strong incentive to execute the SPs. If an AS can somehow realize that it is not on the optimal path for a SP, it probably does not execute it. In this case, there is no harm in not executing the SP because even if it were executed, the resulting path would not have been selected by endpoints. It is, however, highly unlikely that an AS can realize whether or not it is on an optimal path for an SP because (1) an AS only sees the path attributes up to its AS and not further downstream, and (2) a non-curious AS only sees the bytecode of the SP and not the source code; thus, they do not see the selection criteria unless they perform analysis on the bytecode.

3.12 RELATED WORK

Extensible Inter-Domain Routing XIA [57], Trotsky [58], and xBGP [20] take significant steps towards an extensible Internet. XIA proposes an expressive Internet architecture with native support for multiple principals and the ability to evolve to accommodate new ones. Trotsky enables the introduction of new designs and Internet architectures through its backward-compatible framework, and xBGP [20] enables BGP to extend by introducing user-defined and dynamically addable eBPF bytecodes. Thus, network operators can provide and deploy their own extensions without waiting for standardization or vendors. However, they do not consider multi-criteria path optimization as a design objective and do not provide end domains with means of expressing their diverse criteria to the control plane. In

particular, in xBGP, each AS decides its routing policy and optimization criteria independently, not providing means for global optimality guarantees for different criteria. Furthermore, each AS can use only a single selection operation at a time to optimize paths and is oblivious to the criteria of endpoints outside of its domain.

Multi-Criteria Path Optimization Multi-objective path problems consider path performance metrics as the Cartesian product of elementary metrics, extending either by addition, e.g., latency, or by min/max, e.g., capacity, and ordering paths partially using the product order of their term-wise total orders [59–61]. These problems, however, consider only a limited category of performance metrics. Sobrinho et al. [40] guarantee optimality on multiple criteria by taking the intersection of all criteria, finding the largest isotonic reduction of the intersection if it is not isotonic, and computing the dominant attributes, i.e., the ones to which no other path is preferable and which are incomparable to each other. Despite building a sound theoretical foundation for multi-criteria path optimization, which we also build upon, extensibility is not their main objective: They do not seek solutions to overcome the operational obstacles to modify the routing criteria over time.

On-Demand Routing Yampolskiy et al. [62] propose a resource reservation algorithm to establish connections with multiple QoS constraints per endpoints' requests in inter-domain networks. Route Bazaar [63] introduces a system enabling customers and ASes to agree on routes according to QoS criteria. These methods have two fundamental differences with IREC: (1) Their path discovery is accompanied by a QoS agreement between ASes, while IREC is a general-purpose routing architecture, and (2) They route only on routing constraints, i.e., to satisfy thresholds for performance metrics, while IREC's main goal is to compute optimal (the best) paths for certain criteria. However, IREC can also be used to satisfy constraints through destination- and source-defined RCs.

3.13 SUMMARY

Despite tremendous advancements witnessed in the realm of information technology over the past decades, along with a continuous increase in dependence on communication networks, inter-domain routing has endured a conspicuous lack of transformation over the course of the past 25 years. In particular, despite the ever-increasing diversity of applications' communication requirements, BGP has largely remained static since its inception.

IREC overcomes those limitations by enabling the extension of routing optimality criteria. Taking advantage of the opportunities provided by the SCION path-aware Internet architecture, IREC introduces parallel route computations, laying the foundation for the extension of path optimization criteria. Building on parallel route computations, we introduce source- and destination-defined route computations, enabling both end ASes of traffic to express their path optimality criteria to the control plane without waiting for standardization, vendor implementation, or adoption by operators.

By emulating our uBPF-based implementation in the SCION codebase for a realistic topology, we showed the practicality of the introduced mechanisms. Through simulations with ns-3 with large realistic topologies, we demonstrate that parallel route computations, i.e., the building block of IREC, require less effort, in terms of message complexity, than a single computation selecting dominant attributes on the intersection of the same criteria.

IREC is the necessary foundation for a competitive path market. Without IREC, the control plane would compute paths for a limited and static set of criteria that are not suitable for the wide variety of evolving criteria of different customers. Such paths would not be able to attract a variety of customers to the market.

IREC also opens up new opportunities for exciting research in inter-domain routing, ultimately leading to enhanced communication quality for endpoints and applications—all in real time, with negligible effort and no interruption in communication.

Part II

DISSEMINATING PATH INFORMATION

PART OVERVIEW

In the previous part of the thesis, we develop the mechanism that discovers optimal inter-domain paths according to the various criteria of endpoints. However, such path optimization can only be performed if path performance information with regard to the relevant performance metrics is provided. Due to the distributed nature of inter-domain networks, providing such information requires designing dedicated systems to disseminate the required information throughout the network. Furthermore, as different path performance metrics are calculated in different ways, dedicated systems need to be designed to disseminate information about each metric.

In the two chapters of this part of the thesis, we develop two systems for disseminating information about (1) carbon intensity (cf. Chapter 4) and (2) propagation latency (cf. Chapter 5) of inter-domain paths. These systems provide examples for developing such systems for other performance metrics.

The designed information dissemination systems in this part, however, do not propagate real-time information about the performance of each path, mainly to ensure the scalability of the systems. Furthermore, the provided information may not represent the performance of a path under all different circumstances, e.g., different types of packets. In Part III of this thesis, we develop a complementary system that enables endpoints to examine the experienced performance of inter-domain paths and their constituent segments associated with every ISP on a path. Such a mechanism allows endpoints to avoid ISPs with undesirable performance, thus encouraging ISPs to improve their infrastructure and provide better performance to attract more traffic.

4

CARBON-AWARE GLOBAL ROUTING IN PATH-AWARE NETWORKS

In this chapter, we develop the first of the two systems for disseminating inter-domain path performance information, and it is specifically designed for the carbon intensity of inter-domain paths. The contributions of this chapter are based on a publication at ACM e-Energy 2023 [2].

4.1 INTRODUCTION

Growing concerns regarding climate change encourage companies to measure and reduce their carbon footprint, i.e., the amount of carbon emission that can be attributed to them. This also applies to their use of Information and Communication Technology (ICT), as ICT has a notable contribution of 2.7% to global CO_2 emissions [64], which is expected to grow significantly – approximately four times – until 2030 [65]. Hence, reducing the carbon footprint of ICT use is becoming increasingly relevant for enterprises, manifesting in carbon-neutrality statements of major technology corporations.

While these efforts are laudable and impactful, promising opportunities for further carbon-footprint reduction exist. Indeed, previous research has identified a range of such opportunities. However, most of these proposals apply to local aspects: intra-domain networking (i.e., within a single domain), data-center optimizations, or neighbor-domain cooperation (cf. Section 4.7). In contrast, inter-domain networking (i.e., among multiple domains), which accounts for around 13% of total ICT energy consumption, has so far received less attention. An exception is the work by Zilberman et al. [66], who identify carbon-aware networking as a high-potential research area and sketch the concept of carbon-intelligent routing, i.e., to leverage differences in network paths' carbon intensity (i.e., carbon emission per unit of data transmitted) to reduce the carbon footprint of communications.

Previous research on green inter-domain networking applies carbon efficiency to the optimization metric of the Border Gateway Protocol (BGP) [67]. Unfortunately, this direction faces several challenges. **Inefficient Green Route**: A strict carbon-optimal path can result in a highly inefficient end-to-end path in terms of monetary cost, latency, bandwidth, loss, or jitter (cf. Section 4.6.1). Depending on the application requirements, an optimization subject to all these constraints needs to be made, requiring path selection within a fine-grained metric space. **Ossification**: Carbon-optimal paths can thus only be offered as additional options, not as replacements for the conventional BGP route. When using BGP to provide carbon-efficient alternative paths, routers would thus require multiple forwarding tables, and packets would need to indicate the desired optimization criteria. Updating BGP and router hardware represents a challenge – as we have experienced in securing the BGP protocol through BGPSEC [68], which has been an effort for over two decades. **Volatility of Carbon Emissions**: Renewable energy sources (e.g., solar and wind) can fluctuate greatly within short time spans. The dynamic nature of carbon data used in routing metrics would introduce a large number of re-routing events, likely pushing BGP beyond its scalability limits.

This chapter aims to overcome these challenges to enable carbon-aware inter-domain routing. Concretely, we pose two main research questions:

1. *How can we design a viable system that enables carbon-aware inter-domain routing?*
2. *How beneficial for endpoints can this system be in terms of the carbon footprint of their Internet usage?*

To answer the first question, we identify a promising opportunity in Path-Aware Networking (PAN). PAN architectures provide endpoints with path information and allow them to choose paths based on their individual criteria. Thus, endpoints can optimize paths for multiple criteria, thus addressing the inefficient green route problem. Furthermore, some PANs employ the packet-carried forwarding state where packets convey the forwarding path information in the header field, making routers stateless and addressing the ossification problem.

One way to achieve carbon-aware routing in PANs is to provide endpoints with paths' carbon information and let them perform fine-grained path optimization, weighing carbon intensity against other performance criteria [69]. Figure 4.1 illustrates carbon-aware inter-domain routing in a PAN architecture. To that end, a PAN architecture needs to be extended to provide carbon information in a practical way. More precisely, path carbon intensity needs to be estimated and disseminated in a way that (1) provides up-to-date information, (2) is scalable despite the volatility of carbon intensity. We discuss these requirements in more detail in Section 4.2.1,

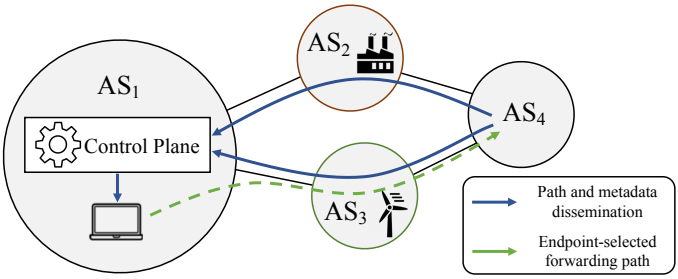


Figure 4.1: The concept of carbon-aware inter-domain routing in a PAN architecture.

and (3) does not require autonomous systems (AS[1]) to disclose their topology and electricity providers.

In this chapter, we design CIRo (**C**arbon-aware **I**nter-domain **Ro**uting), a system that forecasts and disseminates carbon intensity of inter-domain paths in a PAN architecture. It satisfies all the aforementioned requirements by *forecasting* day-ahead carbon intensity of inter-domain paths in a *distributed* manner and disseminating the data via the path distribution mechanism of PAN (cf. Section 4.2.2 and Section 4.4). To compute these forecasts, CIRo uses our new model for the carbon intensity of inter-domain paths (cf. Section 4.3).

To prove that scalable green routing can be deployed and used in the short term, we build CIRo based on the commercially deployed SCION PAN [10, 8] architecture, implement a proof of concept on the open-source SCION codebase [42]. We anticipate that a concrete open-source system provides an important baseline for future research to extend and improve upon.

To answer the second question, i.e., the carbon-footprint benefits for endpoints and end domains, we carry out an investigation by simulating CIRo on a large-scale realistic Internet topology. This topology reflects environmentally relevant characteristics of today's Internet and allows for the simulation of carbon-aware path selection. Our simulations show that CIRo can reduce the carbon intensity of inter-domain traffic by at least 47% between half of the domain pairs. Further we show that 87% of end domains can benefit from at least 50% reduction in the footprint of their Internet usage, i.e., the amount of carbon emission that can be attributed to them for their Internet use.

Importantly, the reduction in the carbon footprint of end domains only means that less carbon emission can be attributed to individual end domains, and it does

1 In this thesis, we use AS and domain interchangeably.

not necessarily translate to an actual reduction in carbon emission. The high-carbon footprint paths that are avoided still cause carbon emissions due to their idle power consumption. However, the shift of traffic of environmentally conscientious endpoints from polluting paths to greener ones is expected to eventually result in actual reductions in carbon emissions through competition among ISPs that is enabled by CIRo: to attract more traffic and increase revenue, ISPs enhance their carbon efficiency to entice environmentally conscientious endpoints to send traffic through their networks, which will, in turn, encourage other ISPs to do the same. This phenomenon is studied in detail in a companion publication [5].

Contributions This chapter makes the following contributions:

- Propose a carbon-intensity model for inter-domain paths;
- design CIRo, a system to forecast the carbon intensity of inter-domain paths and disseminate it to endpoints;
- implement a proof of concept for CIRo on SCION's open-source codebase;
- investigate the impact of carbon-aware inter-domain path selection on the carbon footprint of end domains by implementing CIRo in the ns-3-based SCION simulator and simulating it on a large-scale topology.

4.2 DESIGN PRINCIPLES

This section describes the design requirements and challenges of a system that estimates and disseminates carbon-intensity of inter-domain paths in a path-aware Internet and provides an overview of CIRo, which overcomes these challenges.

4.2.1 *Requirements and Challenges*

We identify two requirements for estimating and disseminating network paths' carbon intensity across the Internet.

R1: Reliable Carbon Information. The carbon intensity of network paths must be quantified in an accurate and reliable manner. This characterization is complex because the carbon intensity of a network path varies over time and depends on the location of all devices on the path, and their electricity mix, which in turn depends on the daytime and the weather conditions.

R2: Scalable Dissemination. Carbon-intensity information must be disseminated in a scalable way. This is challenging as paths' carbon intensity fluctuates and needs

to be disseminated frequently. However, frequent dissemination of information about numerous Internet paths can overwhelm participating ASes in terms of communication and computation costs.

R3: Not Requiring ASes to Disclose their Topology. The dissemination of carbon intensity information of ASes must not require them to disclose their internal topology or electricity providers. However, accurate computation of an inter-domain path's carbon intensity requires fine-grained information about all devices on the path across multiple ASes.

4.2.2 *System Overview*

We design CIRo such that it satisfies requirements R1–R3 above. We satisfy the reliability requirement (R1) by proposing a model for the carbon intensity of inter-domain paths and using forecasting. The scalability requirement (R2) is met by providing forecasts instead of real-time information to avoid frequent re-dissemination upon changes. Lastly, CIRo does not require ASes to disclose their topology or their electricity providers (R3) through a distributed design with instances in all participating ASes: each AS deploys a local instance of CIRo that forecasts and disseminates the carbon intensity of intra-domain paths without disclosing detailed information about the AS's topology. The carbon intensity of an *inter*-domain path is then calculated by accumulating the carbon intensities of all its constituent *intra*-domain segments, each traversing one AS (cf. Section 4.3).

Each CIRo instance consists of two modules: (1) The forecasting module, and (2) The information dissemination module. The interaction of these modules is illustrated in Figure 4.2.

The **forecasting module** computes day-ahead hourly forecasts (i.e., forecast for 24 hours) for the carbon intensity of intra-domain paths that connect interfaces of the AS. These forecasts are then inserted into the forecast database. The main building block of the forecasting module is the model we propose for the **C**arbon **I**ntensity of **D**ata **T**ransmission over inter-domain paths (*CIDT*, i.e., CO_2 mass emitted per unit of traffic being forwarded on a path in g/bit, cf. Section 4.3). This model takes into account the **E**nergy **I**ntensity of **D**ata **T**ransmission for network devices (*EIDT*, i.e., their energy consumption per bit of forwarded traffic in kWh/bit) and the carbon intensity of their used electricity (*CIE*, which can be retrieved from commercial tools such as ElectricityMaps [70], WattTime [71], or research projects such as DACF [72].

The **information dissemination module** disseminates the carbon intensity forecasts across the Internet. We design this module as an extension to the control-plane infrastructure of a PAN architecture. We extend the functionality of the control

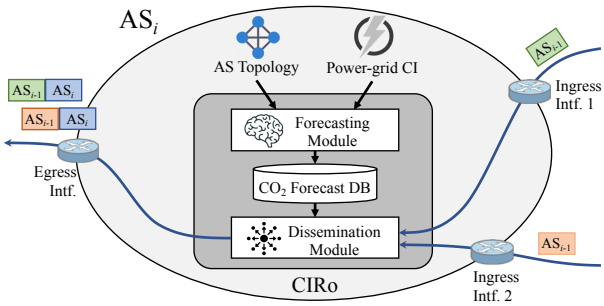


Figure 4.2: Overview of CIRo. A participating AS runs a CIRo instance. An instance consists of a forecasting module and an information dissemination module. The forecasting module computes path carbon intensity forecasts using a model we introduce, the topology of the AS, and power-grid carbon-intensity (CI) forecasts, and inserts its forecasts into the forecast database. The dissemination module disseminates forecasts to other ASes using routing messages. It also provides endpoints within an AS with these forecasts.

plane to encode carbon-intensity forecasts within routing messages. As a result, endpoints receive this information along with paths without requiring additional queries, reducing standardization efforts as well as communication and computation overheads.

4.3 MODELING CARBON INTENSITY OF INTER-DOMAIN PATHS

In this section, we develop a model enabling CIRo to compute and disseminate carbon-intensity forecasts of inter-domain paths in a distributed manner that does not require ASes to disclose their internal topology.

4.3.1 *Carbon Intensity of Data Transmission*

Data transmission over a network path may cause CO_2 emission through two levels of indirection: forwarding traffic causes (electrical) energy consumption on network devices, and the electricity used by each device is generated from energy resources in a process that may emit CO_2, depending on the energy resources.

We define **C**arbon **I**ntensity of **D**ata **T**ransmission over a network path, i.e., *CIDT* in g/bit, as the CO_2 emission that can be attributed to a unit of data for being

transmitted over that path. Since energy is an additive quantity, the CO_2 emission of energy consumption and the CO_2 emission of data transmission are additive.

4.3.2 *Inter-Domain CIDT*

Using CIDT's additive property, we write the CIDT of an inter-domain path (denoted by $\{AS^{src}, ..., AS^{dst}\}$) as the sum of CIDTs of consecutive AS hops it consists of:

$$CIDT_{\{AS^{src}, ..., AS^{dst}\}} = \sum_{AS^{i}_{[ing, eg]} \in \{AS^{src}, ..., AS^{dst}\}} CIDT_{AS^{i}_{[ing, eg]}}, \tag{4.1}$$

where $AS_{[ing, eg]}$ denotes an AS hop on the inter-domain path. *[ing, eg]* denotes the ingress and egress interface pair from/to which the inter-domain path enters/exits the AS hop.

The additive property is essential for distributed functionality of CIRo: each CIRo instance only needs to compute the CIDT of paths within one AS without disclosing topology information.

Equation (4.1) does not explicitly include inter-domain *links* connecting the border routers of neighboring ASes. That is because such links are either (1) direct links connecting two routers in the same data center, or (2) peerings via IXPs, IXP federations, IXP port resellers, or layer-2 links to an IXP. In the first case, the preceding AS on a path includes the CIDT of the link in its hop's CIDT. In the second case, previous research suggests IXPs and resellers constitute ASes in a PAN context [8]. Thus, they compute their CIDT as normal ASes in the same way we propose in this section.

4.3.3 *Per-Hop CIDT*

An AS may forward traffic on multiple internal paths, e.g., using equal-cost multi-path (ECMP). Therefore, we define the CIDT of $AS_{[ing, eg]}$ as the mean CIDT of all active *intra-domain* paths from its ingress interface to its egress interface. Thus,

$$CIDT_{AS_{[ing, eg]}} = \frac{1}{|\mathcal{P}_{[ing, eg]}|} \sum_{P_{[ing, eg]} \in \mathcal{P}_{[ing, eg]}} CIDT_{P_{[ing, eg]}} \tag{4.2}$$

where $P_{[ing, eg]}$ is an *intra-domain* path connecting *ing* and *eg* interfaces within the AS, and $\mathcal{P}_{[ing, eg]}$ is the set of all such paths. In case of weighted ECMP (WECMP), Equation (4.2) is substituted with a weighted mean over all paths.

4.3.4 *CIDT of a Single Intra-Domain Path*

The CIDT over a network path within an AS is composed of two components: (1) The marginal CIDT, which is the result of the actual amount of energy consumed to forward a bit of data over a path, and (2) The amortized CIDT, which is the result of energy consumed to keep network paths operational and is independent of traffic volume over the path. Therefore,

$$CIDT_{P_{[ing,\,eg]}} = CIDT^{marginal}_{P_{[ing,\,eg]}} + CIDT^{amortized}_{P_{[ing,\,eg]}} \tag{4.3}$$

Because of the additive property, we can write

$$CIDT_{P_{[ing,\,eg]}} = \sum_{D \in P_{[ing,\,eg]}} CIDT^{marginal}_{D} + CIDT^{amortized}_{D} \tag{4.4}$$

where D denotes a network device on $P_{[ing,\,eg]}$. Since inter-domain communication mostly takes place over backbone networks, we consider typical backbone-network devices, e.g., IP/MPLS core routers and optical wavelength-division multiplexed (WDM) devices [73, 74].

4.3.4.1 *Marginal CIDT*

For each device D, $CIDT^{marginal}_{D}$ is the result of (1) the marginal energy the device consumes to forward one bit of data (or marginal **E**nergy **I**ntensity of **D**ata **T**ransmission, $EIDT^{marginal}_{D}$ in kWh/bit), and (2) the marginal per-bit energy consumption of the device's external cooling and facilities, which is proportional to $EIDT^{marginal}_{D}$ [74]. Thus, the total marginal energy is also proportional to $EIDT^{marginal}_{D}$ by a constant factor ($\eta_{D,c}$) determined by the power-usage efficiency (PUE) [75, 74].

To compute $CIDT^{marginal}_{D}$, we multiply the total marginal energy by the CIE (in g/kWh) in the location of that device (CIE_D). This value can be retrieved from commercial tools such as ElectricityMaps [70], WattTime [71], or research projects such as DACF [72]. Therefore,

$$CIDT^{marginal}_{D} = \eta_{D,c} EIDT^{marginal}_{D} CIE_D \tag{4.5}$$

$EIDT^{marginal}_{D}$ can be computed using the following formula [76], irrespective of device type:

$$EIDT^{marginal}_{D} = \frac{1}{3600} \frac{P_{D,max} - P_{D,idle}}{C_{D,max}} \tag{4.6}$$

where $P_{D,max}$ is the maximum power consumption of the device (in kW), $P_{D,idle}$ is its idle power consumption (in kW), and $C_{D,max}$ is its maximum capacity (in bps). Note that 1 kWh/bit = 3600 kW/bps.

4.3.4.2 *Amortized CIDT*

Equivalently to the marginal carbon intensity, the amortized carbon intensity $CIDT_D^{amortized}$ of a device D is determined by the amortized idle energy consumption $EIDT_{D,idle}^{amortized}$, scaled by the PUE factor $\eta_{D,c}$. In addition, the model takes into account the set $\mathcal{R}(D)$ of *redundant* devices associated with device D, i.e., devices enabled when the primary device D becomes unavailable. We assume that these redundant devices do not forward any traffic. As a result, we can write:

$$CIDT_D^{amortized} = \eta_{D,c} EIDT_{D,idle}^{amortized} CIE_D + \sum_{D' \in \mathcal{R}(D)} \eta_{D',c} EIDT_{D',idle}^{amortized} CIE_{D'} \tag{4.7}$$

Since redundant devices can be located in a different location from the primary device, they could have different CIEs.

To amortize the idle energy consumption of a device over a bit of data crossing the device, we hold that bit of data accountable for the device's total *idle* energy consumption during its processing time. The capacity of the device (in bps) determines the processing time: On average, processing of a bit on a device takes time $\frac{1}{C_{D,max}}$ (in bit/bps = s). By multiplying the idle power consumption of the device by this duration, we arrive at the idle energy consumption of the device during this interval:

$$EIDT_{D,idle}^{amortized} = \frac{1}{3600} \frac{P_{D,idle}}{C_{D,max}} \tag{4.8}$$

In case of redundant devices, we substitute their $P_{D',idle}$ in Equation (4.8). However, since the *primary* device forwards the traffic, the processing time is still determined by the maximum capacity of the *primary* device. Thus,

$$EIDT_{D',idle}^{amortized} = \frac{1}{3600} \frac{P_{D',idle}}{C_{D,max}} \tag{4.9}$$

In Section 4.6.3, we discuss another method for amortizing idle energy consumption.

4.3.5 *Final Form of Per-Hop CIDT*

We combine Equations (4.2)–(4.9) and derive the following formula for the CIDT of a path within an AS between an interface pair:

$$CIDT_{AS_{[ing,\,eg]}} = \frac{1}{3600 \cdot |\mathcal{P}_{[ing,\,eg]}|} \sum_{P_{[ing,\,eg]} \in \mathcal{P}_{[ing,\,eg]}} \sum_{D \in P_{[ing,\,eg]}} \left(\eta_{D,c} \frac{P_{D,max}}{C_{D,max}} CIE_D + \sum_{D' \in \mathcal{R}(D)} \eta_{D',c} \frac{P_{D',idle}}{C_{D,max}} CIE_{D'} \right) \tag{4.10}$$

In Equation (4.10), the only time-variant variable is CIE, which depends on the availability of renewable electricity, which in turn depends on the weather conditions and the time of day. As a result, $CIDT_{AS_{[ing,\,eg]}}$ is also time-variant.

4.4 CIRO: DESIGN DETAILS

This section describes the design details of the forecasting and information dissemination modules, for which we implement a proof of concept on the SCION codebase [42], available on GitHub [77].

4.4.1 *Forecasting Module*

The forecasting module computes day-ahead hourly CIDT forecasts between interfaces of each AS. This module is distributed across participating ASes, where each instance is administered by one AS. Thus, it does not disclose internal information about the AS to any external entity.

4.4.1.1 *Required Inputs*

Each instance of the forecasting module performs its predictions using Equation (4.10), based on the following AS information.

Topology Information:

1. *AS Interfaces:* Set of AS interfaces and the corresponding border routers.
2. *Intra-Domain Paths:* Device-level intra-domain paths connecting border routers of the AS, computed by intra-domain routing protocols and listed in routers' intra-domain routing information base (RIB).

Device Information:

Table 4.1: Energy intensities of typical devices in IP and WDM layers [74].

Device type	Energy intensity (I_E in W/Gbps = J/Gbit)
Core router	10
WDM switch (OXC)	0.05
Trans/Mux -ponder	1.5
Amplifier	0.03
Regenerator	3

1. *Device Locations:* Locations of all network devices (both primary and redundant) of the AS, including routers and optical devices. If an AS leases optical fibers from other companies, the location and redundancy provisioning of optical devices might not be available. In that case, the module uses models [74] of IP over WDM networks to approximate locations. Moreover, the same model suggests one redundant device identical to and in the same place as the primary device.

2. *Device Specifications:* Maximum power consumption (P_{max}), maximum capacity (C_{max}), and idle power consumption (P_{idle}) of all network devices in the AS network, available in device specifications. If this information is unavailable for device D (primary or redundant), the module substitutes $\frac{P_{D,max}}{C_{D,max}}$ in Equation (4.10) with typical values for the energy intensity of network devices [74] (cf. Table 4.1).

Electricity Information:

1. *Power-Usage Efficiency (PUE):* Power-usage efficiency at all AS's points of presence (PoPs), determining $\eta_{D,c}$. If unknown, the module uses a typical value of 2 [74].

2. *Carbon Intensity of Electricity (CIE):* Day-ahead hourly forecast of CIE at all PoPs from electricity carbon-intensity forecasting sources such as ElectricityMaps [70], WattTime [71], or research projects such as DACF [72].

4.4.1.2 *Forecasting Procedure*

A periodic procedure retrieves updated inputs and re-computes day-ahead hourly CIDT forecasts for all interface pairs. The periodic re-computation is necessary as (1) CIE forecasts are updated with a period of one hour, and (2) paths between interfaces could change even within these 1-hour intervals. Thus, we consider two re-computation periods: T_{CIE}, which is set to 1 hour to retrieve updated CIE forecasts at the beginning of every hour (**:00), and T_{path} to collect updated intra-domain RIBs from routers. T_{path}'s value depends on the frequency of path changes in one AS and is set by each AS independently. T_{path} is only relevant if it is less than T_{CIE}.

At each T_{CIE} (i.e., beginning of every hour), the module first retrieves all inputs mentioned in Section 4.4.1.1. Then, it computes the day-ahead hourly CIDT forecast for every pair of interfaces in both directions (i.e., both could be ingress or egress interfaces) using Equation (4.10). The result for each interface pair in each direction is a vector of 24 CIDT values associated with the next 24 hours. Finally, the module inserts the results into the *CIDT database* shared with the information dissemination module.

At each T_{path}, the module checks whether intra-domain paths between every interface pair have changed compared to the last T_{path}. If so, the module computes CIDT forecasts for the new path and updates the database accordingly.

The database stores two records for each interface pair, one per direction. Each record has 26 columns; the first two columns store the ingress and the egress interface, respectively, and the following 24 columns store CIDT forecasts for the next 24 hours associated with that pair in the direction from the ingress interface to the egress interface.

4.4.2 *Information Dissemination Module*

This module uses the routing infrastructure (control plane) of a PAN architecture to disseminate CIDT forecasts. It disseminates forecasts across ASes along with path construction by the means of a routing-message extension we introduce. Endpoints can then receive this information when they request paths from the path service of their ASes.

The benefits of building this module based on already existing infrastructure are two-fold: (1) Low message complexity and communication overhead as it does not send additional messages dedicated to CIDT, and (2) Low design and implementation effort by re-using existing protocols.

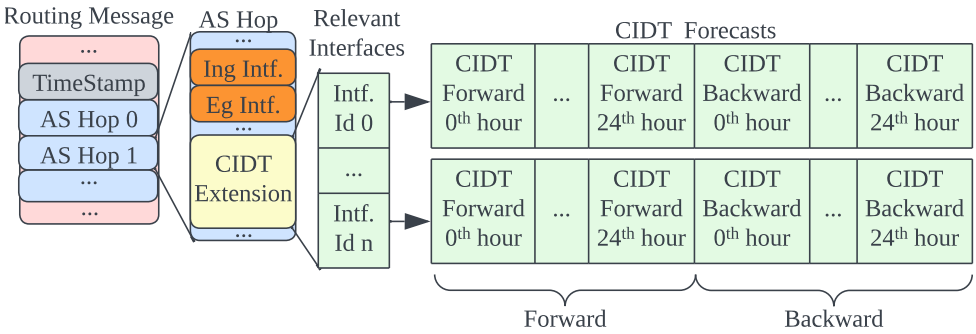


Figure 4.3: A SCION PCB containing `CIDTExtension`.

We develop a concrete prototype of our design based on the control plane of the SCION PAN architecture. Using SCION's control plane further improves this module in terms of (1) scalability to large topologies due to SCION's hierarchical routing architecture [8], and (2) providing up-to-date carbon information due to the periodic routing in SCION. However, this hierarchical routing only gives CIDT forecasts for path segments, not the entire path.

We now introduce `CIDTExtension`, an extension to routing messages to disseminate CIDT forecasts. Later, we explain how ASes use this extension to convey the CIDT forecasts of their paths during the routing process.

4.4.2.1 *CIDTExtension*

Figure 4.3 demonstrates the structure of this `CIDTExtension` within a SCION routing message known as a path-construction beacon (PCB). Every AS originating or propagating a PCB can add one `CIDTExtension` in its `ASEntry` to disseminate its CIDT forecasts. A `CIDTExtension` is a map from the relevant interface identifiers (`IntfId`) of the AS to vectors of CIDT forecasts. Having more than one relevant interface is useful for the composition of the SCION path segments. An `IntfId`, as a key of this map, represents the intra-domain segment from the `IntfId` to the egress interface (`EgIntf`) of the `ASEntry`.

The CIDT forecast vector associated with an `IntfId` provides CIDT forecasts for the forward path from `IntfId` to `EgIntf` ($CIDT_{AS_{[ing,\,eg]}}$) and for the backward path from `EgIntf` to `IntfId` ($CIDT_{AS_{[eg,\,ing]}}$).

A CIDT forecast vector is a sequence of 48 contiguous unsigned 8-bit integer numbers, each specifying one CIDT value in mg/Gbit. We choose this unit based on

our observation of the CIDT range in Internet paths (cf. Section 4.5.2). The first 24 values of the vector represent $CIDT_{AS_{[ing,\,eg]}}$ forecasts for the 24-hour interval starting from the full hour time point before the PCB's `TimeStamp`. The 24 following values provide similar information for $CIDT_{AS_{[eg,\,ing]}}$. This design ensures time-alignment of forecasts added by different ASes at different times.

4.4.2.2 *Timing of CIDT Dissemination*

Since CIDT forecasts have a 24-hour time window, they need to be disseminated regularly to preserve the temporal continuity of CIDT information.

This periodic dissemination, however, does *not* require equal frequency or synchronization among ASes as all `CIDTExtension`s in one routing message are time-aligned.

Therefore, every AS locally chooses its CIDT dissemination period, with a lower bound of 1 hour and an upper bound of 2.6 hours. The lower bound is derived from the update period of CIDT forecasts by the forecasting module, i.e, 1 hour, making it unnecessary to disseminate CIDT forecasts more frequently. The upper bound is required to guarantee the temporal continuity of CIDT forecasts. It is derived for the worst case scenario, where all ASes on a path segment synchronously disseminate routing messages with the same frequency. In that case, the temporal continuity is satisfied when the last AS on the path segment receives routing messages with CIDT forecasts that are valid for at least one period. Therefore, the upper bound is $\frac{24}{n-1}$, where n is the number of AS hops on the path segment. Assuming the longest path segment is 10 AS hops long (twice as long as the average AS-hop length in today's Internet [78]), the upper bound period is 2.6.

4.4.2.3 *Process of CIDT Dissemination*

For each PCB to disseminate each egress interface (`EgIntf`), the dissemination module constructs the corresponding `CIDTExtension`, adds it to the `AS Hop`, extends the PCB with the `AS Hop` and disseminates it on the egress interface to the next-hop AS.

To construct the `CIDTExtension` for each routing message, the dissemination module first queries the *CIDT database* for CIDT forecasts between the `EgIntf` and all `IntfId`s in relevant interfaces. The retrieved results may not be time-aligned with the PCB's `TimeStamp`. This is because the database always provides CIDT forecasts for the next 24 hours from the current time, but the `CIDTExtension` has to provide CIDT forecasts for the next 24 hours after the routing message's `TimeStamp`,

which is in the past. To solve this problem, it shifts the vectors retrieved from the database to the right by the number of hours past from the `TimeStamp`'s hour.

CIDT Dissemination in Core Beaconing The dissemination module uses core beaconing to disseminate CIDT forecasts of core-path segments. For each core PCB that is being disseminated by a core AS, the beacon service constructs a `CIDTExtension` containing the CIDT forecasts only for the intra-domain segment between the ingress and egress interfaces of the PCB.

Including forecasts for only one interface is essential to the scalability of dissemination module among core ASes. This is because core ASes are large ASes with numerous interfaces, thus, including information about all interfaces of every hop on the path would introduce significant overhead to each PCB. This per-PCB overhead in conjunction with the high message complexity of core beaconing [8] would hinder the scalability of the dissemination module. Including the forecasts associated with only one interface does not affect the expressiveness of CIDT forecasts. This is because each core PCB specifies only one core-path segment and core-path segments do not combine with each other.

CIDT Dissemination in Intra-ISD Beaconing The dissemination module uses intra-ISD beaconing to disseminate CIDT forecasts of up- and down-path segments. Therefore, beacon services include the CIDT forecasts between ingress and egress interfaces of disseminated PCBs in `CIDTExtensions`. However, since an end-to-end path can be a combination of path segments and peering links, it is essential to provide CIDT forecast of their combination as well to arrive at the end-to-end CIDT forecast. To cover the combination with core-path segments, a core AS originating a non-core PCB includes the CIDT forecasts between the `EgIntf` of the PCB and all interfaces connecting to all core neighboring ASes. To cover the combination of up- and down-path segments, any AS disseminating a non-core PCB includes the CIDT forecasts between the `EgIntf` of the PCB and all interfaces connecting to all other customer ASes whose `IntfId` is smaller than `EgIntf`. In this way, the required information is either encoded in the up segment or in the down segment, halving the overhead. To cover the combination of up- or down-path segments with peering links, any AS disseminating a non-core PCB includes the CIDT forecasts between the `EgIntf` of the PCB and all interfaces connecting to all other peering ASes.

4.4.2.4 *Obtaining the CIDT of Full Inter-Domain Paths*

Once an endpoint retrieves path segments to construct an inter-domain path, it computes the CIDT of the full path for the current hour using `CIDTExtensions` provided by all segments. Note that different path segments have different origination time, thus, their CIDT forecast vectors are not necessarily aligned. Therefore, endpoints compute the index of current time for each path segment independently. Then, they accumulate the CIDT forecast of all AS hops associated with the current hour to arrive at the CIDT of the end-to-end path.

4.5 CIRO'S IMPACT ON CARBON FOOTPRINT

This section answers the second research question of the chapter: To what extent could carbon-aware inter-domain path selection reduce the carbon footprint of the Internet usage of endpoints and end domains (end ASes, i.e., source or destination of the traffic)? This potential is critical to the attractiveness of CIRo, and in turn to the prospect, span, and pace of CIRo's deployment. We describe our method in Section 4.5.1, and we present our results in Section 4.5.2.

4.5.1 *Methodology*

Endpoints could benefit from carbon-aware path selection by sending their traffic over paths with lower CIDT (compared to the BGP path in the traditional Internet). This benefit can be found by computing the CIDT difference between CIRo and BGP paths and multiplying this difference by the traffic volume between domain pairs.

Since SCIONLab is an overlay network with a modest number of virtual ASes, it does not resemble real Internet paths. Therefore, we cannot rely on our proof-of-concept implementation on SCIONLab to approximate real CIDT of paths in the Internet. Thus, we develop a method based on simulations and large-scale realistic datasets. Our method consists of the following steps:

1. Approximation of the Internet topology (Section 4.5.1.1),
2. reconstruction of intra-domain paths (Section 4.5.1.2),
3. carbon-intensity estimation for intra-domain paths (Section 4.5.1.3),
4. reconstruction of BGP paths for comparison (Section 4.5.1.4),

5. discovery of CIRo paths (Section 4.5.1.5), and
6. synthesis of a traffic matrix (Section 4.5.1.6).

4.5.1.1 *Creating an Inter-Domain Topology*

To find realistic inter-domain paths, we use the inter-domain topology suggested by Krähenbühl et al. [8], which contains a total of 2000 ASes. These are the highest-degree Tier-1 and Tier-2 ASes of the CAIDA AS-Rel-Geo dataset [53]. We extract this topology from the AS-Rel-Geo dataset by iteratively removing the lowest-degree ASes. The benefits of CIRo are most likely realized within these top-tier ASes, as the main differences in the CIDT of paths stem from path diversity provided by massively interconnected ASes in the Internet core.

4.5.1.2 *Finding Intra-Domain Paths*

To compute CIDT of inter-domain paths, we need the CIDT of intra-domain segments they consist of. In turn, the CIDT of intra-domain segments relies on finding intra-domain paths. Since intra-domain paths are not publicly available, we compute them by applying Dijkstra's algorithm [79] on the internal topology of each AS.

First, we locate border routers associated with AS interfaces, where the interfaces are available from the AS-Rel-Geo dataset [53]: For each interface between AS A_1 and A_2, we identify all routers of AS A_1 that are also connected to AS A_2, and select the router closest to the interface. This information is available via the CAIDA ITDK dataset [80]. If no routers satisfy these conditions, we add a router at the interface location. Given these border routers, we compute the shortest router-level path between every pair of border routers by running the Dijkstra algorithm [79] on the intra-domain topology of each AS, given by the ITDK dataset. If we cannot find such a path, we assume that the number of routers is an increasing integer-valued step function of the distance between interfaces with the range of $\{1, 2, 3, 4, 5\}$ and steps at 1 km, 20 km, 100 km, and 1000 km distance.

Once router-level paths are established, we add optical devices between consecutive routers. As each packet crosses every WDM switch, transponder, and muxponder once before entering a router and once after leaving it, we assume two of these devices per router on the path [74]. The number of amplifiers and regenerators, however, depends on the distance between consecutive routers. We assume one amplifier at each 80 km interval, and one regenerator at each 1500 km interval [74].

4.5.1.3 *Estimating CIDT of Intra-Domain Paths*

The CIDT of intra-domain paths is necessary to compute the CIDT of inter-domain paths. Thus, we compute $CIDT_{AS_{[ing,\,eg]}}$ for all ingress and egress interface pairs of all ASes in the topology using Equation (4.10) and computed paths in Section 4.5.1.2.

However, not all the required information by Equation (4.10) is available. First, we do not know using which simultaneous internal paths ASes achieve multi-path routing. Second, we do not know device specifications, the number and location of redundant devices per primary device, and the PUE at PoPs of each AS. Third, hourly *CIE* forecasts are not available for all device locations.

To address the first issue, we assume single-path intra-domain routing in simulations, as opposed to the real deployment of CIRo. To solve the second problem, we use the model Heddeghem et al. [74] propose. They suggest typical values for the energy intensity of various types of backbone network devices (cf. Table 4.1). They also propose the typical value of 2 for PUE at each PoP. Further, for each primary device, they consider one identical redundant device. We further assume that both devices are at the same location. To overcome the third challenge, we use each country's *annual average CIE* as the CIE_D of all devices located in that country. We compute the *CIE* of countries using their mix of electricity-production technology [81], and the carbon intensity of electricity-production technologies [82] shown in Table 4.2. To find the location of routers, we use the CAIDA ITDK dataset [80].

Table 4.2: 50^{th} percentile CO_2 emission of different energy resources. From Edenhofer's IPCC report [82].

Energy resource	Carbon intensity (gCO_2/kWh)
Coal	1001
Natural gas	469
Biomass	230
Solar	46
Geothermal	45
Nuclear	16
Wind	12
Hydroelectric	4

4.5.1.4 *Finding BGP Paths and their CIDT*

As not all ASes publish their routing tables, we simulate BGP using the SimBGP simulator [83] to find the BGP-selected paths. Then, we compute their CIDT as the sum of CIDTs of all their intra-domain segments.

4.5.1.5 *Finding Green Inter-Domain Paths*

To compute green inter-domain paths in the mentioned topology, we implement CIRo in the ns-3-based [26] SCION simulator [8]. Furthermore, we change the beaconing (routing) algorithm such that all ASes optimize the CIDT of paths, i.e., they advertise paths with the lowest CIDT. However, as this algorithm is heuristic, it cannot guarantee optimality.

We use simulations for practicality as setting up a large-scale testbed is not feasible in terms of cost and effort. Further, Krahenbühl et al. [8] have shown that the SCION simulator is highly predictive of actual testbed results in terms of discoverd paths, so simulation does not affect the reliability of results.

4.5.1.6 *Synthesizing a Traffic Matrix*

We compute the global inter-domain traffic matrix for all ASes in the CAIDA AS-Rel dataset [39] using the model provided by Mikians et al. [84]. In this model, the amount of traffic from AS_i to AS_j is the sum of traffic sent from all users and servers in AS_i to all servers and users in AS_j for all application types:

$$T_{i,j} = \sum_{A \in applications} m_A \left(S_i p_i^A(j) + d_A S_j p_j^A(i) \right), \tag{4.11}$$

where $p_i^A(j)$ is the relative popularity of servers in AS_j for users in AS_i with respect to application A, S_i represents the size of an AS in the number of its IP addresses, d_A denotes the asymmetry factor for application A, which is the ratio of the response flow size to the request flow size for application A, and m_A scales the computed relative traffic for application A to its absolute real traffic volume.

We set $\log_{10} d_{http} = 1$ which is in the range of $(0.4, 1.5)$ for HTTP(S) data and media [84]. Furthermore, we assume the size of each AS to be its number of IPv4 addresses from the CAIDA Pfx2AS dataset [85], except Tier-1 ASes with no users and large CDNs with no users requesting a service. Moreover, $p_i^{http}(j)$ follows the Zipf distribution [86] with a slope of 1.2 [84]. For each source AS, we thus generate a vector of relative popularities from the Zipf distribution. Then, we assign the largest popularity values to the most popular destination ASes, and the remaining

popularity values randomly to other ASes. We define the most popular ASes as the ASes whose hosted websites have the largest accumulated inverse ranks in the Tranco1M dataset [87]. We find the website to AS mappings using DNS queries, and the Pfx2AS dataset. Once the matrix is computed, we scale it to the global HTTP(S) traffic, which is estimated to be 82 Exabytes per month [89, 88].

In addition to HTTP(s) traffic, we also include video traffic in the traffic matrix. For this video traffic, we consider Netflix, YouTube, and Amazon Prime Video, which are responsible for 15, 11.4, and 3.7 percent of total Internet traffic, respectively [89]. We construct the video traffic matrix by assuming that the amount of traffic any other AS receives from these services is proportional to its number of users in Pfx2AS dataset, and these services receive negligible traffic from other ASes. Finally, we add the video traffic matrix to the HTTP(S) traffic matrix.

To construct the traffic matrix for the core topology of 2000 ASes used in our simulation, we assume that the amount of traffic between two core ASes is the sum of all traffic in the global traffic matrix between their customer cone ASes in the AS-Rel dataset. If an AS is in the customer cones of multiple core ASes, its inbound and outbound traffic is evenly divided across its core AS providers.

4.5.2 *Results*

This section presents the result of our study on the effect of carbon-aware inter-domain path selection on the carbon footprint of end domains. We first compare the CIDT of greenest available paths with BGP paths. Then, we present the absolute and relative reductions in CIDT of communication between each pair of ASes. Finally, we demonstrate the reduction in the carbon footprint of end ASes.

Note that since the granularity of the traffic matrix is end ASes, not endpoints, we can only study the impact on CIDT (i.e., the intensity of carbon footprint) of end domains.

4.5.2.1 *Path CIDT Comparison*

Figure 4.4 shows the CIDT distribution of different types of paths available between AS pairs. The results indicate that CIRo-paths almost halve CIDT for almost all percentiles of AS pairs (and their endpoints). It also suggests that CIRo can find *multiple* paths with significantly lower CIDTs than BGP paths.

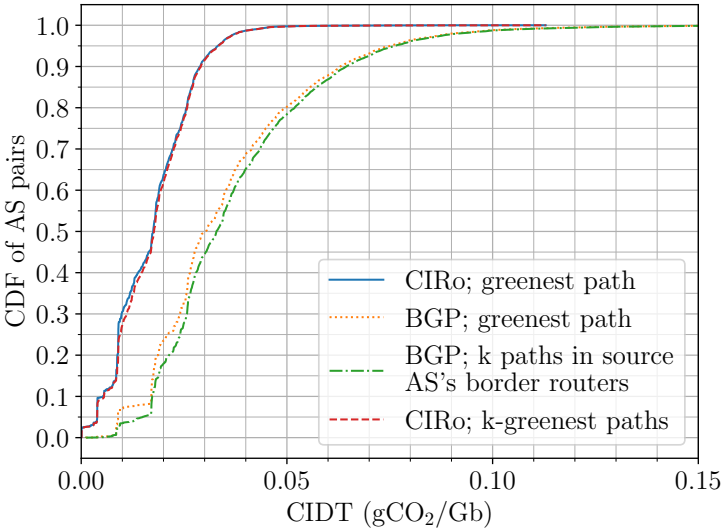
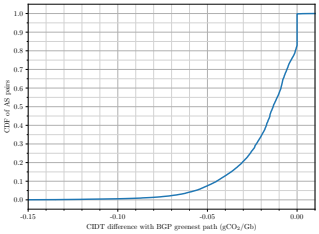
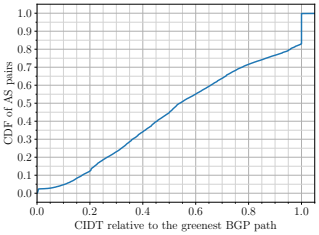


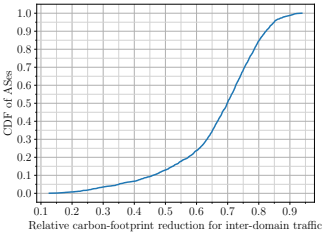
Figure 4.4: Distribution of path CIDT across AS pairs, distinguished by path types: (1) The greenest path discovered by CIRo, (2) The greenest BGP path among the k BGP paths in RIBs of the source AS's border routers, k being the number of border routers, (3) The average of available BGP paths, and (4) The average of the k-greenest paths discovered by CIRo.



(a) Absolute CIDT difference between the greenest CIRo path and the greenest BGP path, for all AS pairs.



(b) CIDT of the greenest CIRo path relative to the greenest BGP path, for all AS pairs.



(c) Relative annual reduction in the carbon-footprint of outbound inter-domain traffic, for all ASes.

Figure 4.5: Greenness comparison between carbon-aware path selection, and BGP.

Table 4.3: Carbon-footprint reduction for the outbound *inter-domain traffic* (HTTP(S) and video streaming) of the top-8 beneficiaries of carbon-aware path selection. This results are with the assumption that content providers do not have caches in local ISPs, and all their traffic to other ASes traverses inter-domain paths.

Source AS (ASN)	Carbon footprint reduction (t/yr)	Relative carbon footprint reduction for outbound inter -domain traffic (%)	Traffic (EB/yr)
Netflix (2906)	47640	68	423
YouTube (36040)	43032	55	326
Amazon (16509)	30240	65	230
Cloudflare (13335)	28728	79	147
Google (15169)	7536	69	57
Fastly (54113)	5520	57	55
Microsoft (8075)	4284	74	29
Incapsula (19551)	3876	76	22

4.5.2.2 *Absolute CIDT Reduction*

Figure 4.5a demonstrates the distribution of CIDT difference between the greenest CIRo path and the greenest BGP path, across all AS pairs. It suggests that 83% of AS pairs (and their endpoints) can benefit from CIDT reduction by using CIRo's greenest path. This reduction is at least 0.015 g/Gbit for half of AS pairs. Only a negligible portion of AS pairs experience increased CIDT as the used heuristic algorithm does not guarantee optimality.

4.5.2.3 *Relative CIDT*

Figure 4.5b illustrates the distribution of CIDT of the greenest CIRo path relative to the greenest BGP path, across all AS pairs. According to this figure, half of AS pairs (and their endpoints) can use paths that are at least 47% greener.

4.5.2.4 *Reduction in Carbon Footprint of End Domains*

Figure 4.5c depicts the distribution of ASes' relative reduction in annual carbon footprint of their outbound inter-domain traffic enabled by CIRo. According to this figure, this reduction is at least 50% for more than 85% of ASes.

Table 4.3 demonstrates absolute and relative annual carbon-footprint reductions of outbound inter-domain traffic through carbon-aware path selection for its top-8 beneficiaries. It also suggests that the most popular ASes, i.e., the largest CDNs, can benefit from the largest *absolute* carbon-footprint reduction.

4.5.3 *Limitations*

Our results provide an *upper* bound on the carbon-footprint reduction for endpoints and end-domains due to two assumptions:

No local caching. In our inter-domain traffic-matrix synthesis, we assume all the traffic between any pair of ASes traverses inter-domain paths, and thus, is not cached by local ISPs.

Device energy consumption. For the energy consumption of network devices, we make an effort to use data sources that are both peer-reviewed and comprehensive, i.e., report energy intensity for a wide range of network devices, and not just a small sample of devices. Hence, we rely on the most recent article that satisfies both conditions [74], with the caveat that it does not reflect the growing energy efficiency of modern hardware.

4.5.4 *We Report Reduction in Carbon Footprint, Not Emission*

As mentioned in Section 4.1, the *carbon footprint* of Internet usage of an endpoint (or an end domain) is the amount of carbon emission that can be attributed to the Internet data transmission of the endpoint.

Our system provides end domains with the opportunity to reduce their carbon footprint, i.e., emission they are responsible for, by sending their traffic on greener paths, which is beneficial for carbon accounting of end domains. However, these reductions in carbon footprint *do not directly translate into global carbon-emission reduction*: Even after shifting traffic to greener paths, the network devices on more polluting paths consume energy, and cause carbon emissions. Since idle energy consumption dominates energy consumption in network devices, the traffic shift to

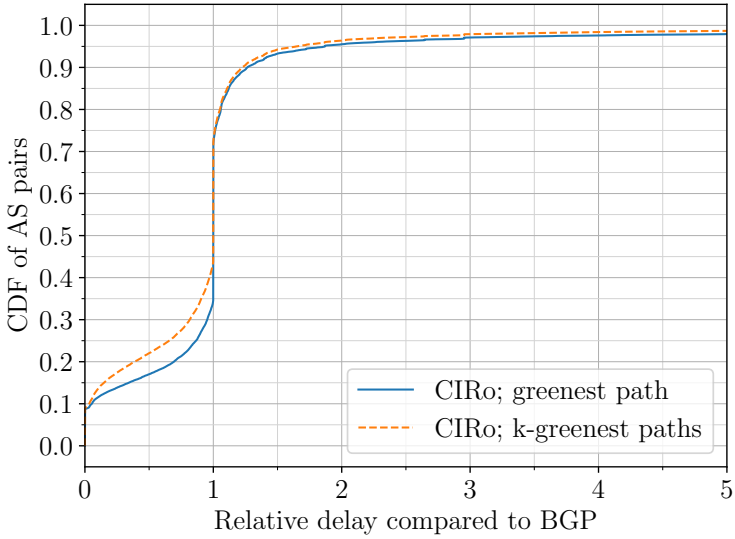


Figure 4.6: AS-pair distribution regarding the relative propagation delay compared to BGP for (1) the greenest path discovered by CIRo, and (2) the average of the k-greenest paths discovered by CIRo, k being the number of the source AS's border routers.

greener paths does not significantly reduce global carbon emission – the reduction in global carbon emissions would be affected through competition dynamics among ISPs to counteract the traffic shifts [5].

4.6 DISCUSSION

This section discusses several important aspects of carbon-aware inter-domain routing and outlines future research directions related to carbon-aware routing.

4.6.1 *Effects on Transmission Quality*

Latency We evaluate the impact of optimizing CIDT of paths on their propagation delay as an indicator of latency. We compute the propagation delay of an inter-domain path as the sum of the propagation delays of all AS hops, which we approximate using the great circle delay between ASes' border routers available in

the CAIDA dataset. This is a lower bound for the actual latency, which on average underestimates by a factor of ∼1.5 [90].

Figure 4.6 illustrates the distribution of relative propagation delay compared to BGP for paths discovered by CIRo, across AS pairs. According to this figure, optimizing paths only for CIDT does not increase the delay for 75% of AS pairs, and even reduces it for 35% of AS pairs. However, around 5% of AS pairs can suffer from significant inflated delay by a factor of more than 2 due to *inefficient green routes* resulting from optimizing paths exclusively for CIDT. Nevertheless, endpoint path selection in PANs provides an opportunity for multi-criteria path optimization to address this issue. If endpoints specify the AS-level path in every packet and border routers can thus be stateless (as in SCION [10, 8]), this multi-criteria optimization can also be scalable. Developing PAN-based multi-criteria path-optimization algorithms is an interesting task for future work.

Congestion By enabling endpoints to select network paths based on carbon intensity, many endpoints might select green paths, raising concern regarding congestion on such paths. However, due to the vast diversity of paths in the Internet, every AS pair is connected by numerous low-emission paths, with almost the same carbon intensity as the greenest possible paths. Figure 4.4 confirms that multiple green paths with carbon intensities as low as the greenest path exist between all AS pairs. Thus, endpoints have multiple choices to reduce the carbon footprint of their inter-domain communication, lowering the probability of congestion on one single green path.

Traffic Oscillation Oscillation arises in load-adaptive path selection, where delayed information about path load entices endpoints to switch paths. This shift causes a high load on the newly selected path and subsequent abandonment of that path [91, 19]. Therefore, such oscillation arises if the path attribute used as a selection criterion changes in response to the selection of the path. In contrast to path load, path carbon intensity is generally no such responsive criterion and therefore is unlikely to cause oscillation. However, we note that path selection based on carbon intensity may be combined with load-adaptive path selection to avoid congestion on green paths. The load-adaptive part of this multi-criteria strategy then needs to follow the guidelines for oscillation-free path selection developed in previous research [91, 92]. We expect congestion risk on green paths to decrease continuously as competitive pressure erodes greenness differences between paths.

4.6.2 *Operational Overhead and Carbon Footprint*

Overhead of CIRo An important concern is the energy consumption and thus the carbon footprint of running CIRo itself resulting from its computation and communication overhead. In the following paragraphs, we provide a detailed analysis of CIRo overhead, showing that a single mid-size server is more than sufficient to run a whole CIRo instance for a large AS with a global operation. Thus, CIRo introduces negligible overhead.

For each instance of CIRo, the overhead of the forecasting module includes *hourly* querying *CIE* forecasts, collecting AS internal topology and routes, computing CIDT for all interface pairs using Equation (4.10), and inserting them into the forecast database. The overhead of the dissemination module includes querying the forecast database, and encoding forecast information into routing messages, increasing the message size.

The overhead of the forecasting module only depends on the size of the AS it is located in. For a large AS with global operation with $O(10^4)$ interfaces, the per-period (hourly) forecasting overhead translates to $O(10^3)$ *CIE* queries (on the order of number of geographical zones in ElectricityMaps), writing $O(10^8)$ rows in a database table, and $O(10^{10})$ addition and multiplication operations. An efficient implementation using modern databases can carry out this procedure in $O(10^2)$ seconds on a single machine with a few CPU cores with $O(10^1)$ GB of memory. We do not consider collecting internal topology and routes as an overhead of CIRo, since ASes already continuously monitor and collect information about their network, irrespective of CIRo's presence.

The dissemination module's database queries per period (at least once in 2.6 hours. cf. Section 4.4.2) are upper bounded by the square of the number of interfaces in an AS; thus, a large AS with $O(10^4)$ interfaces requires $O(10^8)$ database queries.

The communication overhead of the dissemination module depends on the number of interfaces of the AS, the number of ASes in the inter-domain topology, and the number of paths per origin AS that the AS advertises to its neighbors. Our SCIONLab experiments with 100 ASes suggest a maximum of 125 kB overhead per period per interface. By extrapolating to $O(10^4)$ ASes in the network, we arrive at a maximum of 12.5 MB overhead per period per interface. Each inter-domain link needs to transfer this additional data in a period, which is a negligible effort considering the period length and the typical capacity of inter-domain links.

Overhead of PAN Architectures Since the stateless routers in a PAN architecture are significantly more energy efficient than routers in today's Internet [93], using a PAN architecture can further reduce the carbon footprint of the Internet.

4.6.3 *Amortizing Idle Carbon Emission*

In this thesis, we take each bit of data responsible for the idle energy consumption of fully utilized devices by amortizing their idle power consumption over their capacity, irrespective of traffic load.

Another approach is to amortize idle power based on the traffic load on a device. In this way, a bit of data is responsible for more carbon emission if the device is under-utilized and vice versa. This can cause a positive feedback on greener paths: with more traffic they attract, because of their initial greenness, their carbon intensity reduces, making them more attractive. However, this method complicates carbon intensity prediction because (1) it requires predicting traffic on a device level over the next 24 hours, and (2) it creates hard-to-predict dynamics through the positive feedback (i.e., paths becoming greener with more traffic).

4.6.4 *Deployment and Ossification*

The deployment of CIRo in a PAN architecture with stateless routers and a control plane separate from the data plane is relatively easy, as it introduces no change to the data plane and only modest, backward-compatible extensions of the control plane. Thus, no hardware or protocol update is required. While the deployment of a PAN Internet itself requires significant effort, the SCION Internet has in fact been experiencing growing deployment [8] in recent years as a result of its incremental deployability.

4.6.5 *Economic Effects of Green Routing*

From an economic perspective, it is central to predict the impact of carbon routing on the business performance of ISPs and on the competitive environment in the Internet, considering that ISPs might reduce their carbon footprint if they can thereby attract traffic. These competitive dynamics may result in a virtuous cycle, where ISPs compete for traffic shares by offering paths with ever-decreasing carbon intensity. In fact, this effect is confirmed by recent work based on game theory and economic modeling [5].

4.7 RELATED WORK

Much work has studied how to reduce the carbon footprint of the ICT sector by either improving the energy efficiency of end devices and networks [102, 99, 101, 97, 67, 96, 95, 98, 100, 94], or increasing the utilization of available renewable energy resources [105, 106, 103, 104].

4.7.1 *Green data centers*

A large body of research proposes ecological improvements in the area of data centers [107]: energy-efficient software [97], capacity planning for better resource utilization [98], improved energy management [99], better virtualization technologies [100], power-saving cooling systems [101], greenness-oriented load balancing across data centers [108], or simply using more renewable energy [106]. In a recent work, Radovanović et al. [109] propose a method to minimize the carbon footprint of computing among globally distributed data centers by temporally delaying flexible workloads. They achieve this goal by predicting the day-ahead carbon intensity and compution demands.

4.7.2 *Carbon-/energy-aware networking*

Research conducted on green routing and traffic engineering falls into two main categories: they either make the network more energy-efficient or route packets through paths whose energy resources are green.

In a visionary paper, Gupta and Singh point out the energy utilization of Internet routers, and suggest energy savings by placing devices in sleep mode [110]. Later, Zhang et al. [94] propose a heuristic to reduce energy by turning off line cards and rerouting traffic to underutilized links. Vasic et al. [111] perform an energy optimization on real-world networks by inactivating unnecessary network elements. Vasic and Kostic [112] propose Energy-Aware Traffic Engineering (EATe), to allocate traffic in an energy-optimal manner within an ISP. Andrews et al. [113] study network optimization from a theoretical perspective, with energy minimization as an objective. Chabarek et al. [114] study the power consumption of core and edge routers, optimizing the intra-domain energy consumption.

Among the studies also taking energy-efficient routing into account, Van der Veldt et al. [104] propose a path provisioning method to reduce the CO_2 emission of communications in a national research and education network (NREN). Aksanli et al. [102] design green energy-aware routing policies for wide area traffic between

data centers. Ricciardi et al. [103] use integer linear programming for routing and wavelength assignment considering a network node's carbon intensity. Gattulli et al. [105] propose a CO_2 and energy-aware routing mechanism for intra-domain routing. They find two paths for each source and destination pair; one with routers whose energy resources have the lowest emission, and one with the lowest energy consumption. Then, they compare these paths and select the one with the lower emission.

4.7.3 *Green inter-domain communication*

To the best of our knowledge, only two previous studies propose methods to reduce the carbon footprint of inter-domain communications. Shi et al. [95] extend the aforementioned traffic aggregation method [94] by taking into account the traffic between border routers of an AS. Thus, it does not tackle global carbon-aware routing. Nafarieh et al. [67] propose extensions for OSPF-TE and BGP to propagate information about the emission of links on each path to routers inside and outside an AS. Using this information, each router selects the path with the minimum path emission to all other routers in its own AS, and each border router shares this information with the neighboring AS. During BGP route propagation, the emission of paths is accumulated in update messages, and used by multi-homed ASes to select the provider of the greenest route. However, this BGP-based approach provides limited transparency and control over the carbon footprint of inter-domain communications compared to CIRo.

4.7.4 *Internet Energy models*

Numerous studies estimate the energy consumption and energy intensity of the Internet, where the methodologies can be categorized in bottom-up, top-down, and model-based approaches.

Bottom-up approaches Bottom-up approaches generalize the energy-intensity values obtained through direct measurement of selected network devices. The study conducted by Coroama et al. falls into this category [115]. In that study, they estimate the energy intensity of the communication between two conference locations in Japan and Switzerland, where the communication path was determined in advance. This work is the only one we are aware of that has estimated the energy intensity of a network *path*, and thus has a relation to our proposed estimation of the *carbon intensity* of inter-domain paths.

Top-down approaches In top-down approaches, researchers divide the total energy consumption of a network by the amount of traffic transited by the network over a particular time period. Studies conducted by Koomey et al. [116], Taylor et al. [117], Weber et al. [118], Lanzisera et al. [119], and Andrae et al. [65] are examples of top-down approaches.

Model-based approaches Model-based approaches rely on modelling parts of the Internet based on network-design principles, and on energy-consumption information of device vendors in order to find the total energy consumption of specific Internet parts. Baliga et al. [121, 120], Vishwanath et al. [122], and Hinton et al. [123] propose model-based approaches to estimate the energy consumption of the Internet.

4.8 SUMMARY

This chapter presents an important step towards realizing carbon-aware global routing by designing and implementing CIRo, a practical system to forecast and disseminate carbon-intensity information of inter-domain paths. Our large-scale simulation results suggest that CIRo enables endpoints to substantially reduce their carbon footprint of Internet usage.

Importantly, we note that CIRo may also support the renewable-energy transition of entire energy systems in two main ways. First, renewable-energy sources such as solar and wind have distinct production peaks, which make it challenging to optimally utilize and monetize the electricity produced during these peaks. In this regard, CIRo is valuable because it allows to shift energy consumption to locations where production peaks occur and abundant green energy is currently available. Second, CIRo creates a global competitive environment in which ISPs worldwide can attract traffic by using green energy. This new competitive pressure is also felt in countries where the renewable-energy transition has so far been hindered by inert and monopolistic electricity markets. In these countries, ISPs thus have an incentive to convince local electricity producers to expand the green-energy supply, which may accelerate the green-energy transition especially in developing and emerging countries. In our further research, we are eager to investigate the role carbon-aware routing can play in the global quest for sustainable energy.

5

TOWARDS GLOBAL LATENCY TRANSPARENCY

In this chapter, we develop the second of the two systems for disseminating inter-domain path performance information, specifically designed for the propagation latency of inter-domain paths. The contributions of this chapter are based on a publication at IEEE/IFIP Networking 2024 [3].

5.1 INTRODUCTION

With the deployment of path-aware network (PAN) architectures such as SCION [8], new challenges and opportunities arise for latency-sensitive applications such as video conferencing, online gaming, holographic communication, or the tactile Internet [29]. In contrast to today's Internet, which only offers a single, typically cost-optimized, path, a PAN can offer multiple path options, providing advantages to latency-sensitive applications since endpoints can select the shortest-latency path. In the current deployment of SCION, we witness that a large number of distinct paths are available, with many destinations having over 100 different paths. With the steady expansion of the commercial SCION network, we expect this number to further increase.

For latency-sensitive applications, a research challenge thus emerges on which path to use. The path length in terms of AS hops is unfortunately a weak predictor of the end-to-end latency [124]. Path probing with active measurements would waste the benefit in the case of dozens, or even hundreds, of available paths. In particular, single-packet request-response protocols, such as DNS, would not permit any probing for achieving low-latency operation.

In this work, we explore an approach that adds *propagation* latency information to PAN path information, enabling an endpoint to compute an estimate for the end-to-end propagation latency. The reasons for disseminating the propagation latency instead of attempting to predict the experienced end-to-end latency, which

also comprises the transmission, processing, and queuing latency, are as follows. Transmission and processing latencies are typically negligible compared to propagation and queuing, and often exhibit little variance. Since the amount and nature of cross traffic is usually unpredictable, an accurate estimation of the queuing latency is not possible in most cases. On the other hand, propagation latency is a useful metric, as it enables the computation of the experienced queuing latency, is typically static, and can be measured without relying on a model. Finally, as we will outline below, for use cases such as efficient path probing and improving the fairness of congestion control algorithms, knowledge of the propagation latency provides significant benefits.

We propose the Global Latency Information Dissemination System (GLIDS) for estimating propagation latency information of end-to-end paths in an inter-domain setting. We study the research problem of how to achieve high accuracy for latency estimates while minimizing the network overhead. Furthermore, since GLIDS is based on SCION, where partial paths in the form of path segments are combined into end-to-end paths, we ensure that end-to-end propagation latency estimation is possible for all path segment combinations.

One goal of GLIDS is to find the lowest latency path in a multipath network with a large number paths. For long-lived latency-critical flows, a sender will (continuously) probe available paths and switch to the lowest end-to-end latency path available. Knowledge of the propagation latency gained through GLIDS provides the sender with a clear order and cutoff latency for path probing instead of relying on heuristics which may not find the lowest experienced latency path.

For short-lived flows that finish in a single round trip, e.g., consisting of a single request and response packet pair or where all data fits into the initial congestion window, performing latency measurements to decide where to send a packet may take as long as, or even longer than, the flow itself and thus negate any benefits of the measurement. Even for longer-lived flows, knowledge of the propagation latency gained from GLIDS is valuable to make an informed path selection when sending the *first* packet.

Another use of GLIDS is to improve delay-based congestion control algorithms (CCAs), which adjust a sender's congestion window based on the measured latency inflation as a signal of queuing on intermediate nodes. BBR is a delay-based CCA which has shown to be unfair to competing loss-based CUBIC CCA flows, due to overestimating the propagation latency [23, 24]. By providing a propagation latency estimate, GLIDS enables BBR to infer the current latency inflation and overcome this unfairness.

Compared to the current Internet, the key ingredient enabling GLIDS in the SCION path-aware Internet architecture is the stability of disseminated paths. GLIDS leverages this stability to directly use the latency information disseminated through the control-plane for path selection and thus provides propagation latency transparency. The main contributions of this work are:

- Design of the scalable system GLIDS for estimating propagation latency of end-to-end paths in SCION.
- Evaluating GLIDS through the SCIONLab testbed, emulation in a Mininet-based BBR testbed, and simulations on real-world topologies.

5.2 MOTIVATION AND CHALLENGES

In this section, we highlight issues prevalent in existing latency prediction systems, motivate the need of GLIDS through three concrete use cases, and discuss design challenges.

5.2.1 *Prior Work on Latency Prediction*

Latency prediction for intra- and inter-domain networks has been extensively studied in the past. Latency prediction approaches can be separated into deterministic approaches that distribute aggregated *measured latencies*, and model-based approaches that use statistical *latency models* to infer latency based on network delay monitoring between various vantage points in the network [127, 126, 125]. Due to challenges such as asymmetric routing, unpredictable paths, dynamic load balancing, and lack of transparency on the actually used inter-domain forwarding paths, recent advancements in latency prediction have been focused on model-based approaches.

5.2.1.1 *Intra-domain Latency Prediction*

In the space of intra-domain networking, there has been much work on low-latency networking [128] and latency prediction (often in the context of data centers) [130, 129, 131]. This shows the usefulness of latency prediction in a controlled environment to enhance application performance, reduce overhead, and potentially minimize cost in terms of hardware or computational resources.

5.2.1.2 *Inter-Domain Latency Prediction*

In the space of inter-domain networking, low-latency networking and latency prediction is hindered by a multitude of factors. Since forwarding devices are not controlled by a single entity, they may behave unexpectedly. Additionally, external factors, such as varying queuing delay caused by background traffic, impact the prediction accuracy. Moreover, the lack of transparency in today's Internet reveals little about the used network paths, further increasing the uncertainty of any form of latency predictions. Nevertheless, there has been extensive research in latency prediction in inter-domain networking, demonstrating the desire for latency prediction [132, 127, 133, 126, 125].

5.2.2 *Use Cases*

With the emergence of inter-domain path-aware network architectures, some of the issues prevalent in latency prediction can be addressed. Path-aware networks enable path transparency, which allows endpoints to know the network paths of their traffic, and consequently make use of latency-related information about these paths. Furthermore, path-aware *multipath* networks provide multiple (partially) disjoint network paths for an endpoint to choose from. Endpoints can thus locally optimize paths based on a metric, e.g., latency. We argue that recent developments in inter-domain path-aware networks allow users to accurately *estimate* propagation delay through concrete measurements, enabling innovative and novel use cases. Concretely, we present three use cases.

5.2.2.1 *Delay-Based Congestion Control*

The first use case is providing propagation delay information to delay-based congestion control algorithms to accurately derive the current latency inflation on a path and infer the level of congestion. The knowledge of propagation delay is thus critical for the competition of the delay-sensitive BBR algorithm with the traditional loss-based CUBIC algorithm. In this competition, BBR has been found to be unfair towards CUBIC in terms of capacity sharing, because the sending rate of the BBR flows is directly proportional to their estimate of the path propagation delay. Crucially, BBR flows overestimate this propagation delay as a result of queuing caused by competing CUBIC flows, and therefore maintain an excessively large sending rate [23, 24]. This unfairness can be eliminated with explicit knowledge of path propagation delay, as offered by GLIDS.

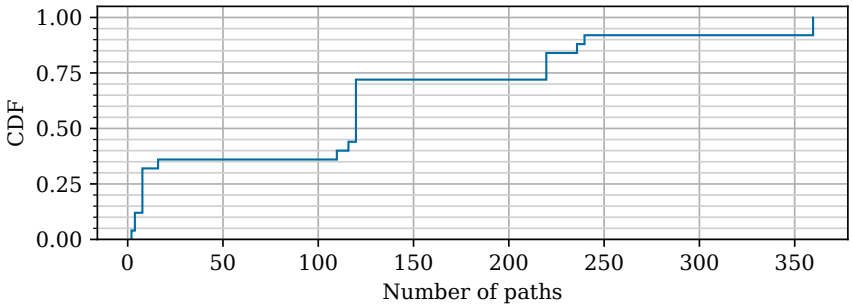


Figure 5.1: Number of distinct paths to all reachable ASes in the commercial SCION network from a single vantage point.

5.2.2.2 *(First-Packet) Latency Estimate*

The next two use cases illustrate the importance of accurate latency information in path-aware networks that offer multiple paths to choose from. For short-lived connections, e.g., query-response protocols such as DNS, it is essential that an RTT estimate is available *before* sending the first packet. There is no benefit for an endpoint to actively measure the latency if all relevant data can be transmitted in the first reply, i.e., the reply may arrive before the measurements are finished.

5.2.2.3 *Efficient Probing*

Finally, we argue that a propagation latency estimate is *necessary* to efficiently probe a multipath network for low-latency paths. Consider an endpoint establishing and maintaining a long-lived low-latency connection. If only a few network paths are available, the endpoint could simply continuously measure the paths and switch to a lower latency path if available. However, this approach does not scale if the network offers hundreds of potential paths, i.e., in a massive multipath network, as this would incur substantial processing, time, and bandwidth overhead. The current commercial SCION network provides a concrete example of such a scenario. Figure 5.1 depicts the number of different paths we can observe from ETH Zürich to 30 ASes located in 5 different ISDs. Although the SCION network is small compared to the Internet, the median number of paths to each destination is over 100, and we expect this number to further increase with the expansion of the commercial SCION Internet. The endpoint could heuristically select and probe a subset of paths

but this does not guarantee finding latency-optimal paths. We discuss a concrete algorithm for efficient path probing based on GLIDS in Section 5.3.3.

5.2.3 *Challenges*

In this section, we list the most important challenges encountered in designing a latency transparency system.

5.2.3.1 *Variable Delays*

One major challenge in latency estimation is to correctly model all variable delays in the network [125]. The dominating variable delay is typically queuing delay due to cross traffic filling up a packet queue on the path. The packet processing and transmission delays are usually negligible in comparison to the propagation and queuing delays, but may be included in a latency measurement or calculated based on the link bandwidth. Since the queuing delay cannot be predicted, our work focuses on the predictable part of the end-to-end latency, in particular the propagation delay.

5.2.3.2 *Disclosing Internal Topology*

While entities can enhance latency estimates by revealing detailed information about their internal topologies, this might reveal sensitive information to a competitor. It is thus imperative that participating entities can decide how much information is revealed.

5.2.3.3 *Load Balancing and Variable Routes*

Internet traffic is rerouted for various reasons, e.g., economic impact, SLAs, and bottlenecks. This can happen for inter-domain paths or for intra-domain paths, e.g., due to traffic engineering or failing links. Accurate latency estimation is challenging under these circumstances and requires regularly updated measurements.

5.3 SYSTEM DESIGN

We propose a latency transparency system that measures and distributes the propagation latency of inter-domain paths at Internet scale and present an efficient path probing algorithm for multipath networks. In designing such a system, we make use of the facts that latency is an additive metric, and any inter-domain

path can be split into several *intra-domain* paths and *inter-domain* links connecting them. The propagation latency of an inter-domain path can thus be computed by accumulating the propagation latencies of all intra-domain paths and inter-domain links. Therefore, we divide GLIDS into two subsystems: (1) The *measurement system* that accurately measures the latency of intra-domain paths and inter-domain links, and (2) The *dissemination system* that globally disseminates latency information.

5.3.1 *Performing Latency Measurements*

In GLIDS, we focus on measuring propagation delay, instead of modeling queuing delay. Note that depending on the measurement method, the propagation delay measurement may include the processing and transmission delay, but in practice, they are typically negligible in comparison to propagation and queuing delay. Hence, GLIDS requires a participating AS to be able to measure the propagation latency of intra-domain paths between their own border routers and of the links connecting them to neighboring ASes' border routers as shown in Figure 5.2.

There exist a wide variety of ways to measure latency, which can generally be separated into three groups [134]: (1) Traditional network measurements [136, 135], (2) SDN-based measurements [137, 138], and (3) Telemetry-based measurements [140, 139], each with different tradeoffs. Since we are interested in the *one-way propagation* latency, the network operator must ensure that queuing delay is factored out, e.g., via packet prioritization or exclusion of "packet queue time", and that potential path asymmetry is taken into consideration, e.g., using one-way latency measurements with synchronized clocks instead of RTT measurements. Additionally, multiple paths with varying latency may exist between border routers due to redundancy, load balancing, or traffic engineering, and paths may change over time. The network operator should measure all possible paths and re-measure changed paths to revoke the out-of-date path segment and re-disseminate it. Optionally, the network operator may also enhance the measurement with the used measurement methodology and a level of measurement (un-)certainty, e.g., assigning a higher level of certainty if a more sophisticated latency measurement approach is used.

5.3.2 *Disseminating Latency Information*

To calculate the latency of an inter-domain path, the endpoints must be provided with the latencies of its constituent intra-domain paths and inter-domain links. GLIDS achieves this in a scalable fashion through path exploration and dissemination. In the exploration phase, each AS on an inter-domain path encodes the latency

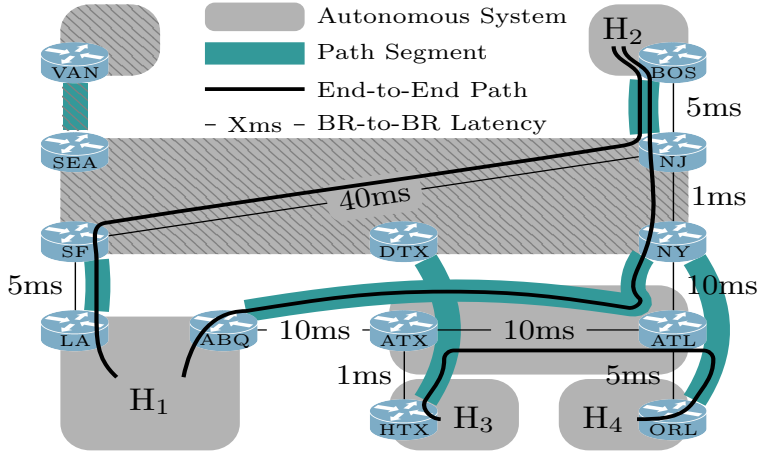


Figure 5.2: Topology where four hosts (H_{1-4}) combine path segments to end-to-end paths and leverage GLIDS to select paths. Diagonal lines indicate core ASes and core path segments.

information of its AS hop in the forwarded path segment. In the dissemination phase, the endpoint retrieves the path segments with the included latency information. Note that GLIDS uses an opt-in approach and an AS can choose to disclose latency information with appropriate granularity based on the desired level of topology secrecy. The privacy aspects are discussed in more detail in Section 5.5.1.

The latency information of each AS hop consists of the propagation latency of the intra-domain path between the ingress and the egress border routers (e.g., SF to NJ in Figure 5.2) as well as the propagation latency of the inter-domain links between the border routers of the neighboring ASes (e.g., SF to LA). Due to possibly asymmetric latencies, the AS encodes the latency in both directions. If the exact intra-domain path each packet can take is not predictable, e.g., due to load-balancing or backup routes, the AS must disseminate the minimum propagation latency of all possible routes—which is necessary for efficient path probing, see Section 5.2.2. In addition to this minimum latency, an AS can optionally provide more information, such as the maximum propagation latency among these paths or a latency distribution. Alternatively, the network operator could make such intra-domain paths (including their latency) accessible via FABRID policies [141] and allow endpoints to explicitly select a specific intra-domain path.

5.3.2.1 *Disseminating Latency in Hierarchical Architectures*

In hierarchical routing architectures like SCION, path segments need to be combined to construct full inter-domain paths. Hence, not only do we need to add the latency information in each path segment, but we also need to ensure that endpoints can reconstruct the latency information of all possible segment combinations. Figure 5.2 shows an example where the lack of intra-AS latency information between SF and NJ would cause H_1 to incorrectly select the path through LA (via SF, NJ, BOS) to communicate with H_2 even though the alternative path through ABQ (via ATX, ATL, NY, NJ, BOS) has a smaller propagation latency. Hence, GLIDS must disseminate the latency information between all border router pairs of ASes at the path segment junctions. To satisfy this requirement of SCION's hierarchical routing with two levels, i.e., core and intra-ISD routing, we propose two different information dissemination mechanisms, one for each level.

In core routing, where the topology is densely connected and routing messages are flooded to a subset of neighbors, we need a mechanism with small overhead per routing message to achieve scalability. To that end, each AS only encodes the latency for the intra- and inter-domain path that the routing message traversed. This is possible since core-path segments are always combined with intra-ISD-path segments, which will contain the necessary latency information to interfaces that are not traversed by the core-path segment.

In intra-ISD routing, where routing messages are only forwarded from providers to customers and the topology is typically sparse, more latency information can be added to routing messages without introducing significant overhead. A trivial approach is the addition of latency information between the egress interface of the routing message and all other interfaces. However, this approach wastes network bandwidth, since some latency information will be duplicated, i.e., two combined path segments will both contain the latency information between two interfaces. To prevent this, only the latency information between the egress interface and interfaces which have a lower AS-local identifier is added. This ensures that any two intra-ISD-path segments, or any core and any intra-ISD-path segment can be combined. For example, in Figure 5.2, hosts H_3 and H_4 need the intra-AS latency between ATX and ATL even though neither segment contains this link.

5.3.2.2 *Scalability*

GLIDS achieves scalability using three mechanisms. First, it disseminates the minimum propagation latency of a path, which does not change frequently, and thus does not require frequent re-dissemination. Second, the existing path-segment

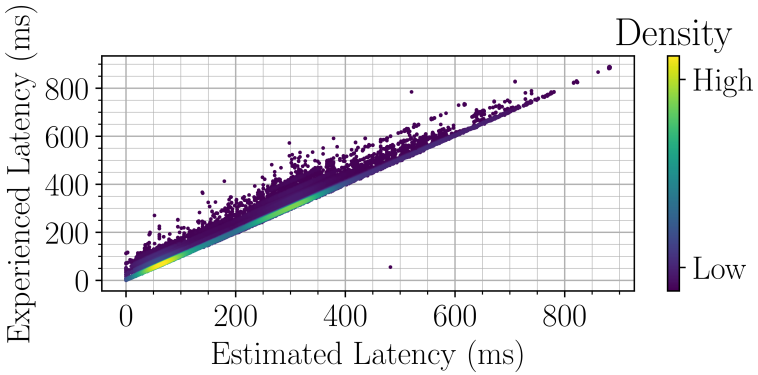


Figure 5.3: Difference between the estimated and the experienced propagation latency in SCIONLab.

establishment and dissemination process is used to disseminate the latency information without introducing additional messages. Third, it leverages the observation that core path segments are always combined in their entirety with intra-ISD path segments, thus no additional intra-AS latency information needs to be added to the core path segments, but only to the intra-ISD path segments.

5.3.2.3 *Obtaining Full Inter-Domain Path Latency*

Once provided with the path segments to construct full inter-domain paths, endpoints compute the full path latency by accumulating latencies of constituent AS hops. Note that an endpoint may only receive partial latency information if some on-path ASes do not reveal this information. In such a case, the endpoint locally decides whether to use the partial information, e.g., by assuming a propagation delay of zero for the missing latency information, or to discard the incomplete latency information.

5.3.3 *Efficient Path Probing*

We propose an efficient path probing algorithm based on GLIDS, which works as follows: (1) sort all paths based on their propagation latencies (assuming zero latency for path segments with missing latency information), (2) measure them one-by-one (or batch-wise) and remember the smallest measured latency, and (3) stop as soon as the propagation delay of the next path is higher than the

smallest measured latency. Since all following paths *must* have higher latency, we can terminate the algorithm early. A requirement is that, if a path has multiple intra- or inter-domain links with different propagation delays, the system always returns the *minimum* value. Otherwise, the algorithm might discard and thus not probe a potential candidate path. The number of probed paths depends on the distribution of latencies, e.g., a higher variance in propagation delays leads to an earlier termination.

5.4 IMPLEMENTATION AND EVALUATION

We prove the feasibility of GLIDS by implementing it on the SCIONLab testbed, evaluate the impact of propagation latency knowledge on the fairness of delay-based congestion control, and show the benefits of GLIDS's first packet latency estimation through simulation on Internet-scale topologies.

5.4.1 *Implementation and Deployment in SCIONLab*

We evaluate GLIDS in the SCIONLab testbed [142]. Our implementation uses RTT measurements, i.e., the minimum measured RTT over the span of 60 s, to infer propagation latency between AS interfaces. The experienced latency is then measured by taking the median of 5 ping measurements.

Figure 5.3 shows that inferring propagation latency based on the RTT, which is arguably the simplest measurement approach, provides a reasonable approximation of the propagation latency since less than 5% of measurements experienced a lower latency than the advertised propagation latency. The overall experienced latency, with a median of 234 ms and a 95th percentile of 551 ms, is relatively high since the SCIONLab network partially consists of overlay links, which may lead to longer paths, and since the SCIONLab nodes are globally distributed in Europe, the United States, and East Asia. However, note that SCIONLab, as a research network, experiences little congestion. In networks carrying large amounts of data and experiencing more congestion, these results might differ substantially and require more sophisticated latency measurement systems.

In addition to the feasibility of the system, we measure the latency reduction of an endpoint that chooses the path with the lowest estimated propagation latency over the default path. Figure 5.4 shows that using GLIDS, 50% of endpoints reduced the latency by over 40 ms, or 25% relative to the original value. In 10% of the cases, endpoints reduced the latency by over 200 ms, or 70%. These results show that a

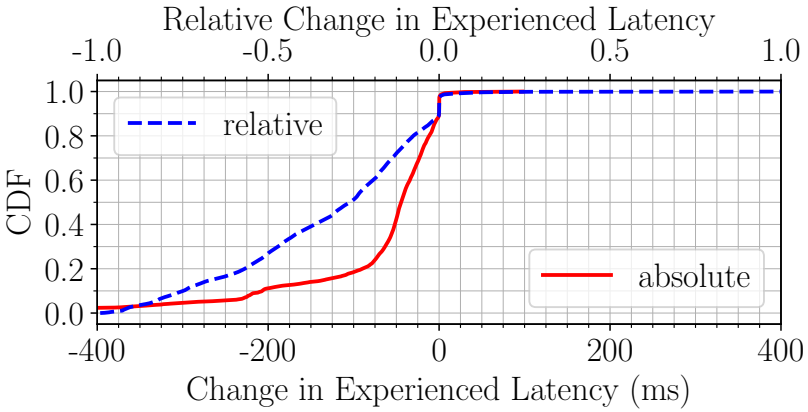


Figure 5.4: Latency reduction (in relative values or ms) when selecting the path with the lowest propagation latency.

multipath-capable network with inter-domain propagation latency estimate can indeed significantly reduce the experienced latency.

5.4.2 *Delay-Based Congestion Control*

In the following, we demonstrate how information about path propagation delay can improve the deployment effects of delay-based congestion-control algorithms (CCAs). In particular, we show that the knowledge of path propagation delay improves the fairness of the delay-sensitive BBR CCA [143] toward the traditional loss-based CUBIC CCA [144].

5.4.2.1 *Setup*

For our evaluation, we emulate the competition among 10 flows on a 100 Mbps bottleneck link using Mininet [145]. Each flow is using a path with 20 ms one-way propagation delay, composed of 10 ms propagation delay on non-shared links and 10 ms propagation delay on the shared bottleneck link. The non-shared links and the shared bottleneck link are intermediated by a switch with a queue size that corresponds to 1.5 bandwidth-delay products of the network path. In this setup, we measure the capacity share obtained by all BBR flows together, depending on

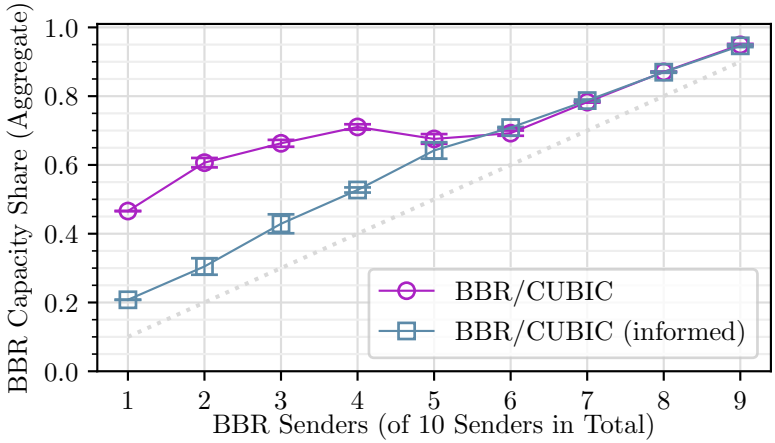


Figure 5.5: Bandwidth-sharing fairness between BBR and CUBIC flows, with and without explicit propagation delay knowledge.

how many of the 10 flows adopt BBR, while the remaining flows adopt CUBIC. We also distinguish two different BBR versions, namely the official BBR version, and a variant named "informed BBR" that we implemented by adapting the BBR source code: While the official BBR attempts to estimate path propagation delays by regular probing, informed BBR works with actual propagation delays, which are known when using GLIDS.

5.4.2.2 *Results*

Figure 5.5 presents the results of this experiment. For all tested configurations, the BBR capacity share is strictly above the dotted gray line that marks the proportional capacity share; hence, BBR is unfair towards CUBIC in every evaluated case. However, this unfairness is particularly pronounced if fewer than 5 flows adopt the official BBR version.

Crucially, this unfairness stems from the RTT-probing behavior of the BBR flows, which try to discover the propagation delay by emptying on-path buffers with sharp contractions of their sending rate, and use the resulting measurement to adjust their congestion window [143]. While this probing behavior indeed allows discovering the propagation delay if only BBR flows share the bottleneck link [146], competing CUBIC flows preserve buffer utilization when the BBR flows are probing, and thus inflate the propagation-delay estimate of the BBR flows [23, 24]. As a

result, the BBR flows maintain an excessively large congestion window, resulting in an unfairly high sending rate. Notably, this effect arises only if the CUBIC flows are numerous enough to keep up buffer utilization in the 200 ms time window when the BBR flows are probing; in our setting, fewer than 4 CUBIC flows cannot fill the buffer during this RTT probing, and thus do not distort the BBR propagation-delay estimate.

Clearly, the propagation-delay estimate is never distorted for the *informed* BBR version, as this version learns the actual propagation delay from GLIDS. This knowledge thus eliminates the overdimensioning of the BBR congestion window, and improves the fairness of BBR towards CUBIC.

5.4.3 *Large-scale Simulations*

In this section, we analyze the benefits of GLIDS for first packet latency estimation, in particular DNS resolution. We perform large-scale simulations on real-world Internet topologies from CAIDA [53] to compare the performance of GLIDS with BGP-based (shortest AS path) routing. This topology contains the relationships and geo-locations of inter-domain links between 12 000 ASes, of which, we extract and keep 2000 ASes, including Tier-1 and the highest-degree Tier-2 ASes. For this evaluation, we simulate BGP with SimBGP [83], and SCION+GLIDS with a in-house simulator based on ns-3 [26]. These simulations discover inter-domain paths in the current Internet and in SCION+GLIDS. We approximate the propagation latency of a path as the accumulation of intra-domain latencies between ingress and egress border routers of each AS-hop on the path. This latency is in turn calculated as the product of the great circle distance between the border routers—calculated using their geo-locations—and the speed of light in the fiber.

Based on these simulations, we analyze by how much GLIDS could reduce this lower-bound (great circle) latency between clients and DNS root servers. Figure 5.6 illustrates the difference between the simulated latency in SCION and BGP, showing that SCION's latency-aware routing and path selection can reduce the lower-bound latency to the 5 root servers in our topology for at least 40% of probes. Furthermore, this latency deflation is at least 20 ms for 20% of probes to any of the root servers. However, the latency to root servers from a negligible portion of probes can be inflated by less than 10 ms due to suboptimal choice of destination AS ingress interface. This is caused by a lack of information on the relative position of the destination root server in the destination AS.

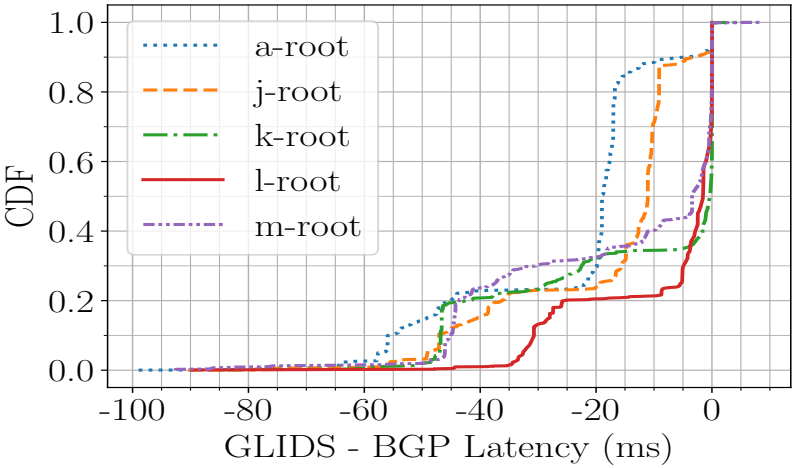


Figure 5.6: Comparison of lower-bound latencies from probes to DNS root servers between GLIDS and BGP.

5.5 DISCUSSION

This section focuses on the practical aspects of achieving latency transparency through GLIDS. In particular, we focus on GLIDS's incremental deployment model, potential hurdles for adoption, and its privacy aspects. Furthermore, we discuss the veracity of the disseminated latency information and highlight additional use cases related to bandwidth reservation.

5.5.1 *Deployment Model*

A crucial aspect of any new Internet protocol is deployability [147], i.e., to ensure compatibility with existing and legacy systems in a partial deployment while providing clear incentives for early adopters. Ideally, existing latency measurement mechanisms and hardware can be reused for internal latency measurements, allowing GLIDS to be deployed at low cost.

GLIDS can be incrementally deployed by having only a subset of ASes include latency information in their routing messages. While this does not yet provide full propagation latency transparency for all Internet users, as long as ASes that are

topologically close to the communicating endpoints offer GLIDS, partial latency information inferred by endpoints can be used for latency-based path optimization. Hence, two endpoints purchasing Internet connectivity from Tier 2 ISPs can achieve propagation latency transparency if their ISPs support GLIDS and are customers of the same GLIDS-enabled Tier 1 AS. Even if no Tier 1 AS supports GLIDS, ISPs can leverage peering and transit links to other ISPs to locally achieve propagation latency transparency.

The incentive of an early GLIDS adopter is to attract network traffic by providing propagation latency information. Note that even if the early adopter does not offer better performance than its competitors, simply the presence of the latency information may lead to endpoints sending traffic on these paths to benefit from the various use cases presented in Section 5.2.2. This holds true especially for delay-based congestion control and first-packet latency estimation, while the usefulness of efficient probing increases together with the adoption rate since more paths can be pruned.

Another important aspect of GLIDS is privacy, since the participating ASes reveal propagation latencies of internal paths. We argue that privacy is often not a concern in GLIDS since it typically does not reveal more sensitive information, such as the exact configuration of internal routers and links, compared to the existing SCION paths. Furthermore, each AS can decide not to reveal certain (privacy-sensitive) latencies. Finally, even in today's Internet, propagation latencies can often be inferred through measurements between different vantage points (although typically with a lower precision).

5.5.2 *Veracity of Latency Information*

One important aspect of GLIDS is ensuring the veracity of the disseminated latency information. A dishonest AS may disseminate wrong information for a financial gain. Artificially increasing the disseminated propagation latencies is typically less problematic since users will send their latency-sensitive traffic on alternative paths. Artificially decreasing the disseminated propagation latencies may attract users to send their latency-sensitive traffic on a sub-optimal path. In GLIDS, the veracity of latency information is protected by including a timestamped signature from the AS that provides the latency information in the respective routing message. Hence, GLIDS provides non-repudiation to ensure that a misbehaving AS can be punished while preventing framing attacks. Note that while this provides the necessary building blocks to implement a system to detect misbehaving ASes, the full design of such a system is beyond the scope of this paper.

5.5.3 *Opportunities For Propagation Latency Transparency*

In addition to the presented use cases, another intriguing use of propagation latency transparency is to enhance path selection for traffic that intrinsically exhibits low queuing delays and where the propagation latency is thus close to the experienced latency. For example, constant rate traffic, which is typically observed in fixed bit rate video conferencing, may not incur significant queuing delay as long as the aggregate rate is below the minimum link capacity and packet pacing is used. Furthermore, instead of relying on all communicating parties to send traffic at their allowed maximum rate, the allowed rate can be guaranteed through bandwidth reservation systems [148, 149]. Propagation latency knowledge may then in turn allow users to select an optimal low latency path with bandwidth reservation, i.e., optimizing both for the required bandwidth and low latency.

5.6 RELATED WORK

We separate related work into three categories: latency measurement systems, measurement-based intra-domain latency prediction, and model-based latency prediction.

5.6.1 *Latency Measurement Systems*

Tan et al. [134] provide an extensive overview of network measurement approaches. They divide them into three categories: (1) Traditional network measurement using active (e.g., ping and traceroute), passive (e.g., NetFlow [135], sFlow [150], or IPFIX [136]), and hybrid measurements, i.e., combination of the two, (2) Software-defined measurements based on SDN [137] and PDP [151] (e.g., SLAM [138]), and (3) telemetry-based approaches, in particular in-band telemetry, which adds telemetry data to data-plane packets. Examples of in band telemetry are: In-band Network Telemetry (INT) [139], In-situ Operation Administration and Maintenance (IOAM) [152], Alternate Marking-Performance Measurement (AM-PM) [140], and Active Network Telemetry (ANT) [153]. All of these measurement systems can be used in GLIDS to measure one-way delays of both intra-domain paths and inter-domain links.

5.6.2 *Intra-Domain Latency Prediction*

Latency prediction is well researched in the setting of intra-domain networks, where high bandwidth, low latency, and many-to-one communication is common. However, this work focuses on inter-domain latency transparency, which faces different challenges, such as the multi-domain environment, less agency over distant links and nodes, and scalable dissemination of latency information.

LACO [154] provides latency-driven network slicing in 5G radio networks by allowing a provider to slice its network. It requires a single provider to offer network slices to tenants, which is different from the distributed nature of GLIDS.

Hermes [155] reroutes traffic in datacenters to avoid congestion and active path probing through RTT measurements. However, detecting congested links and routing around them in an Internet-scale topology is significantly more challenging than in datacenters with limited topology size.

Pingmesh [131] is a large-scale ping-based latency measurement system analyzing the health and performance within and between data centers of a single operator. Deploying such a system in an inter-domain network, brings additional challenges regarding trust and cooperation.

5.6.3 *Model-Based Latency Prediction*

Model-based approaches attempt to construct a latency model for all Internet paths based on available measurements for known paths. Systems proposed by Tabatabaeimehr et al. [125], Krasniqi et al. [127], and Perdices et al. [126] are notable examples. However, providing global latency transparency using model-based latency predictions is challenging since they not only rely on temporal but also on spatial predictions, which reduces the certainty of the provided guarantees. In comparison, GLIDS does not require high-quality traffic matrices, measures and deterministically combines concrete latency measurements of each segment, and distributes latency information via SCION's scalable control plane.

5.7 SUMMARY

GLIDS offers exciting new possibilities for inter-domain networks and moves us closer to the prospect of global latency transparency. In path-aware networks that provide endpoints with hundreds of paths to different destinations, GLIDS enables an informed path selection based on latency information about paths. Such information can further foster competition among ISPs for traffic from latency

critical applications. While speculative, this can increase the competitive pressure on ISPs and provide incentives for the creation of an open market for inter-domain paths.

Part III

NETWORK DEBUGGING

PART OVERVIEW

In the first two parts of this thesis, we develop mechanisms that provide information about inter-domain paths and allow the computation of optimal paths based on the provided information and the various criteria of endpoints. The designed information dissemination systems, however, do not propagate real-time information about the performance of each path, mainly to ensure the scalability of the systems. Furthermore, the provided information may not represent the performance of a path under all different circumstances, e.g., different types of packets.

In this part of the thesis, we develop a network debugging system that enables endpoints to reproduce and examine the experienced performance of inter-domain paths and their constituent segments associated with every ISP on a path. Such a mechanism allows endpoints to avoid ISPs with undesirable performance, thus encouraging ISPs to improve their infrastructure and provide better performance to attract more traffic.

6

PROGRAMMABLE AND VERIFIABLE INTER-DOMAIN NETWORK TELEMETRY.

6.1 INTRODUCTION

Internet applications have become mission-critical for many businesses. As teleconferencing increasingly replaces business travel, and as collaboration between multiple worksites increasingly relies on Internet applications, the Internet has become critical infrastructure for everyday life. When encountering Internet outages or reduced performance, users and technicians alike want to know: which link or system is responsible? Who can help to resolve the issue?

On today's Internet, end-user debugging is largely limited to rudimentary tools such as `ping` and `traceroute`, supplemented by purpose-built services such as bandwidth measurement, website failure detection, and uptime monitors. Unfortunately, these tools suffer from three major shortcomings. First, they are either end-to-end [156–158], complicating fault localization, or they are not based on real data traffic [159–161], allowing networks to preferentially treat measurement traffic over regular traffic, such that their measurements do not accurately reflect the fate of real data traffic. Second, techniques that require participation by forwarding routers [162, 163], such as `traceroute`, do not always get the desired support, or may get replies from non-public IP addresses, making fault localization difficult. Finally, the results of a measurement are not verifiable by external entities.

In this chapter, we propose a distributed network debugging infrastructure, *Debuglet*, in which Autonomous Systems (ASes) deploy services intended for network-measurement applications. The *Debuglet* executor, distributed at the edge of ASes, provides a policy-constrained remote code execution environment. By distributing and executing applications tailored to the intended network measurement to the executors, endpoints perform real-world data-plane operations at different vantage points, allowing them to localize the origins of network failures with high accuracy.

Endpoints pay to run their network debugging applications in these environments using some form of micro-payments, using traditional currencies or cryptocurrencies brokered through their Internet Service Providers (ISP). The output of these applications can then be certified by the deploying AS, allowing third parties to verify the measurement results.

The main contributions of this chapter are:

- Demonstrating that real-world network performance depends on the type of network traffic being sent, thus to achieve accurate measurement the packet type of the probes should match the packet type of the traffic whose performance is being debugged,
- Designing the distributed *Debuglet* architecture, allowing programmable and verifiable network measurements in inter-domain networks, enabling fine-grained network fault localization, and
- Introducing a marketplace model for purchasing measurement service from remote ISPs, incentivizing the deployment of the architecture and paving the path to form an ecosystem for Internet debugging.

The contributions of this chapter are based on my publication at IEEE ICDCS 2024 [4].

6.2 MOTIVATION

The inadequacy of today's ICMP-based network measurement tools to reproduce and accurately locate network faults in federated networks motivates us to design a new system addressing these problems. This inadequacy stems from the differential forwarding treatment by routers, i.e, routers' forwarding decisions do not only rely on the destination of packets but also on type, size, and values in the header fields. We expose this problem through real-world measurements. We design and carry out experiments demonstrating that the protocol of data packets can affect the network performance.

Protocols We conduct experiments on the following protocols:

- UDP packets,
- TCP packets without any special flag, with random sequence numbers,
- ICMP echo requests,

- Custom IP packets with an unassigned protocol number (201)

Performance Metrics We consider two network performance metrics: round-trip time (RTT) and drop rate.

Experiment Setup We set up identical virtual machines (VM) hosted by Digital Ocean in seven locations around the globe: Bangalore, Frankfurt, London, New York, San Francisco, Singapore, and Sydney. We measure RTT and drop rate of considered protocols between the VM in London and other VMs in the other six locations. To achieve this, we deploy a client Go application in those six locations, each of which sends probes to the VM in London in a round-robin manner, rotating between the considered protocols with a period of one second and sending one probe packet of that protocol's type. The server Go application in London replies to these probes, and the client collects experienced RTTs and drop rates based on the received replies. To ensure that any performance differences are only due to the difference in protocols, the applications ensure that the total length of each packet, from the beginning of the layer-3 header, is the same for all four protocols. We conduct measurements during 24 hours.

Based on our investigation using traceroute, traffic between these datacenters traverses the public Internet as the cloud provider does not own a global network connecting all its data centers; thus, our results represent the performance of inter-domain networks.

Empirical Results Table 6.1 summarizes the measurement results. The results suggest that the network's forwarding behavior differs for different transport protocols. ICMP's and raw IP's RTT demonstrate greater stability compared to UDP and TCP. For ICMP packets, this phenomenon could be explained by routers treating them specially, as they are mainly used for network debugging.

Conversely, UDP demonstrates the highest variation in measurements. Because it is generally assumed that applications using UDP can tolerate some packet re-ordering, the variation may be caused by route diversity, from routers load-balancing UDP on a more fine-grained basis than per-flow or per-flowlet, as is typical for TCP flows.

TCP experiences the highest drop rate. One explanation is that routers deprioritize TCP packets on congested links to make senders reduce their transmission rate, since UDP may not respond to loss as significantly.

Figure 6.1 shows a 4-hour window of the 24-hour RTT measurement between London and New York. UDP and TCP consistently experience lower RTT than ICMP

Table 6.1: RTT and drop rate between the specified location and London. Each measurement consists of 86400 packets, one per second over a day. Data is expressed in milliseconds. Loss rate is given in per-thousandths (‰).

Location	UDP		TCP		ICMP		Raw IP	
	mean	std	mean	std	mean	std	mean	std
Bangalore	146.01	7.01	158.05	5.27	145.44	3.89	151.44	2.87
	0.23 ‰ lost		1.72 ‰ lost		0.57 ‰ lost		0.41 ‰ lost	
Frankfurt	14.75	1.78	14.72	1.22	11.95	0.51	15.36	0.55
	0.00 ‰ lost		1.09 ‰ lost		0.01 ‰ lost		0.00 ‰ lost	
New York	73.94	6.64	71.58	6.12	76.08	3.98	76.47	4.02
	5.59 ‰ lost		16.19 ‰ lost		0.24 ‰ lost		0.27 ‰ lost	
San Francisco	134.79	1.00	134.42	0.70	134.62	0.66	135.09	1.71
	0.00 ‰ lost		1.56 ‰ lost		0.02 ‰ lost		0.03 ‰ lost	
Singapore	176.14	10.04	176.95	4.33	181.74	3.00	178.98	4.61
	0.09 ‰ lost		1.74 ‰ lost		0.06 ‰ lost		0.03 ‰ lost	
Sydney	274.01	7.79	278.60	5.19	277.99	5.15	278.44	5.18
	0.50 ‰ lost		1.09 ‰ lost		0.96 ‰ lost		1.01 ‰ lost	

and raw IP. However, in some periods, a sudden increase of about 5 ms is observable, signaling a change in forwarding behavior, e.g., route changes. Figure 6.2 plots 24-hour results between London and Frankfurt. UDP results show four clearly visible clusters, which we hypothesize as representing four different routes on which UDP traffic is forwarded. Another interesting observation is that for several hours the RTT of UDP and raw IP packets show a noticeable increase that is not reflected in ICMP nor TCP. Figure 6.3 shows that UDP's RTT between Bangalore and London is distributed over a 30 ms range, almost randomly. Other measures, even though being consistent for relatively short periods of time, vary several times during a day, without any clear correlation.

These results indicate that measurement packets and data packets of different types can receive differing treatment by networks. This differing treatment means that performance issues for a TCP application are best diagnosed using TCP packets, rather than using ICMP or UDP packets. Although this conclusion is driven from ping measurements, it also holds for traceroute, which also uses ICMP, making traceroute packets distinguishable from data packets. However, it is possible to perform traceroute with TCP or UDP packets, i.e., encapsulated in IP packets with increasing TTL. However, regardless of the protocol used, there are two important limitations with traceroute: (1) Responding with ICMP TTL exceeded message is disabled or rate-limited on many routers, and (2) Routers responding with ICMP

TTL exceeded message process such messages on the slow path, while data packets are processed in the fast path, resulting in different performance experienced by data and measurement packets.

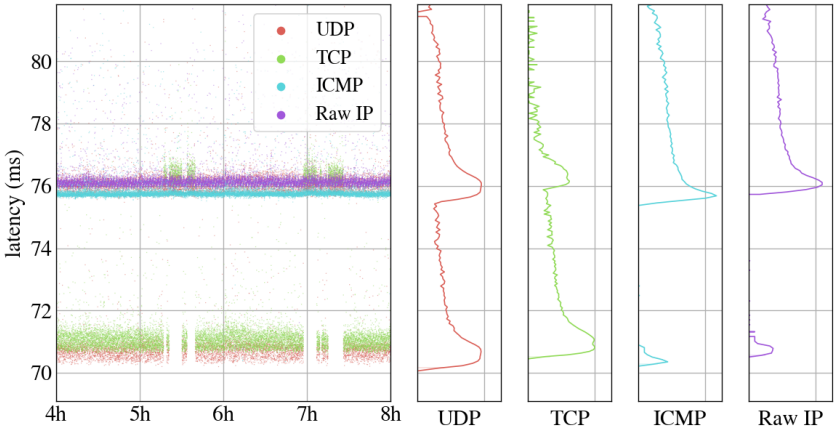


Figure 6.1: New York - London RTT, over 4 hours. The leftmost plot shows latency variations over time, while the four vertical line plots represent the density function of latency distribution for each protocol type, logarithmic scale.

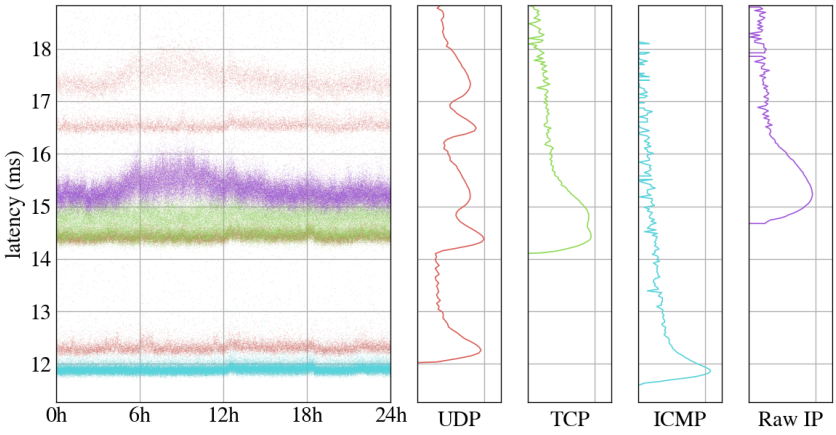


Figure 6.2: Frankfurt - London RTT, 24 hours.

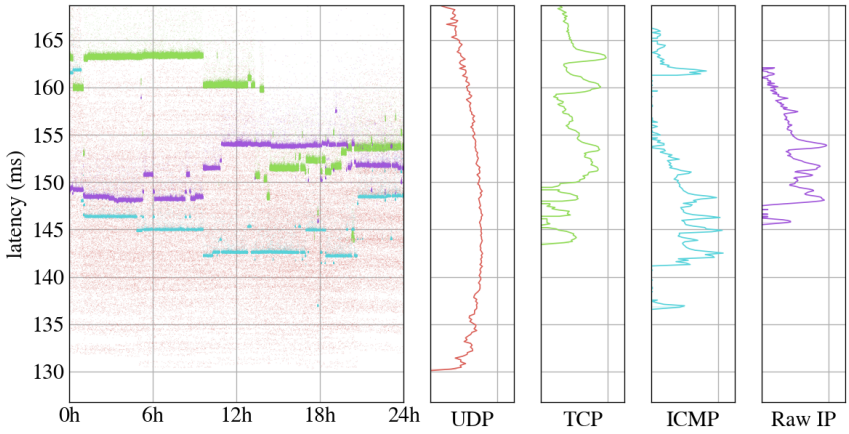


Figure 6.3: Bangalore - London RTT, 24 hours.

6.3 DESIGN PRINCIPLES

The *Debuglet* architecture aims to enable network administrators to locate network failures in an inter-domain network. Inspired by the motivation presented in Section 6.2, we lay down the principles we pursue in the design of *Debuglet*.

Segment-by-Segment Network Measurements To accurately locate network failures, it is necessary to perform debugging per network segment. For instance, if a network failure occurs on an inter-domain link, the optimal source and destination locations for measurement packets for network debugging would be the edges of consecutive ASes. Through this approach, network inspection is performed individually for specific inter-domain or intra-AS links, isolating each measurement to identify network failures independently.

Reproducibility As demonstrated earlier, the application data packet type affects the forwarding decision of on-path routers. For example, as shown in Figure 6.4, probing packets may be assigned to a priority queue over regular data packets and can be transmitted through more stable links. Thus, forwarding delays can vary based on the protocol, size, and header field values. Additionally, changes in the destination address—and sometimes also the source address—can influence the decision of forwarding interfaces. Therefore, to ensure a consistent network

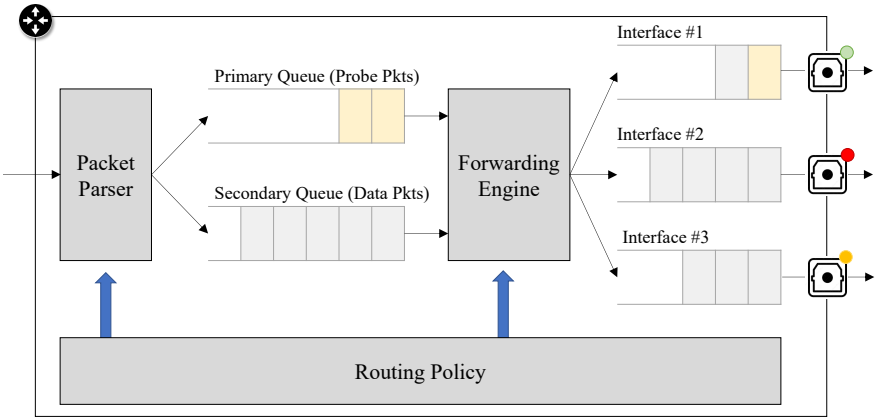


Figure 6.4: Forwarding devices may handle packets differently based on their type, resulting in different forwarding decisions.

experience, replicating the same type of application data packets and transmission through the same path (i.e., the same interface port) is required.

Enabling Unidirectional Measurements It is necessary to allow unidirectional fault localization on the Internet because of two reasons: (1) Internet paths may not be symmetric, and (2) Load distribution on different directions of each link on a path can be different, resulting in different performance in forward and backward directions, e.g., one link can be congested in the forward direction and uncongested on the backward direction, resulting in more queuing delay and drop rate on the forward direction. To distinguish faults on the forward path from the ones on the backward path, *Debuglet* should provide the ability to measure the performance of each direction of the path.

Transferability and Verifiability of Measurement Results It might be beneficial for networks to reuse the measurement results so that no other domains need to perform similar measurements to locate the same network fault, scaling down the total number of probes sent and ultimately saving resources and costs for network debugging. Therefore, the *Debuglet* architecture should support publishing the results of measurements if desired. The main challenge in enabling this capability is the verifiability of measurement results: how can a domain trust that the results published by other end domains provide a realistic picture of the measured segment

in the network? In other words, how can the system provide a guarantee that no domain can deliberately provide an ungrounded worse or better picture of an ISP's performance by publishing fake measurement results or by cherry-picking the worst or best real measurements and just publishing those results? Such behavior can be motivated by domains trying to damage the reputation of other domains, or claiming a refund in case of a service level agreement.

6.3.1 *Requirements*

We describe the prerequisites we need to follow the principles of designing *Debuglet*.

Remote Code Execution (RCE) To be able to reproduce data packets close to faults in an inter-domain network, where faults can occur out of end domain's administration, it is necessary that end domains execute code at remote domains near network faults. Furthermore, RCE is necessary for enabling unidirectional network measurements as it facilitates the programmability of debugging applications, providing control over both sender and receiver sides of measurements.

Path Awareness Control over forwarding paths is one of the design principles of *Debuglet* which can only be achieved through path-aware networking [164]. Path-aware network architectures provide endpoints with information about network paths to any destination and allow them to select paths based on their needs. Therefore, to achieve control over forwarding paths, *Debuglet* relies on path-aware architectures such as SCION [10] and segment routing [165]. The main benefit of leveraging these two specific architectures is that they provide path control at a fine granularity – in the case of SCION the ingress and the egress links of each AS on the path to the destination.

6.4 THE DEBUGLET ARCHITECTURE

6.4.1 *Overview*

The *Debuglet* architecture consists of a *data plane* that enables running custom network measurements at precise locations in a public inter-domain network and a *control plane* that enables the scheduling of measurements and verifiable publishing of the results. Every measurement performed using *Debuglet* requires two distinct executors at two points of a path and two pieces of application: the *Debuglet* client

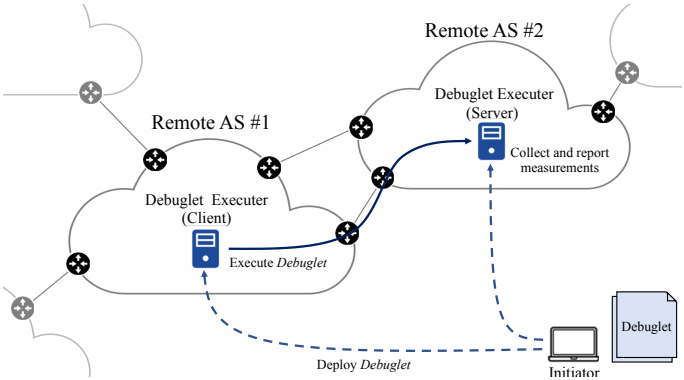


Figure 6.5: High-level overview of *Debuglet*.

that starts measurements and *Debuglet* server that receives measurement packets, which are deployed by the initiator as shown Figure 6.5.

Network measurement in *Debuglet* architecture is a five-step process:

1. Upon experiencing network anomalies, network endpoints request debugging from their ISP.
2. The ISP generates *Debuglet* applications (or just *Debuglet*s) based on the reported issue, and requests the ASes with executor instances close to the points of interest on the forwarding path to run the *Debuglet*s.
3. The ASes who accept the request deploy the *Debuglet* on the agreed executor instances.
4. Executors run the *Debuglet*s, and collect measurements.
5. Executors report the results back to the requesting AS.

6.4.2 *Data Plane*

The data plane of the *Debuglet* architecture has two building blocks: (1) *Debuglet*s, which are pieces of application that produce and send data packets to measure network performance at selected locations, and (2) *Debuglet* executors, which are machines deployed at ASes to execute *Debuglet*s on behalf of end domains, collect the results, and submit them to the control plane.

Debuglet Executor In the *Debuglet*'s distributed network-debugging infrastructure, *Debuglet* executors are small-scale cloud services intended for network measurement applications. To allow the safe execution of unverified code from other ASes, the received application must be executed in a sandboxed environment that provides memory safety, and finishes in a bounded number of instructions. Although virtual machines (VMs) can achieve these properties, significant setup time and effort of VMs inhibit agile network debugging. To overcome this, we build our architecture using WebAssembly (WA) [166, 25] to run *Debuglet*s[1].

By default, a WA application operates within an isolated environment and lacks direct communication capabilities with the external world, except for input parameters and return values. The executor therefore provides these bytecodes with (1) buffers from/to which WA applications can read or write raw application data, and (2) an API through which WA applications can request sending or receiving packets with a variety of transport, network and link layer protocols (if applicable). Therefore, WA applications have the freedom to choose the application layer protocol, content, and data rate. To facilitate network transmissions, dedicated memory regions are designated to serve as buffers. These buffers are assigned to specific namespaces, such as `udp_send_buffer` or `tcp_receive_buffer`, making them easily accessible as global variables. Another buffer is also allocated to store the result of the execution. The `output` buffer serves as the location where a WA bytecode stores the results of network measurements. The output can then be certified by the deploying AS of the executor, allowing third parties to verify the measurement results.

Debuglets A *Debuglet* is compiled as WA bytecode that provides the `run_debuglet` function as the entry point, serving as a *main* function in a typical program. The bytecode can perform arbitrary computations until its execution concludes, after which it must return a result.

In order to successfully execute a *Debuglet*, it is accompanied by a manifest upon submission of the application. The manifest is evaluated by the remote AS prior to execution. This manifest contains the following information:

- Resource requirements, including CPU time, execution duration, peak memory usage, and the number of packets sent and received,
- The list of addresses desired to be contacted, and

1 Some alternatives are also available, e.g., *Extended Berkely Packet Filter* (eBPF) [21], enabling the execution of customized code in the kernel, or uBPF [22], a user-space implementation of the same concept, or the Dandelion system [167].

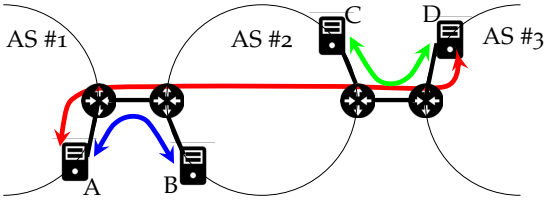


Figure 6.6: Executors are co-located with all border routers of ASes, achieving network debugging in an inter-domain link granularity.

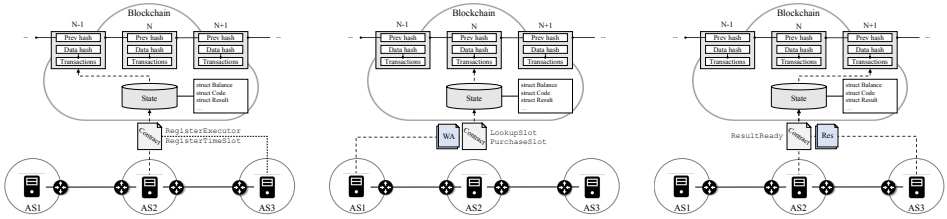
- Specific capabilities required for execution.

Location of Executors The location of executors in each AS is a crucial part of the design of *Debuglet* architecture. Network operators need to know where they should locate *Debuglet* executors for two reasons: (1) Resource provisioning, and (2) Ensuring that *Debuglet* measurements do not disclose secret internal network topology. Furthermore, the locations of executors in each AS determine the achievable accuracy of fault localization. Thus, identifying the location of executors requires the identification of the desired fault-localization accuracy.

We consider co-locating executors with border routers—a router connecting an AS to a neighboring AS—meaning that an AS deploys executors preferentially at its borders (other alternative deployment scenarios are discussed in Section 6.6.6). This is because in inter-domain fault localization, end domains are interested in finding which ASes or inter-domain links on a path are responsible for a network failure.

Figure 6.6 illustrates how such a deployment model enables distinguishing faults within an AS and on inter-domain links. Consider if an end domain has suspected that the segment between the egress border router of AS #1 and the ingress border router of AS #3 is responsible for the performance degradation, suggesting that the fault can occur (1) between AS #1 and #2, (2) within AS #2, or (3) between AS #2 and #3. Assuming that all ASes have deployed one executor co-located with each of their border routers, a remote domain can validate the mentioned three hypotheses by pursuing the following procedure:

1. Deploying a pair of *Debuglet*s at executors A and D that run measurements to validate that the segment (red) between the egress border router of AS #1 and the ingress border router of AS #3 is in fact faulty, and measuring the performance on this segment,



(a) ASes register their executors along with available time slots to the blockchain network through a global smart contract.

(b) An initiator looks up available slots and, upon finding them, submits *Debuglet* applications and the tokens to be paid.

(c) After running measurements, executors report results and retrieve the payment from the smart contract.

Figure 6.7: Basic *Debuglet* execution model with a smart contract.

2. Deploying a pair of *Debuglet*s at executors A and B that run measurements to evaluate the performance of the link between ASes #1 and #2 (blue),

3. Deploying a pair of *Debuglet*s at executors C and D that run measurements to evaluate the performance of the link between ASes #2 and #3 (green),

4. Deriving the performance of the part of path *within* AS #2 based on the measurements collected in the last three steps for the whole segment and each of the inter-domain links connecting to AS #2.

Note that we intentionally do not measure the performance of AS #2 by deploying a pair of *Debuglet*s at executors B and C directly inside the target AS, because in that case the traffic would be *intra-domain* traffic within AS #2, which might be treated differently from the original *inter-domain* traffic coming from AS #1 and going to AS #3, hindering the accurate reproduction of the original forwarding behavior.

6.4.3 *Control Plane*

The responsibility of the *Debuglet* control plane is to (1) provide a public list of *Debuglet* executors, (2) schedule *Debuglet* executions such that it enables the synchronized execution of two applications in two independent ISPs in a Internet-scale environment where only a limited number of requests can be accommodated

at each executor due to finite resources, and (3) allow ISPs to publish the results of *Debuglet* executions publicly in a verifiable manner, so that other entities cannot alter or create fake results. Furthermore, the control plane *can* provide an interface between the initiators of *Debuglet* applications and the executors, through which they can exchange application and results. In this section, we present a centralized design, while we sketch a possible decentralized alternative design in Section 6.6.1.

Blockchain-based Marketplace As an instantiation of such a control plane, we propose a marketplace model for *Debuglet* measurements through which domains trade time slots for measurements, exchange *Debuglet* applications, and *can* publish measurement results. This model allows for simple scheduling of measurements without requiring a complex algorithm: an initiator can purchase two measurement slots for the same timespan at the two executors of its interest, and publish the bytecode for each purchased slot. The use of blockchains for implementing such a marketplace ensures verifiable publishing of results: All transaction history for the results published by the *executors* is traced in the blockchain, i.e., none of the *Debuglet* participants can modify the results without others noticing. Using blockchains also allows for integrating a payment method with the marketplace.

Integrating with Smart Contract We design such a marketplace as a smart contract. Thus, all executors and initiators join a blockchain as nodes.

The smart contract defines application and results as objects: An application object contains an object ID, a string containing the actual WA bytecode, and the tokens that the initiator embeds to be transferred to the executor upon completion of the *Debuglet* execution. A result object contains its object ID and a string containing the actual results.

The smart contract defines the following data structures as its state to accomplish its task as a measurement marketplace:

- ExecutorAddressMap: A map indexed by the concatenation of the AS number and inter-domain interface[2] identifier ($< AS, intf >$) associated with executors to their node's address. The map enables deploying *Debuglet* applications to the intended executors. If multiple physical executors exist for a $< AS, intf >$, the smart contract does not distinguish between them. Instead, it is the task of their administrating domain to coordinate them.

2 An inter-domain interface specifies either end of an inter-domain link connecting two neighboring ASes.

- ExecutionSlotsMap: A map indexed by $< AS, intf >$ to the sorted list of available non-overlapping time slots for the corresponding executor. Similar to the cloud's IaaS model, ASes provide the executors' time slots from which initiators can choose. Each time slot is associated with the following five tuples: (1) The number of available CPU cores, (2) The available memory capacity, (3) The assigned network bandwidth, (4) The start and end time in Unix UTC time, and (5) The price for the slot.
- ApplicationsMap: A map indexed by the concatenation of the *Debuglet* client's AS number and interface ID, the *Debuglet* server's AS number and interface ID, and the time slot ($< AS_c, intf_c, AS_s, intf_s, t_{start}, t_{end} >$) to the list of vector of *Debuglet* applications associated with the tuple, pointing to the applications stored in the blockchain.
- ResultsMap: A map indexed by the object IDs of *Debuglet* applications to the results struct.

Figure 6.7 shows the life cycle of *Debuglet* for a measurement. The smart contract defines the following functions that are called by different parties.

Bootstrapping An executor calls `RegisterExecutor()` to register itself at the smart contract. It provides its AS number and interface ID as arguments to the function. This function adds the mapping between the AS and interface ID of the caller and its address to ExecutorAddressMap.

Later, via `RegisterTimeSlot()`, executors register their available time slots by provide their AS number, interface ID, and the list of time slots (a time slot is defined by a 5-tuple explained in the ExecutionSlotsMap in this section). The function call first checks that the provided AS number and interface ID are, in fact, associated with the calling executor based on ExecutorAddressMap and updates the ExecutionSlotsMap.

Initiating Network Measurements An initiator calls `LookupSlot()` to first look for available slots for its desired measurements. It provides as an argument the AS number and interface ID of the executor running the client application and the same information for the server side, the required number of cores, amount of memory, and bandwidth for both client and server applications. The function checks by looking up the ExecutionSlotsMap, when the first available time slot that both to-be-involved executors can accommodate the measurement would be, and how many execution slots need to be purchased at each executor. The function returns the price that needs to be paid and the first possible time slot to the initiator.

To buy a slot, the initiator calls `PurchaseSlot()`. The initiator provides the following information as the arguments to the function: the client's and server's AS and interface IDs, the time slot's 5-tuple, the number of execution slots it desires to purchase from each of the executors, the string representations of the WA bytecodes of the client and server, and the cryptocurrency tokens required for running each of the client and the server application on each executor. The function first verifies that the embedded tokens suffice for the specified execution slots. Then, it creates two application objects, embeds the tokes in them and adds them to ApplicationsMap, and emits events to notify the executors. The executors, which must have subscribed to the event with arguments containing their AS number and interface ID, retrieve the applications and schedule them for the requested time slot.

Reporting Results An executor calls `ResultReady()` after it has executed the *Debuglet*. The executor provides the object ID of the executed *Debuglet* and the string representation of the results as the arguments. The function transfers the embedded tokens in the application object to the executor's address. It also creates a result object, in which it embeds the result string, and inserts it into the ResultsMap. Finally, it emits an event to notify the initiator, which should have subscribed to notifications with the object ID of its application, of the readiness of the result.

`LookupResult()` allows any node to search for the result of any measurement. The function looks up ResultsMap for measurements and returns the list of object IDs of results together with the time they were executed and the object ID of their associated applications. Note that an initiator may want to keep the results private by encrypting the results in the client and server applications using a cryptographic key embedded in the applications. In that case, the results are not readable by third parties.

6.5 IMPLEMENTATION AND EVALUATION

6.5.1 *Implementation*

We implement a proof-of-concept of all components of the *Debuglet* architecture, i.e., executors, sample *Debuglet* server and client application, and the smart contract to control all *Debuglet* procedures. We implement each executor as a Go application using Wasmer [168] WA [25] runtime to run *Debuglet* WA bytecodes. We write *Debuglet* client and servers as Rust applications and compile them to WA bytecodes. We write the smart contract in Move language [169] and deploy it on a local instance of the Sui blockchain, a modern blockchain capable of thousands of

transactions per second, sub-second finality, and low transaction cost. We deploy our implementation in the SCIONLab [142, 170] testbed, a global testbed for the SCION Internet architecture.

6.5.2 *Evaluation Results*

We evaluate the feasibility and practicality of the *Debuglet* architecture by evaluating three important aspects of its design, i.e., (1) the impact on the accuracy of running WA bytecode for measurements, (2) the delay-to-measurement introduced by the control plane design, and (3) the cost of using blockchains as the control plane of the system.

The Impact of Running WA on Measurement Accuracy To this end, we developed:

- Two native Go applications, one acting as a UDP client, sending packets at a steady rate, and the other as a UDP echo server, and
- Two WA Debuglets with exactly the same functionalities as the native Go applications mentioned above.

We then performed four experiments to quantify the delay added by using WA.

1. Debuglet to Debuglet (D2D), where both the client and server are Debuglets,
2. Application to Debuglet (A2D), where the client is a native application, and the server is a Debuglet.
3. Debuglet to Application (D2A), where the client is a Debuglet, and the server is a native application.
4. Application to Application (A2A), where both client and server are native Go applications.

We ran these experiments simultaneously for one day, between virtual machines located in London and New York, sending one packet per second for each experiment. In total, we collected $4 \cdot 86400$ data points.

Figure 6.8 illustrates the results of these measurements. According to this figure, D2D measurements have a mean latency of 75.12 ms while A2A measurements have a mean latency of 74.81 ms. Therefore, WA execution adds a delay of approximately 300 microseconds, which is attributable to the additional operations for switching between Go and WA. D2A and A2D measurements fall in between, with latencies

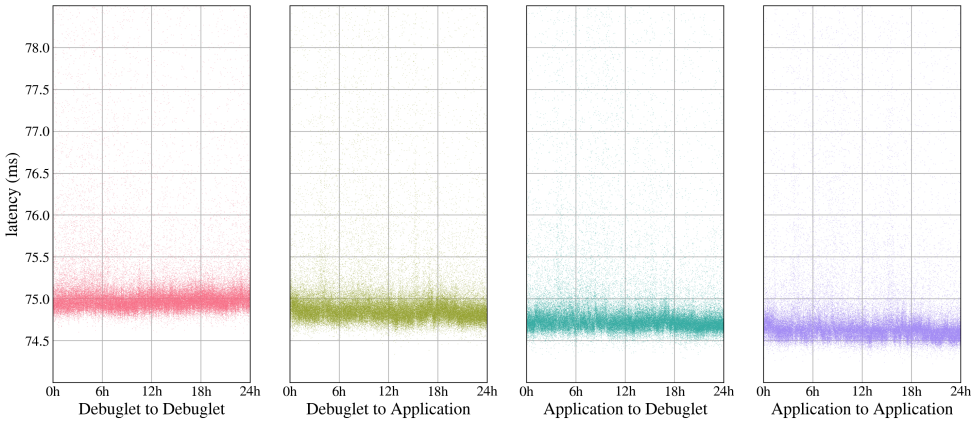


Figure 6.8: Latency measurements using different Debuglet and application combinations. Each dot represents a measurement.

of 75.01 ms and 74.88 ms, respectively. Importantly, the figure suggests that WA execution does introduce some noise to the measurements, but an almost constant delay, which can be offset from the results if the execution environment is known, thus enabling extraction of the ground truth measurement results.

In terms of packet loss, our measurements show negligible difference between measurements: D2D, 1.68% of the packets are lost, while A2D experiences 1.38% loss, D2A 1.66%, and A2A 1.71%.

The Delay-to-measurement The delay-to-measurement is an important aspect of the design and implementation as it is a determining factor in how fast the system can locate faults. Importantly, if faults are short and transient, long delay-to-measurement can cause missing the fault. Delay-to-measurement of *Debuglet* comprises three delays: (1) The blockchain operations delay, (2) The waiting time until the scheduled time slot, and (3) The WA environment's setup time. Regarding the first delay, modern blockchains like Sui achieve very high transaction throughput (more than 100,000 per second) and very low finality latency (less than half a second) per transaction [171]. In our implementation, two transactions are on the critical path of running a measurement, i.e., calling `LookupSlot()` and `PurchaseSlot()`, which are negligible. The second delay, which contributes to the majority of delay-to-measurement, can only be evaluated in an operation as it

Table 6.2: Cost of submitting *Debuglet* application to the Sui blockchain in SUI for different application sizes. Each SUI is equivalent to $0.94 as of May 14th, 2024 [172]. The storage rebate is refunded after the stored data is freed up.

application size	Total Cost (in SUI)	Storage Rebate (in SUI)
0 B	0.01369	0.00430
100 B	0.01585	0.00632
1 kB	0.03527	0.02456
5 kB	0.12160	0.10562
10 kB	0.22953	0.20696

depends on the demand for measurements, the available resources for executors in ISPs, and how often faults occur in the network. Therefore, despite its determining effect, this delay is neither a matter of design nor the implementation of *Debuglet*. However, because of the marketplace-based IaaS model we propose in this work for executing *Debuglet* application, we expect ISPs to dedicate resources to executors proportionally to the demand for running application, resulting in enough resources available for running measurements, and thus, limited wait time for measurements. We evaluate the third type of delay, i.e., execution environment setup time, for different WA bytecodes, and observe an almost constant setup time of around 10 ms across all executions. To conclude, the design and implementation of *Debuglet* data and control planes allow for a *sub-second* reaction to an experienced fault.

Blockchain Costs Blockchains charge nodes for transactions they perform and the data they store on the blockchain. This means that deploying the *Debuglet* infrastructure introduces additional costs both to initiators and executors. We evaluate these costs for submitting and storing a *Debuglet* application using the Sui blockchain. The cost for results has a similar pattern, i.e., a constant cost and a storage cost proportional to required storage. Table 6.2 shows the price (in SUI) for submitting and storing applications with different sizes on the Sui blockchain (the main net). The cost can be significantly lowered by storing applications or results off-chain and only storing a link to the stored data and a hash of data on the chain, so that the data can be verified against the on-chain hash. Consequently, keeping only the hashes of the data and code on-chain, the Sui transaction fees amount to about 1 cent, which is sufficiently low for most use cases. If desired, a

reduction in transaction costs can be achieved by using optimizations to perform micropayments on blockchains [173].

6.6 DISCUSSION

6.6.1 *Alternative Channel for Discovering Executors*

A logically centralized control plane for the *Debuglet* system, such as the marketplace model proposed in Section 6.4.3, integrates executor discovery with code and result exchange, fine-grained scheduling of executions, verifiable result publication, and payment for the provided service. However, such a model entails a single point of failure, i.e., the marketplace, and assumes that all ISPs agree on what entity should operate the marketplace. To address these issues, we present an alternative decentralized approach to discover executors without relying on a centralized controller. In the decentralized approach, ISPs advertise the address and location of their executors as route metadata encoded in routing messages they originate. Thus, other domains learn about the executors through the inter-domain routing protocol, which is a distributed protocol executed by all domains. An initiator of *Debuglet* applications uses these addresses to directly contact the executors. They need to bilaterally negotiate the scheduling and the price of the execution and the method of payment. The initiator sends the applications directly to the executors, and upon the end of the execution, the executors return the results directly back to the initiator. As the results are not publicized by the trusted marketplace, they would not be publicly verifiable. Both the centralized and the decentralized approach can coexist, and we anticipate that both would be used catering to different application scenarios.

6.6.2 *Incremental Deployment*

Though full deployment provides the best picture of how each AS impacts path performance of real network packets, partial deployments allow direct measurement between any two deploying ASes, and between a deploying AS and either endpoint. For applications such as service-level agreement (SLA) enforcement, a small deployment of *Debuglet* services at the ISP, or, ideally, at neighboring ASes, will allow for verifiable measurements of the performance between a subscriber (as measured at the Customer Premises Equipment, for example) and their ISP and its providers. Further deployment of *Debuglet* services would help ISPs prove their innocence.

An AS that knows it contributes to poor performance may be disincentivized from deploying *Debuglet* services as it tries to conceal its contribution to poor network performance; however, increased deployment of *Debuglet* services at other ASes will make it increasingly clear where the bottleneck lies. For example, a Tier 3 ISP with poor performance will be exposed because its customers suffer poor performance that cannot be isolated to its providers. Also, a Tier 1 or Tier 2 ISP, despite not deploying *Debuglet* services, will still be on the routing path for many *Debuglet*-to-*Debuglet* or *Debuglet*-to-client paths, and its contribution to network bottlenecks will become more clear as its neighbors (both upstream and downstream) deploy *Debuglet* services. Therefore, though an AS can initially hide its poor performance by not deploying *Debuglet* services, as the system gains adoption, that AS will be increasingly exposed over time.

At the same time, well-performing ASes will seek to provide *Debuglet* services, as they will provide three important benefits: (1) they will help validate the performance of that AS' network, (2) they will help the AS' customers verify the actual bottlenecks and improve customer support, and (3) they will obtain additional revenue from running *Debuglet*s.

6.6.3 *Economic Impact*

Debuglet applications, which are generally short-lived and require relatively little resources, should be fairly inexpensive to run. Considering cloud computing prices as a baseline, running a *Debuglet* can cost less than 100 millicents. When *Debuglet* application demand is low, deploying ISPs can either deploy virtual machines on computers already running for other purposes, such as network management, or by deploying a low-cost and low-power Single Board Computer, such as a Raspberry Pi 4, which draws between 4–6 W of power, or less than 53 kWh each year.

The low cost, both of deploying *Debuglet* services, and of executing *Debuglet* applications, can generate significant value for ISPs. For example, a Customer Premises Equipment (CPE) that can interact with *Debuglet*s in firmware can help technical support isolate a customer's connectivity or performance issues. For example, the system may determine that the issues are entirely contained within the home network, where support personnel can suggest resetting the WiFi router, or plugging directly into the CPE. In other cases, the ISP may be able to determine that the network issues arise outside of the ISP's network, for example on the downstream path toward a particular service. Because technical support staffing is a significant cost to ISPs, even minor productivity improvements to their workflow

can provide significant Return on Investment for amounts spent on deploying *Debuglet* services or for executing *Debuglet* applications.

6.6.4 *ISPs' Effort to Hide their Network Faults*

rISPs may aim to hide faults from *Debuglet* measurements to protect their reputation and revenue, e.g., refund claims for SLA violations. They may attempt this by prioritizing packets to/from *Debuglet* executors or manipulating measurement results pre-submission to the blockchain. We argue that both cases are either impractical or costly. In the first case, tracking all the executors' addresses and prioritizing their packets in forwarding devices is challenging. A feasible attack is to prioritize packets from the prefixes of interest, but can be easily cross-validated by running measurements from diverse network vantage points. Similarly, the second case can also be easily detected by running multiple measurements and cross-checking the results.

6.6.5 *Age of Information*

For immediate network diagnostics, results that are more than a few seconds old may no longer be of use. However, historical information can still be useful in some cases. For example, given multiple measurements a common network diagnostic (e.g., latency and loss rate of TCP packets) over a fixed path, the trend in measured results over time might help identify the time at which the path started experiencing performance degradation, as well as the possible location of that degradation. Thus, it would be beneficial for several applications if network measurements were retained for a period between a week and several months. However, such archiving does not necessarily need to be kept on-chain; instead, blockchain explorers or network information monitoring sites could retain measurements that are still potentially useful, and the hash of measurements would be stored on the chain for verifiability purposes.

6.6.6 *Alternative Executor Locations*

We now review other potential scenarios of executor deployment that ASes can consider.

Arbitrary Locations in each AS Deploying *Debuglet* executors at arbitrary locations within each AS introduces challenges in fault localization accuracy. In scenarios where executors are not necessarily on the original data forwarding path, false positives and negatives can occur. Furthermore, arbitrarily located executors struggle to provide fine-grained fault localization between consecutive ASes, unable to distinguish network failures in the preceding AS, the inter-domain link, or the succeeding AS. This lack of precision may render measurements inconclusive. Additionally, deploying executors within a network poses a security risk, potentially using *Debuglet* infrastructure for unveiling internal topology, routing policy, and traffic engineering rules.

Co-location with Every Router in the AS Deploying at least one executor on each router within an AS offers potential advantages in minimizing false positives and negatives compared to the arbitrary model. Here, end domains can strategically select an executor on the forwarding path for measurements, enhancing the accuracy of fault localization. However, this model comes with drawbacks. First, it entails a substantial violation of ASes privacy, demanding the explicit disclosure of internal topology and routing policies. Additionally, it provides fault information at a granularity finer than necessary, pinpointing vulnerable points and bottlenecks with high accuracy, thereby exposing the AS to targeted attacks. Moreover, it demands extensive resource requirement, rendering it impractical for real-world implementation.

6.7 RELATED WORK

In-band network telemetry (INT) [134] is a promising and innovative network telemetry approach to monitor network performance and localize faults with high accuracy and in real-time. INT accomplishes this goal by using packets of the original data flow to collect information about network performance. This is done by every router on the forwarding path, adding telemetry information to the header of every data packet. The added information can include but is not limited to timestamps of when the packet arrives at a router, when it leaves a router, and what the queue length and queue waiting times are. The last router on the path collects this information and sends it to a centralized controller for further analysis. Nonetheless, INT is typically applied for *intra-domain* fault localization. Applying INT to public inter-domain scenarios would face challenges: (1) Dependency on the destination domain's support for collecting and forwarding measurements to a (remote) central controller, (2) Potential packet prioritization by domains with INT

headers to hide their faults, and (3) Lack of control by on-path ASes over the rate of INT which leads to increased load during network faults.

RIPE Atlas [174] is a vast Internet measurement infrastructure with probes distributed across the globe, offering real-time insights into Internet dynamics. Users earn points by deploying and running probes, and use the points to conduct network measurements. Besides a default set of pre-programmed measurements, users can also schedule customized measurements. Although *Debuglet* shares the foundational principle of running customized measurements from various vantage points, it seeks different goals. While RIPE Atlas aims for a comprehensive understanding of the Internet, *Debuglet* focuses on providing domains an efficient tool for pinpointing inter-domain network faults. *Debuglet* deploys executors at domain borders, facilitating fine-grained fault localization, diverging from (often) home-based deployment of RIPE Atlas probes. Nonetheless, through running measurements between the RIPE Atlas probes and the *Debuglet* executors, both architectures can be mutually beneficial.

WTF [175] is a status-reporting and fault-localization mechanism that collects health bits and uses them to localize faults using heuristics or machine learning. WTF can combine with a tracing mechanism that gathers relevant health-bit histories for fault-localization. *Debuglet* differs in that they do not rely on other applications (or other instances of the same application) to reliably report health status; instead, they use *active measurement* over sub-paths to establish ground-truth about network conditions.

NetQuery [176] takes an AS' knowledge and measurements and makes them available to applications through a *sanitizer* that removes sensitive topology information. NetQuery's measurements are taken using Trusted Computing, which prevents an AS from sending false measurements to applications. *Debuglet* can adopt NetQuery's sanitizer, but differs in that they allow remote applications to conduct their own measurements, constructing arbitrary probing packets, while incentivizing deployment through micropayments.

iPlane [177] uses an overlay network built on top of PlanetLab to perform periodic distributed measurements of loss rate, capacity, and available bandwidth, and from these results predict network performance. By contrast, *Debuglet* is deployed inside the network, allowing for direct measurements of path components rather than inferring them from overlay measurements. *Debuglet* also provides a different, externally-verifiable mechanism for distributing the results of these measurements.

Network monitoring and debugging has been explored in numerous previous papers, including systems that use information from end-hosts [178, 131, 179],

and information from the infrastructure [180–182]. Many of these systems include localization schemes that would work in a synergistic manner with *Debuglet*.

6.8 SUMMARY

Because different types of traffic experience the network differently, we argue for a distributed network debugging infrastructure, *Debuglet*, to address the shortcomings of purely end-to-end debugging systems. By compensating ASes for hosting *Debuglet* executors in their network, we remove many usage-based attack vectors, and by retaining results within a blockchain, we allow users to prove their performance to third parties and allow users to explore performance trends. Because our design is based on smart contracts, both payment and result logging can be enforced through code rather than trust. *Debuglet* does not aim to replace existing network debugging infrastructure; rather, build on previously proposed mechanisms and provide data points that were so far difficult to obtain. These data points can be combined through the use of fault localization algorithms to provide a more fine-grained and robust view of the network.

CONCLUSION

7

CONCLUSION

Motivated by the promise of scalable massive multi-path forwarding in SCION, this thesis builds the foundations for a competitive market for wide-area network paths. Namely, a system to provide optimal paths according to the criteria of endpoints, systems to provide path performance information, and a system allowing fine-grained examination of path performance.

IREC, i.e., inter-domain routing with extensible criteria, is our solution for finding paths tailored to the criteria of endpoints. The main promises of IREC are the real-time extensibility of routing criteria and optimizing paths according to the criteria of end domains without standardization, vendor implementation, or human involvement. To achieve this goal, we propose an architecture in which multiple route selection operations can co-exist in parallel in each AS, where new operations can be added, or old ones can be removed without interrupting the others. What allows us to design such a scalable and extensible architecture is the packet-carried forwarding state in SCION that ensures the stability of the data plane during the changes introduced to the control plane. Building on parallel selection operations, IREC allows end domains to announce the desired route selection operations for paths pertaining to them as programs encoded within the routing messages they originate. Every other AS receiving such messages automatically deploys and executes these programs on the fly, eliminating the need for standardization or human involvement. What incentivizes ISPs to deploy such programs is attracting traffic and increasing their revenue: If the ISP is on the optimal path according to the criteria of a selection program, executing the program reveals such a path to endpoints and increases the chance of it being selected, increasing the chance of receiving more traffic by the ISP.

CIRo and GLIDS are two systems we propose to disseminate information about the carbon intensity and latency of inter-domain paths throughout the network, respectively. The main challenge in designing both systems and possibly in systems

for other performance metrics is reconciling the scalability of the system with the volatility of performance information. In CIRo, i.e., carbon-aware inter-domain routing, we address this issue by providing endpoints with the 24-hour prediction of the carbon intensity of paths calculated by the model we develop in this thesis. In GLIDS, i.e., global latency information dissemination system, we address this issue by disseminating the propagation delay, which is a relatively stable performance metric.

Using such information, IREC can optimize paths, and endpoints can select paths accordingly: GLIDS enables first-packet latency estimation without probing for short flows and efficient probing for long flows. Combined with bandwidth reservation systems, GLIDS provides a more accurate estimate of experienced latency due to the constant queuing delay. CIRo not only allows end domains to reduce their carbon footprint but also incentivizes ISPs to use more renewable energy sources to attract more traffic.

Finally, Debuglet enables the examination of ISPs' performance and fine-grained fault localization in a path-aware network. This is achieved by allowing interested entities to perform customized measurements between the borders of any two ASes on an inter-domain path. The fundamental principles of Debuglet are segment-by-segment measurements, i.e., allowing the measurement of any subpath of a path individually, and reproducibility, i.e., allowing the reproduction of faults through generating traffic-like probe packets. These principles are realized by deploying debugging nodes near AS borders that accept and execute debugging WebAssembly applications from interested entities. Debuglet ensures the integrity and transferability of the results by using a blockchain-based smart contract as the control plane that also publishes the results. With Debuglet, endpoints can detect and avoid faulty segments on inter-domain paths. Therefore, faulty ASes lose revenue and thus are held accountable for their poor performance. This incentivizes ISPs to innovate and provide high-quality connectivity to attract more traffic and increase their revenue.

8

FUTURE WORK

This thesis provides the foundation for future research in creating a marketplace for inter-domain paths. In addition, each system introduced in this thesis opens new research directions, which we elaborate in this chapter.

8.1 IREC

Development of Route Selection Programs IREC opens up a large research space for the design of selection operation programs, studying their effect on the message complexity and resource usage of the control plane, and tuning them for path optimization and resource usage at the same time.

Verification of Route Selection Programs For enhanced security, SPCs can perform (formal) verification of selection programs specified by end domains and ensure that they satisfy certain security and resource use properties. This is an interesting future research direction for IREC.

End-to-end Performance Optimization Even with route computation per interface groups, end-to-end optimality is not guaranteed unless endpoints know which interface group is suitable for reaching the destination endpoint. This is because PCBs do not carry the necessary information for every destination in the AS due to the possibly large communication overhead. Instead, for measurable metrics such as latency or bandwidth, the client can discover the optimal end-to-end path simply by connecting to the destination over paths traversing interfaces belonging to different interface groups. However, this approach causes a significant delay in finding the optimal end-to-end path. In the future, the DNS information could be augmented to provide information about performance metrics of paths between interface groups and the destination.

8.2 PATH INFORMATION DISSEMINATION

8.2.1 *CIRo*

Real-time Carbon Accounting By providing endpoints with predictions of paths' carbon intensity, CIRo takes an essential step towards carbon transparency on the public Internet and helps end domains with their carbon footprint accounting. However, to achieve full carbon transparency and accurate accounting, real-time information about paths' carbon intensity should be distributed globally. This requires a complementary system to work in parallel with CIRo. Designing such a system has its own specific challenges, tackling which is out of the scope of the present thesis.

Trustworthy Carbon Information Green path selection by endpoints could tempt ASes to claim false carbon-intensity information to attract more traffic. Hence, carbon trust roots are needed to certify carbon-intensity information claimed by ISPs. The main challenges for these trust roots are: to estimate the carbon intensity of paths within ISPs, to ensure that ISPs do not deviate from their certified carbon intensity after obtaining a certificate, and to periodically re-certify carbon intensities. This is a promising future research direction.

Green Data Centers and CDNs A path-aware Internet architecture can also enable endpoints to select the data center providing a service or content [108]. Therefore, providing carbon-intensity information of data centers in combination with information about communication paths enables endpoints to monitor and optimize the *total* carbon footprint of their requests, providing CO_2 optimization for computation-intensive *and* communication-intensive applications.

8.2.2 *GLIDS*

Intra-domain Latency Information and Path Selection Multiple *intra-domain* paths with different propagation latencies can exist between every pair of inter-AS interfaces of an AS. With GLIDS, ASes disseminate the minimum propagation latency and, optionally, the maximum propagation latency. GLIDS can be extended to provide the propagation latency of all *intra-domain* paths between a pair of interfaces. However, this information has limited use cases as the endpoint does not have knowledge nor control over what intra-AS path is used for each packet. This is because the AS border router, not the endpoint, selects the *intra-domain* forwarding

path for a packet traveling from one interface to the other interface of an AS. The latency information about all intra-domain paths can be useful if endpoints can select the *intra-domain* path at each AS hop (in addition to inter-domain path selection in SCION). This idea has been explored for another path property, i.e., the type of network devices on a path and their software version [141]. In this approach, the multiprotocol label switching (MPLS) labels of internal paths are advertised together with the path properties. The endpoints can then choose the internal path and embed the MPLS label for each AS hop in the packet header. The border routers of each AS forward the inter-domain packets on internal paths with the specified MPLS label. The same approach can be used for propagation latency: GLIDS can be extended to provide propagation latency for every internal path together with the associated MPLS label.

8.3 DEBUGLET

Wise Selection of Debuglet Executions by Initiators Fast and efficient fault localization in *Debuglet* demands a strategic selection of executors. This choice profoundly impacts time-to-locate, executor load, and measurement costs. For example, consider a path over 10 consecutive ASes with a fault in the last inter-domain link. Running a series of consecutive measurements from the first to the last AS can impose a long time-to-locate and high cost. Simultaneously examining all the links may not address cost concerns.

In a general case without prior knowledge regarding the fault, a binary search offers a cost- and time-effective solution. Initial measurements from the path's midpoint to each end domain may reveal the faulty half, repeating until all faults are located. Nevertheless, the responsibility lies with initiators to choose the selection strategy, relying on educated initial guesses, historical data, and potentially leveraging machine learning for informed decisions. We envision interesting research to devise smart approaches for different circumstances and leave them for future work.

Secondary Web-based Control Plane Parallel to Blockchain Smart Contract The smart contract control plane for Debuglet allows transferability and ensures the integrity of Debuglet results. However, it requires the initiators to wait for the finality of transactions before being able to execute the measurements. Many initiators, however, are interested in instant and short measurements. To accommodate such a need, a parallel web-based control plane can be developed in the future to

coordinate these measurements. The main challenge is to orchestrate both control planes such that they do not conflict with each other.

DOCTORAL RESEARCH (COVERED)

[1] Seyedali Tabaeiaghdaei, Marc Wyss, Giacomo Giuliari, Jelte van Bommel, Ahad N. Zehmakan, and Adrian Perrig. *Inter-Domain Routing with Extensible Criteria*. 2023. arXiv: 2309.03551 [cs.NI].

[2] Seyedali Tabaeiaghdaei, Simon Scherrer, Jonghoon Kwon, and Adrian Perrig. "Carbon-Aware Global Routing in Path-Aware Networks." In: *Proceedings of ACM International Conference on Future Energy Systems (e-Energy)*. 2023.

[3] Cyrill Krähenbühl, Seyedali Tabaeiaghdaei, Simon Scherrer, and Adrian Perrig. "GLIDS: Toward Global Latency Transparency." In: *Proceedings of the IFIP Networking Conference*. 2024.

[4] Seyedali Tabaeiaghdaei, Filippo Costa, Jonghoon Kwon, Patrick Bamert, Yih-Chun Hu, and Adrian Perrig. "Debuglet: Programmable and Verifiable Inter-domain Network Telemetry." In: *Proceedings of the IEEE International Conference on Distributed Computing Systems (ICDCS)*. 2024.

DOCTORAL RESEARCH (NOT COVERED)

[5] Simon Scherrer, Seyedali Tabaeiaghdaei, and Adrian Perrig. "Quality Competition Among Internet Service Providers." In: *Proceedings of the IFIP International Symposium on Computer Performance, Modeling, Measurements and Evaluation (PERFORMANCE)*. 2023.

[6] Marc Frei, Jonghoon Kwon, Seyedali Tabaeiaghdaei, Marc Wyss, Christoph Lenzen, and Adrian Perrig. "G-SINC: Global Synchronization Infrastructure for Network Clocks." In: *Proceedings of the IEEE Symposium on Reliable Distributed Systems (SRDS)*. 2022.

[7] Adrian Perrig Seyedali Tabaeiaghdaei. "Data-Plane Energy Efficiency of a Next-Generation Internet Architecture." In: *Proceedings of the Symposium on Computers and Communications (ISCC)*. 2022.

[8] Cyrill Krähenbühl, Seyedali Tabaeiaghdaei, Christelle Gloor, Jonghoon Kwon, Adrian Perrig, David Hausheer, and Dominik Roos. "Deployment and Scalability of an Inter-Domain Multi-Path Routing Infrastructure." In: *Proceedings of the ACM SIGCOMM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. 2021.

REFERENCES

[9] *Path Aware Networking RG (panrg) Internet Drafts, year=2021*. https://www.potaroo.net/ietf/html/ids-wg-panrg.html archived at https://perma.cc/2RME-WVST. (Visited on 04/19/2021).

[10] Laurent Chuat, Markus Legner, David Basin, David Hausheer, Samuel Hitz, Peter Müller, and Adrian Perrig. *The Complete Guide to SCION*. Springer, 2022.

[11] SIX. *Secure Swiss Finance Network (SSFN) For secure, flexible and resilient data communication*. https://www.six-group.com/en/products-services/banking-services/ssfn.html archived at https://perma.cc/5KGL-S54B. (Visited on 05/03/2024).

[12] Swisscom. *SCION – for secure data transmission*. https://www.swisscom.ch/en/business/enterprise/offer/wireline/scion.html archived at https://perma.cc/8MDB-62PS. (Visited on 05/03/2024).

[13] GÉANT. *Infoshare: SCION Access for Universities and Research Institutes*. https://connect.geant.org/2022/11/18/infoshare-scion-access-for-universities-and-research-institutes-24-nov-2022 archived at https://perma.cc/PE4K-S36K. (Visited on 05/03/2024).

[14] Intercloud. *Securing cloud connectivity with SCION*. https://www.intercloud.com/our-resources/blog/securing-cloud-connectivity-with-scion archived at https://perma.cc/CYX4-2H25. (Visited on 05/03/2024).

[15] Telindus. *SCiON A New Architecture that Overcomes the Limitations of the Internet*. https://www.proximusnxt.lu/en/scion archived at https://perma.cc/5QQK-J2GP. (Visited on 05/03/2024).

[16] cyberlink. *SCION-Internet - Hochsicherer Datenaustausch*. https://www.cyberlink.ch/scion?gad_source=1&gclid=Cj0KCQjwltKxBhDMARIsAG8KnqUWVp-1a7Gne3HUTk0Qwu92oLExYa1wBrDfWNFGB0VM32pUmANHskkaArPhEALw_wcB archived at https://perma.cc/G7XC-ZBKK. (Visited on 05/03/2024).

[17] Karrierone. *Building the future of connectivity*. https://scion.karrier.one archived at https://perma.cc/UH8S-DBZM. (Visited on 01/03/2025).

[18] Xiaowei Yang, David Clark, and Arthur W. Berger. "NIRA: A New Inter-Domain Routing Architecture." In: *IEEE/ACM Transactions on Networking* (2007).

[19] Simon Scherrer, Markus Legner, Adrian Perrig, and Stefan Schmid. "Incentivizing stable path selection in future internet architectures." In: *Performance Evaluation* (2020).

[20] Thomas Wirtgen, Tom Rousseaux, Quentin De Coninck, Nicolas Rybowski, Randy Bush, Laurent Vanbever, Axel Legay, and Olivier Bonaventure. "xBGP: Faster Innovation in Routing Protocols." In: *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 2023.

[21] *eBPF-Dynamically program the kernel for efficient networking, observability, tracing, and security*. https://ebpf.io archived at https://perma.cc/7TSN-89WR. (Visited on 11/18/2024).

[22] *Userspace eBPF VM - Github.com*. https://github.com/iovisor/ubpf archived at https://perma.cc/6TVX-7AA3. (Visited on 11/18/2024).

[23] Mario Hock, Roland Bless, and Martina Zitterbart. "Experimental evaluation of BBR congestion control." In: *Proceedings of the IEEE Conference on Network Protocols (ICNP)*. 2017.

[24] Ayush Mishra, Wee Han Tiu, and Ben Leong. "Are we heading towards a BBR-dominant Internet?" In: *Proceedings of the ACM Internet Measurement Conference (IMC)*. 2022.

[25] *WebAssembly*. https://webassembly.org archived at https://perma.cc/DWT5-EMJW. (Visited on 01/09/2024).

[26] University of Washington NS-3 Consortium. *ns-3 Network Simulator*. https://www.nsnam.org/ archived at https://perma.cc/4S4N-8VGV. (Visited on 04/19/2021).

[27] João Luís Sobrinho. "Algebra and Algorithms for QoS Path Computation and Hop-by-Hop Routing in the Internet." In: *IEEE/ACM Transactions on Networking* (2002).

[28] K Oanh Ha. *When Your Boss Becomes a Hologram*. https://www.bloomberg.com/news/articles/2022-03-03/big-tech-and-startups-look-to-3d-hologram-for-travel-free-communication archived at https://perma.cc/5KXV-LMUX. (Visited on 05/26/2022).

[29] Bin Da and Marco Carugi. *Representative use cases and key network requirements for Network 2030*. Tech. rep. ITU-T, Jan. 2020.

[30] Henry Birge-Lee, Sophia Yoo, Benjamin Herber, Jennifer Rexford, and Maria Apostolaki. "TANGO: Secure Collaborative Route Control across the Public Internet." In: *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 2024.

[31] Igor Ganichev, Bin Dai, P. Brighten Godfrey, and Scott Shenker. "YAMR: Yet Another Multipath Routing Protocol." In: *ACM SIGCOMM Computer Communication Review (CCR)* (2010).

[32] Donghong Qin, Jiahai Yang, Zhuolin Liu, Jessie Wang, Bin Zhang, and Wei Zhang. "AMIR: Another Multipath Interdomain Routing." In: *Proceedings of the IEEE International Conference on Advanced Information Networking and Applications (AINA)*. 2012.

[33] Yong Liao, Lixin Gao, Roch Guerin, and Zhi-Li Zhang. "Reliable Interdomain Routing through Multiple Complementary Routing Processes." In: *Proceedings of the ACM SIGCOMM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. 2008.

[34] Ming Zhu, Dan Li, Ying Liu, Dan Pei, KK Ramakrishnan, Lili Liu, and Jianping Wu. "MIFO: Multi-path Interdomain Forwarding." In: *Proceedings of the ACM International Conference on Parallel Processing (ICPP)*. 2015.

[35] Xia Yin, Dan Wu, Zhiliang Wang, Xingang Shi, and Jianping Wu. "DIMR." In: *Computer Networks* (2015).

[36] Feng Wang and Lixin Gao. "Path Diversity Aware Interdomain Routing." In: *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*. 2009.

[37] Nate Kushman, Srikanth Kandula, Dina Katabi, and Bruce M. Maggs. "R-BGP: Staying Connected In a Connected World." In: *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*. 2007.

[38] Murtaza Motiwala, Megan Elmore, Nick Feamster, and Santosh Vempala. "Path Splicing." In: *Proceedings of the ACM SIGCOMM Conference*. 2008.

[39] Center for Applied Internet Data Analysis (CAIDA). *AS-Relationships*. https://www.caida.org/data/as-relationships/ archived at https://perma.cc/RYN2-BSNB. (Visited on 04/19/2021).

[40] João Luís Sobrinho and Miguel Alves Ferreira. "Routing on Multiple Optimality Criteria." In: *Proceedings of the ACM SIGCOMM Conference*. 2020.

[41] *Juniper vs Cisco BGP Med Attribute*. https://skminhaj.wordpress.com/2014/12/23/juniper-vs-cisco-bgp-med-attribute/ archived at https://perma.cc/4STV-P6DW. (Visited on 12/18/2024).

[42] *Scalability, Control and Isolation On next-generation Networks (SCION)*. https://github.com/scionproto/scion. (Visited on 12/14/2024).

[43] Gaetano Bonofiglio, Veronica Iovinella, Gabriele Lospoto, and Giuseppe Di Battista. "Kathará: A container-based framework for implementing network function virtualization and software defined networks." In: *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS)*. 2018.

[44] Egbert Harzheim. *Ordered Sets*. Springer, 2005.

[45] *Kathará*. https://www.kathara.org/index.html archived at https://perma.cc/KUE4-Q6AA. (Visited on 11/18/2024).

[46] *docker*. https://www.docker.com/ archived at https://perma.cc/4APP-FENR. (Visited on 11/18/2024).

[47] *gRPC*. https://grpc.io/ archived at https://perma.cc/G8C5-VBCT. (Visited on 12/16/2024).

[48] *SQLite*. https://www.sqlite.org/index.html archived at https://perma.cc/34PT-R3EC. (Visited on 12/16/2024).

[49] *SCION Control Service Documentation*. https://docs.scion.org/en/latest/manuals/control.html archived at https://perma.cc/M8ZB-U4AD. (Visited on 12/16/2024).

[50] *libxBGP*. https://github.com/pluginized-protocols/libxbgp archived at https://perma.cc/9F8N-BB4D. (Visited on 11/18/2024).

[51] *Cgo Documentation*. https://pkg.go.dev/cmd/cgo archived at https://perma.cc/5N5A-2UND. (Visited on 11/18/2024).

[52] *FlatBuffers: Memory Efficient Serialization Library*. https://flatbuffers.dev/ archived at https://perma.cc/MC9D-DNVQ. (Visited on 11/18/2024).

[53] Center for Applied Internet Data Analysis (CAIDA). *AS Relationships – with Geographic Annotations*. https://www.caida.org/data/as-relationships-geo/ archived at https://perma.cc/S4XX-Y2EK. (Visited on 04/19/2021).

[54] Mariano Scazzariello, Lorenzo Ariemma, Giuseppe Di Battista, and Maurizio Patrignani. "Megalos: A Scalable Architecture for the Virtualization of Large Network Scenarios." In: *Future Internet* (2021).

[55] *Kubernetes*. (Visited on 11/19/2024).

[56] *Prometheus*. (Visited on 11/19/2024).

[57] David Naylor et al. "XIA: Architecting a More Trustworthy and Evolvable Internet." In: *SIGCOMM Computer Communication Reviews (CCR)* (2014).

[58] James McCauley, Yotam Harchol, Aurojit Panda, Barath Raghavan, and Scott Shenker. "Enabling a Permanent Revolution in Internet Architecture." In: *Proceedings of the ACM SIGCOMM Conference*. 2019.

[59] J. Brumbaugh-Smith and D. Shier. "An Empirical Investigation of some Bicriterion Shortest Path Algorithms." In: *European Journal of Operational Research* (1989).

[60] Pierre Hansen. "Bicriterion Path Problems." In: *Proceedings of the Multiple Criteria Decision Making Theory and Application*. 1980.

[61] Ernesto Queirós Vieira Martins. "On a Multicriteria Shortest Path Problem." In: *European Journal of Operational Research* (1984).

[62] Mark Yampolskiy, Wolfgang Hommel, Bernhard Lichtinger, Wolfgang Fritz, and Matthias K. Hamm. "Multi-domain End-to-End (E2E) Routing with Multiple QoS Parameters – Considering Real World User Requirements and Service Provider Constraints." In: *Proceedings of the IARIA International Conference on Evolving Internet (INTERNET)*. 2010.

[63] Ignacio Castro, Aurojit Panda, Barath Raghavan, Scott Shenker, and Sergey Gorinsky. "Route Bazaar: Automatic Interdomain Contract Negotiation." In: *Proceedings of the USENIX Conference on Hot Topics in Operating Systems (HOTOS)*. 2015.

[64] Josip Lorincz, Antonio Capone, and Jinsong Wu. "Greener, Energy-Efficient and Sustainable Networks: State-Of-The-Art and New Trends." In: *Sensors* (Nov. 2019).

[65] Anders S. G. Andrae and Tomas Edler. "On Global Electricity Usage of Communication Technology: Trends to 2030." In: *Challenges* (2015).

[66] Noa Zilberman, Eve M Schooler, Uri Cummings, Rajit Manohar, Dawn Nafus, Robert Soulé, and Rick Taylor. "Toward Carbon-Aware Networking." In: *Proceedings of the Workshop on Sustainable Computer Systems Design and Implementation*. 2022.

[67] Alireza Nafarieh, Yashar Fazili, and William Robertson. "Dynamic Interdomain Negotiation for Green Algorithms in Optical Networks." In: *Procedia Computer Science* (2013).

[68] M. Lepinski and K. Sriram. *BGPsec Protocol Specification*. RFC 8205. IETF, Sept. 2017. URL: http://tools.ietf.org/rfc/rfc8205.txt.

[69] A. Clemm, L. Dong, G. Mirsky, L. Ciavaglia, J. Tantsura, and M-P. Odini. *Green Networking Metrics*. https://datatracker.ietf.org/doc/draft-cx-green-metrics/ archived at https://perma.cc/K6WQ-R6VX. July 2022. (Visited on 12/16/2024).

[70] Electricity Maps. *Electricity Maps*. https://www.electricitymaps.com archived at https://perma.cc/S4ND-PRAZ. (Visited on 11/24/2022).

[71] WattTime. *WattTime*. https://www.watttime.org archived at https://perma.cc/B63V-8D9L. (Visited on 11/24/2022).

[72] Diptyaroop Maji, Ramesh K. Sitaraman, and Prashant Shenoy. "DACF: Day-Ahead Carbon Intensity Forecasting of Power Grids Using Machine Learning." In: *Proceedings of ACM International Conference on Future Energy Systems (e-Energy)*. 2022.

[73] Arsalan Ahmad, Andrea Bianco, Edoardo Bonetto, Luca Chiaraviglio, and Filip Idzikowski. "Energy-Aware Design of Multilayer Core Networks." In: *Optical Communications and Networking* (Oct. 2013).

[74] Ward Heddeghem, Filip Idzikowski, Willem Vereecken, Didier Colle, Mario Pickavet, and Piet Demeester. "Power Consumption Modeling in Optical Multilayer Networks." In: *Photonic Network Communications* (Oct. 2012).

[75] C. Belady and Andy Rawson. "Green Grid Date Center Power Efficiency Metrics: PUE and DCIE." In: 2008.

[76] Kerry Hinton, Fatemeh Jalali, and Ashrar Matin. "Energy Consumption Modeling of Optical Networks." In: *Photonic Network Communications* (Aug. 2015).

[77] Tabaeiaghdaei, Seyedali. *CIRo*. https://github.com/SeyedaliTaba/CIRo. Jan. 2023.

[78] Geoff Huston. *BGP in 2020 – The BGP Table*. https://blog.apnic.net/2021/01/05/bgp-in-2020-the-bgp-table/ archived at https://perma.cc/9U97-98UN. (Visited on 04/19/2021).

[79] Edsger W Dijkstra. "A note on two problems in connexion with graphs." In: *Numerische mathematik* (1959).

[80] Center for Applied Internet Data Analysis (CAIDA). *Macroscopic Internet Topology Data Kit (ITDK)*. https://www.caida.org/data/internet-topology-data-kit/ archived at https://perma.cc/CU8X-7GRU. (Visited on 04/19/2021).

[81] International Energy Agency (IEA). *Data & Statistics*. https://www.iea.org/data-and-statistics?country=WORLD&fuel=Electricity%20and%20heat&indicator=ElecGenByFuel archived at https://perma.cc/72QW-26D9. (Visited on 04/19/2021).

[82] Ottmar Edenhofer. *Renewable Energy Sources and Climate Change Mitigation: Special Report of the Intergovernmental Panel on Climate Change*. Intergovernmental Panel on Climate Change (IPCC), 2012.

[83] Laurent Vanbever. *dragon_simulator*. https://github.com/network-aggregation/dragon_simulator archived at https://perma.cc/4QCZ-7JWU. (Visited on 04/19/2021).

[84] J. Mikians, N. Laoutaris, A. Dhamdhere, and P. Barlet-Ros. "ITMgen — A First-Principles Approach to Generating Synthetic Interdomain Traffic Matrices." In: *Proceedings of the IEEE International Conference on Communications (ICC)*. 2013.

[85] Center for Applied Internet Data Analysis (CAIDA). *Routeviews Prefix to AS Mappings Dataset (pfx2as) for IPv4 and IPv6*. https://www.caida.org/data/routing/routeviews-prefix2as.xml archived at https://perma.cc/RB6L-PEVA. (Visited on 04/19/2021).

[86] George Kingsley Zipf. *Human Behavior and the Principle of Least Effort: An Introduction to Human Ecology*. Addison-Wesley Press, Inc, 1949.

[87] Victor Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. *Tranco*. https://tranco-list.eu archived at https://perma.cc/PK5Q-Z2W5. (Visited on 01/11/2022).

[88] Statista. *Consumer Internet Data Traffic Worldwide by Application Category from 2016 to 2022 (in Eb per Month)*. https://www.statista.com/statistics/454951/mobile-data-traffic-worldwide-by-application-category/ archived at https://perma.cc/XLH7-J8W3. (Visited on 04/19/2021).

[89] Sandvine. *The Global Internet Phenomena Report*. https://www.sandvine.com/hubfs/downloads/phenomena/2018-phenomena-report.pdf archived at https://perma.cc/SY2E-C7N8. (Visited on 04/19/2021).

[90] Ankit Singla, Balakrishnan Chandrasekaran, P. Brighten Godfrey, and Bruce Maggs. "The Internet at the Speed of Light." In: *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*. 2014.

[91] Simon Fischer and Berthold Vöcking. "Adaptive routing with stale information." In: *Theoretical Computer Science* (2009).

[92] Frank Kelly and Thomas Voice. "Stability of end-to-end algorithms for joint routing and rate control." In: *ACM SIGCOMM Computer Communication Review (CCR)* (2005).

[93] Seyedali Tabaeiaghdaei and Adrian Perrig. "Data-Plane Energy Efficiency of a Next-Generation Internet Architecture." In: *Proceedings of the Symposium on Computers and Communications (ISCC)*. 2022.

[94] M. Zhang, C. Yi, B. Liu, and B. Zhang. "GreenTE: Power-Aware Traffic Engineering." In: *Proceedings of the IEEE International Conference on Network Protocols*. 2010.

[95] J. Shi and B. Zhang. "Making Inter-Domain Routing Power-Aware?" In: *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*. 2014.

[96] Thilo Schöndienst and Vinod Vokkarane. "Renewable Energy-Aware Grooming in Optical Networks." In: *Photonic Network Communications* (Aug. 2014).

[97] Chao Jin, Bronis R. de Supinski, David Abramson, Heidi Poxon, Luiz DeRose, Minh Ngoc Dinh, Mark Endrei, and Elizabeth R. Jessup. "A survey on software methods to improve the energy efficiency of parallel computing." In: *International Journal of High Perform. Computing Applications* (2017).

[98] Erica Sousa, Fernando Lins, Eduardo Tavares, Paulo Cunha, and Paulo Maciel. "A Modeling Approach for Cloud Infrastructure Planning Considering Dependability and Cost Requirements." In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2015).

[99] Saurabh Kumar Garg and Rajkumar Buyya. "Green cloud computing and environmental sustainability." In: *Harnessing Green IT: Principles and Practices*. Wiley-IEEE Computer Society Press, 2012.

[100] Xiaoying Wang, Guojing Zhang, Mengqin Yang, and Lei Zhang. "Green-Aware Virtual Machine Migration Strategy in Sustainable Cloud Computing Environments." In: *Cloud Computing - Architecture and Applications*. IntechOpen, 2017.

[101] Weixiang Jiang, Ziyang Jia, Sirui Feng, Fangming Liu, and Hai Jin. "Fine-Grained Warm Water Cooling for Improving Datacenter Economy." In: *Proceedings of the ACM/IEEE International Symposium on Computer Architecture (ISCA)*. 2019.

[102] Baris Aksanli, Tajana Simunic Rosing, and Inder Monga. "Benefits of green energy and proportionality in high speed wide area networks connecting data centers." In: *Proceedings of the Automation & Test in Europe Conference & Exhibition (DATE)*. 2012.

[103] S. Ricciardi, D. Careglio, F. Palmieri, U. Fiore, G. Santos-Boada, and J. Sole-Pareta. "Energy-Aware RWA for WDM Networks with Dual Power Sources." In: *Proceedings of the IEEE International Conference on Communications (ICC)*. 2011.

[104] Karel van der Veldt, Cees de Laat, Inder Monga, Jon Dugan, and Paola Grosso. "Carbon-aware path provisioning for NRENs." In: *Proceedings of the ACM International Green and Sustainable Computing Conference (IGSC)*. 2014.

[105] M. Gattulli, M. Tornatore, R. Fiandra, and A. Pattavina. "Low-Carbon Routing Algorithms for Cloud Computing Services in IP-over-WDM Networks." In: *Proceedings of the IEEE International Conference on Communications (ICC)*. 2012.

[106] Syed Raza, Isam Janajreh, and Chaouki Ghenai. "Sustainability index approach as a selection criteria for energy storage system of an intermittent renewable energy source." In: *Applied Energy* (Dec. 2014).

[107] Sukhpal Singh Gill and Rajkumar Buyya. "A Taxonomy and Future Directions for Sustainable Cloud Computing: 360 Degree View." In: *ACM Computing Surveys* (Dec. 2018).

[108] Zhenhua Liu, Minghong Lin, Adam Wierman, Steven H Low, and Lachlan LH Andrew. "Greening geographical load balancing." In: *ACM SIGMETRICS Performance Evaluation Review* (2011).

[109] Ana Radovanovic et al. "Carbon-Aware Computing for Datacenters." In: *IEEE Transactions on Power Systems* (2022).

[110] M. Gupta and S. Singh. "Greening of the Internet." In: *Proceedings of the ACM SIGCOMM Conference (SIGCOMM)*. 2003.

[111] Nedeljko Vasic, Dejan Novakovic, Satyam Shekhar, Prateek Bhurat, Marco Canini, and Dejan Kostic. "Identifying and Using Energy-Critical Paths." In: *Proceedings of the ACM SIGCOMM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. 2011.

[112] N. Vasic and D. Kostic. "Energy-Aware Traffic Engineering." In: *Proceedings of the ACM International Conference on Energy-Efficient Computing and Networking (e-Energy)*. 2010.

[113] M. Andrews, A. Anta, L. Zhang, and W. Zhao. "Routing for energy minimization in the speed scaling model." In: *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*. 2010.

[114] Joseph Chabarek, Joel Sommers, Paul Barford, Cristian Estan, David Tsiang, and Steve Wright. "Power Awareness in Network Design and Routing." In: *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*. 2008.

[115] Vlad C. Coroama, Lorenz M. Hilty, Ernst Heiri, and Frank M. Horn. "The Direct Energy Demand of Internet Data Flows." In: *Industrial Ecology* (2013).

[116] Jonathan Koomey, Huimin Chong, Woonsien Loh, Bruce Nordman, and Michele Blazek. "Network Electricity Use Associated with Wireless Personal Digital Assistants." In: *Infrastructure Systems* (2004).

[117] Cody Taylor and Jonathan Koomey. "Estimating Energy Use and Greenhouse Gas Emissions of Internet Advertising." In: (Jan. 2008).

[118] Christopher L. Weber, Jonathan G. Koomey, and H. Scott Matthews. "The Energy and Climate Change Implications of Different Music Delivery Methods." In: *Industrial Ecology* (2010).

[119] Steven Lanzisera, Bruce Nordman, and Richard E. Brown. "Data Network Equipment Energy Use and Savings Potential in Buildings." In: *Energy Efficiency* (2011).

[120] Jayant Baliga, Kerry Hinton, and Rodney Tucker. "Energy Consumption of the Internet." In: *Proceedings of the Joint International Conference on the Optical Internet and the Australian Conference on Optical Fibre (COIN-ACOFT)*. July 2007.

[121] J. Baliga, R. Ayre, K. Hinton, W. Sorin, and R. Tucker. "Energy Consumption in Optical IP Networks." In: *Lightwave Technology* (2009).

[122] A. Vishwanath, K. Hinton, R. W. A. Ayre, and R. S. Tucker. "Modeling Energy Consumption in High-Capacity Routers and Switches." In: *IEEE Journal on Selected Areas in Communications* (2014).

[123] Kerry Hinton, Jayant Baliga, Michael Feng, Robert Ayre, and Rodney Tucker. "Power Consumption and Energy Efficiency in the Internet." In: *IEEE Network* (May 2011).

[124] Aiguo Fei, Guangyu Pei, Roy Liu, and Lixia Zhang. *Measurements On Delay And Hop-Count Of The Internet*. Sept. 1998.

[125] Fatemehsadat Tabatabaeimehr, Marc Ruiz, Che-Yu Liu, Xiaoliang Chen, Roberto Proietti, S. J. Ben Yoo, and Luis Velasco. "Cooperative Learning for Disaggregated Delay Modeling in Multidomain Networks." In: *IEEE Transactions on Network and Service Management* (Sept. 2021).

[126] Daniel Perdices, David Muelas, Iria Prieto, Luis de Pedro, and Jorge E. Lopez de Vergara. "On the Modeling of Multi-Point RTT Passive Measurements for Network Delay Monitoring." In: *IEEE Transactions on Network and Service Management* (Sept. 2019).

[127] Filip Krasniqi, Jocelyne Elias, Jeremie Leguay, and Alessandro E. C. Redondi. "End-to-end Delay Prediction Based on Traffic Matrix Sampling." In: *Proceedings of the IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. July 2020.

[128] Weihe Li, Jingling Liu, Shiqi Wang, Tao Zhang, Shaojun Zou, Jinbin Hu, Wanchun Jiang, and Jiawei Huang. "Survey on Traffic Management in Data Center Network: From Link Layer to Application Layer." In: *IEEE Access* (2021).

[129] El Hocine Bouzidi, Duc-Hung Luong, Abdelkader Outtagarts, Abdelkrim Hebbar, and Rami Langar. "Online-Based Learning for Predictive Network Latency in Software-Defined Networks." In: *Proceedings of the IEEE Global Communications Conference (GLOBECOM)*. Dec. 2018.

[130] Sami Alesawi, Minh Nguyen, Hao Che, and Akshit Singhal. "Tail Latency Prediction for Datacenter Applications in Consolidated Environments." In: *Proceedings of the International Conference on Computing, Networking and Communications (ICNC)*. Feb. 2019.

[131] Chuanxiong Guo. "Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis." In: *Proceedings of the ACM SIGCOMM Conference (SIGCOMM)*. 2015.

[132] Ali Safari Khatouni, Francesca Soro, and Danilo Giordano. "A Machine Learning Application for Latency Prediction in Operational 4G Networks." In: *Proceedings of the IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. 2019.

[133] Bang Liu, Di Niu, Zongpeng Li, and H. Vicky Zhao. "Network Latency Prediction for Personal Devices: Distance-Feature Decomposition from 3D Sampling." In: *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*. Apr. 2015.

[134] Lizhuang Tan, Wei Su, Wei Zhang, Jianhui Lv, Zhenyi Zhang, Jingying Miao, Xiaoxi Liu, and Na Li. "In-band Network Telemetry: A Survey." In: *Computer Networks* (2021).

[135] Robin Sommer and Anja Feldmann. "NetFlow: Information loss or win?" In: *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*. 2002.

[136] J. Quittek, T. Zseby, B. Claise, and S. Zander. *Requirements for IP Flow Information Export (IPFIX)*. RFC 3917. IETF, Oct. 2004. URL: http://tools.ietf.org/rfc/rfc3917.txt.

[137] Bruno Astuto A. Nunes, Marc Mendonca, Xuan-Nam Nguyen, Katia Obraczka, and Thierry Turletti. "A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks." In: *IEEE Communications Surveys & Tutorials* (2014).

[138] Curtis Yu, Cristian Lumezanu, Abhishek Sharma, Qiang Xu, Guofei Jiang, and Harsha V. Madhyastha. "Software-Defined Latency Monitoring in Data Center Networks." In: *Proceedings of the Passive and Active Measurement Conference (PAM)*. 2015.

[139] Changhoon Kim, Anirudh Sivaraman, Naga Praveen Kumar Katta, Antonin Bas, Advait Abhay Dixit, and Lawrence J Wobker. *In-band Network Telemetry via Programmable Dataplanes*. 2015.

[140] G. Fioccola, A. Capello, M. Cociglio, L. Castaldelli, M. Chen, L. Zheng, G. Mirsky, and T. Mizrahi. *Alternate-Marking Method for Passive and Hybrid Performance Monitoring*. RFC 8321. IETF, Jan. 2018. URL: http://tools.ietf.org/rfc/rfc8321.txt.

[141] Cyrill Krähenbühl, Marc Wyss, David Basin, Vincent Lenders, Adrian Perrig, and Martin Strohmeier. "FABRID: Flexible Attestation-Based Routing for Inter-Domain Networks." In: *Proceedings of the USENIX Security Symposium (USENIX Security)*. Aug. 2023.

[142] Jonghoon Kwon, Juan A. García-Pardo, Markus Legner, François Wirz, Matthias Frei, David Hausheer, and Adrian Perrig. "SCIONLAB: A Next-Generation Internet Testbed." In: *Proceedings of the IEEE Conference on Network Protocols (ICNP)*. 2020.

[143] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. "BBR: Congestion-Based Congestion Control." In: *Communications of the ACM* (2017).

[144] Sangtae Ha, Injong Rhee, and Lisong Xu. "CUBIC: a new TCP-friendly high-speed TCP variant." In: *ACM SIGOPS Operating Systems Review* (2008).

[145] Bob Lantz, Nikhil Handigol, Brandon Heller, and Vimal Jeyakumar. *Introduction to Mininet*. 2021. URL: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet.

[146] Dominik Scholz, Benedikt Jaeger, Lukas Schwaighofer, Daniel Raumer, Fabien Geyer, and Georg Carle. "Towards a Deeper Understanding of TCP BBR Congestion Control." In: *Proceedings of the IEEE/IFIP Networking Conference*. 2018.

[147] D. Thaler. *Planning for Protocol Adoption and Subsequent Transitions*. RFC 8170. IETF, May 2017. URL: http://tools.ietf.org/rfc/rfc8170.txt.

[148] Giacomo Giuliari, Dominik Roos, Marc Wyss, Juan Angel García-Pardo, Markus Legner, and Adrian Perrig. "Colibri: A Cooperative Lightweight Inter-domain Bandwidth-Reservation Infrastructure." In: *Proceedings of the ACM SIGCOMM International Conference on Emerging Networking EXperiments and Technologies (CoNEXT)*. Dec. 2021.

[149] Marc Wyss, Giacomo Giuliari, Jonas Mohler, and Adrian Perrig. "Protecting Critical Inter-Domain Communication through Flyover Reservations." In: *Proceedings of the ACM Conference on Computer and Communications Security (CCS)*. Nov. 2022.

[150] P. Phaal, S. Panchen, and N. McKee. *InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks*. RFC 3176. IETF, Sept. 2001. URL: http://tools.ietf.org/rfc/rfc3176.txt.

[151] Roberto Bifulco and Gebor Retvari. "A Survey on the Programmable Data Plane: Abstractions, Architectures, and Open Problems." In: *Proceedings of the IEEE International Conference on High Performance Switching and Routing (HPSR)*. June 2018.

[152] Frank Brockners et al. *Requirements for In-situ OAM*. https://potaroo.net/ietf/all-ids/draft-brockners-inband-oam-requirements-03.html archived at https://perma.cc/34UA-UXZU. Mar. 2017. (Visited on 12/16/2024).

[153] Tian Pan, Enge Song, Zizheng Bian, Xingchen Lin, Xiaoyu Peng, Jiao Zhang, Tao Huang, Bin Liu, and Yunjie Liu. "INT-path: Towards Optimal Path Planning for In-band Network-Wide Telemetry." In: *Proceedings of the IEEE International Conference on Computer Communications (INFOCOM)*. Apr. 2019.

[154] Lanfranco Zanzi, Vincenzo Sciancalepore, Andres Garcia-Saavedra, Hans Dieter Schotten, and Xavier Costa-Perez. "LACO: A Latency-Driven Network Slicing Orchestration in Beyond-5G Networks." In: *IEEE Transactions on Wireless Communications* (Jan. 2021).

[155] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. "Resilient Datacenter Load Balancing in the Wild." In: *Proceedings of the ACM SIGCOMM Conference (SIGCOMM)*. Aug. 2017.

[156] Michael Chow, David Meisner, Jason Flinn, Daniel Peek, and Thomas F Wenisch. "The Mystery Machine: End-to-End Performance Analysis of Large-scale Internet Services." In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2014.

[157] Kaihui Gao, Chen Sun, Shuai Wang, Dan Li, Yu Zhou, Hongqiang Harry Liu, Lingjun Zhu, and Ming Zhang. "Buffer-based End-to-End Request Event Monitoring in the Cloud." In: *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 2022.

[158] Jonathan Kaldor, Jonathan Mace, Michał Bejda, Edison Gao, Wiktor Kuropatwa, Joe O'Neill, Kian Win Ong, Bill Schaller, Pingjia Shan, Brendan Viscomi, et al. "Canopy: An End-to-end Performance Tracing and Analysis System." In: *Proceedings of the ACM SIGOPS Symposium on Operating Systems Principles (SOSP)*. 2017.

[159] Srikanth Kandula, Ratul Mahajan, Patrick Verkaik, Sharad Agarwal, Jitendra Padhye, and Paramvir Bahl. "Detailed Diagnosis in Enterprise Networks." In: *Proceedings of the ACM SIGCOMM Conference (SIGCOMM)*. 2009.

[160] Ding Yuan, Soyeon Park, Peng Huang, Yang Liu, Michael M Lee, Xiaoming Tang, Yuanyuan Zhou, and Stefan Savage. "Be Conservative: Enhancing Failure Diagnosis with Proactive Logging." In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2012.

[161] Xu Zhao, Yongle Zhang, David Lion, Muhammad Faizan Ullah, Yu Luo, Ding Yuan, and Michael Stumm. "lprof: A Non-intrusive Request Flow Profiler for Distributed Systems." In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2014.

[162] Praveen Tammana, Rachit Agarwal, and Myungjin Lee. "Simplifying Datacenter Network Debugging with PathDump." In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2016.

[163] Praveen Tammana, Rachit Agarwal, and Myungjin Lee. "Distributed Network Monitoring and Debugging with SwitchPointer." In: *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. 2018.

[164] Brian Trammell, Jean-Pierre Smith, and Adrian Perrig. "Adding Path Awareness to the Internet Architecture." In: *IEEE Internet Computing* (2018).

[165] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir. *Segment Routing Architecture*. RFC 8402. IETF, July 2018. URL: http://tools.ietf.org/rfc/rfc8402.txt.

[166] Andreas Haas, Andreas Rossberg, Derek L. Schuff, Ben L. Titzer, Michael Holman, Dan Gohman, Luke Wagner, Alon Zakai, and JF Bastien. "Bringing the Web up to Speed with WebAssembly." In: *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*. 2017.

[167] Tom Kuchler, Michael Giardino, Timothy Roscoe, and Ana Klimovic. "Function as a Function." In: *Proceedings of the ACM Symposium on Cloud Computing (SoCC)*. 2023.

[168] *Wasmer*. https://wasmer.io archived at https://perma.cc/6DLH-8J6N. (Visited on 01/09/2024).

[169] *Move Concepts*. https://docs.sui.io/concepts/sui-move-concepts archived at https://perma.cc/F5YB-74PF. (Visited on 01/09/2024).

[170] *SCIONLab*. https://www.scionlab.org archived at https://perma.cc/7YXB-UG2E. (Visited on 01/09/2024).

[171] *Sui Lutris: The distributed system protocol at the heart of Sui*. https://mystenlabs.com/blog/sui-lutris-the-distributed-system-protocol-at-the-heart-of-sui# archived at https://perma.cc/287P-LLCM. (Visited on 01/16/2024).

[172] *SUI Price*. archived at https://perma.cc/VG2T-C3Y2. (Visited on 01/15/2024).

[173] Rafael Pass and Abhi Shelat. *Micropayments for Decentralized Currencies*. https://eprint.iacr.org/2016/332.pdf archived at https://perma.cc/BDW4-X2Y4. 2016. (Visited on 12/16/2024).

[174] *Ripe Atlas*. https://atlas.ripe.net archived at https://perma.cc/PDG2-H67L. (Visited on 01/15/2024).

[175] William Sussman, Emily Marx, Venkat Arun, Akshay Narayan, Mohammad Alizadeh, Hari Balakrishnan, Aurojit Panda, and Scott Shenker. "The Case for an Internet Primitive for Fault Localization." In: *Proceedings of the ACM Workshop on Hot Topics in Networks (HotNets)*. 2022.

[176] Alan Shieh, Emin Gün Sirer, and Fred B. Schneider. "NetQuery: A Knowledge Plane for Reasoning about Network Properties." In: *ACM SIGCOMM Computer Communication Review (CCR)* (2011).

[177] Harsha Madhyastha, Tomas Isdal, Michael Piatek, Colin Dixon, Thomas Anderson, Arvind Krishnamurthy, and Arun Venkataramani. "IPlane: An Information Plane for Distributed Services." In: *Proceedings of the USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 2006.

[178] Behnaz Arzani, Selim Ciraci, Boon Thau Loo, Assaf Schuster, and Geoff Outhred. "Taking the Blame Game out of Data Centers Operations with NetPoirot." In: *Proceedings of the ACM SIGCOMM Conference (SIGCOMM)*. 2016.

[179] Yin Zhang, Lee Breslau, Vern Paxson, and Scott Shenker. "On the Characteristics and Origins of Internet Flow Rates." In: *ACM SIGCOMM Computer Communication Review (CCR)* (2002).

[180] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. "One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon." In: *Proceedings of the ACM SIGCOMM Conference (SIGCOMM)*. 2016.

[181] Vikram Nathan, Srinivas Narayana, Anirudh Sivaraman, Prateesh Goyal, Venkat Arun, Mohammad Alizadeh, Vimalkumar Jeyakumar, and Changhoon Kim. "Demonstration of the Marple System for Network Performance Monitoring." In: *Proceedings of the ACM SIGCOMM Posters and Demos*. 2017.

[182] Minlan Yu, Lavanya Jose, and Rui Miao. "Software Defined Traffic Measurement with OpenSketch." In: *Proceedings of the USENIX Conference on Networked Systems Design and Implementation (NSDI)*. 2013.