Polaris: End-to-End Path Optimization by End Hosts

Elham Ehsani Moghadam† Patrick Wicki† François Wirz† Jordi Subirà Nieto† Yih-Chun Hu‡ Adrian Perrig†
†ETH Zurich ‡University of Illinois at Urbana-Champaign

***Abstract*—Path-aware networking (PAN) enables endpoints to locally select end-to-end network paths based on path properties. This approach contrasts with the traditional Internet architecture, where routers determine the next hop towards the destination based on the routing information provided by the Border Gateway Protocol (BGP). By providing this additional transparency and control, PAN opens up opportunities to optimize path selection, with the potential to enhance network performance and user experience metrics.**

In this paper, we evaluate the potential benefits of PAN for enhancing end-to-end performance. We design Polaris, a concrete feedback-driven path optimization mechanism for PAN, and study its impact on Quality of Service (QoS) as compared to current Internet mechanisms. Our extensive simulation results show the viability and effectiveness of Polaris, revealing that it outperforms the current Internet mechanisms by an average of 42% improvement in receiving rate and 81% reduction in median loss, in the presence of background traffic.

***Index Terms*—Path Selection, Path Aware Networking, Quality of Service**

I. Introduction

Path-aware networking (PAN) is a networking paradigm where endpoints are made aware of various potential paths their data can take through the network and can select among these paths [1]. This is in contrast to the traditional Internet architecture, where path selection is primarily controlled by intermediate devices like routers and is usually opaque to endpoints.

While the current Internet predominantly entrusts path selection to intermediary nodes, some exceptions such as Multipath TCP/QUIC [2], [3] allow endpoints to exercise limited control by choosing between or splitting traffic over options such as Wi-Fi or cellular networks for the first hop. This limited control is a first step towards PAN.

The added transparency and control inherent in PANs compared to the hop-by-hop route selection in today's Internet, provides opportunities to optimize path selection, thus potentially improving network performance and user experience.

One advantage of end-host-selected paths is that end hosts know the characteristics and temporal behavior of their own traffic, whereas intermediate nodes do not. However, complete information about their own traffic is not sufficient for optimizing path selection. To effectively select paths, end hosts also need to acquire information about the network state. This challenge exists in today's Internet as well. While a router knows its own state, it lacks adequate information about other routers in the network. In this paper, we conceptualize the end host as the entire network stack including the application, rather than the aggregate of all applications on a single device. We make this distinction because each application can control its sending rate and jointly decide on both the sending rate and the used inter-domain path.

979-8-3315-4940-4/25/$31.00 ©2025 IEEE

Another advantage of end-host-selected paths is that end hosts can make fine-tuned adjustments to the end-to-end path in real time. However, inter-domain path selection in today's Internet, primarily governed by BGP, is limited. A BGP router selects the next hop based on BGP path selection criteria, which are generally agnostic to real-time network conditions such as current traffic load or congestion levels [4], [5].

Once PAN end hosts obtain information about the network state, they need to use it effectively to optimize their path choices. However, allowing end hosts to switch to better paths introduces the issue of selfish routing, which can lead to network instability [6].

These challenges can be framed through two research questions: How can end hosts *efficiently* and *effectively* receive *practical* and *actionable* feedback from the network concerning available paths, and how can they *optimize* their path choices based on this feedback?

In the current Internet, applications already react to network feedback by applying congestion control mechanisms. These mechanisms attempt to optimize performance by adjusting the sending rate as a response to network congestion as indicated by packet loss or latency. These congestion control mechanisms have proven effective; they allow data transmission along a 'single' working path to be optimized. However, significant challenges arise when considering 'alternative' routing paths. The core functionality of congestion control mechanisms is to adjust the sending rate based on the feedback for the current rate over the active path. However, this approach does not apply to alternative paths since the end host is not actively sending traffic over them. Instead, the end host can send probing packets on the alternative paths, but this must be done at limited bandwidth to avoid imposing a substantial load on the network or saturating local links.

Probing can help assess the round-trip time (RTT), one-way latency, or even latency gradients along a path. Nonetheless, the actual available bandwidth share for future flows on an alternative path remains undetermined. Even if a path is free from congested links, the presence of a bottleneck with limited capacity can render it unsuitable for bandwidth-intensive traffic. This limitation of probing underscores the need for a novel feedback mechanism that can, at a minimum, provide an estimate of the available bandwidth share on a candidate alternative path to the end host.

Given such a feedback system, the next challenge is how to incorporate this real-time feedback in the path selection mechanism. The path-switching operation must be seamless and adaptive, ensuring that application performance is optimized while maintaining fairness and avoiding undue strain on the network infrastructure.

The requirements for different types of traffic vary, necessitating tailored path selection mechanisms. One particularly significant and growing category is bandwidth-demanding and latency-sensitive traffic, such as video conferencing. The rapid growth of real-time communications underscores the need to improve QoS of video conferencing traffic [7]. Industry interest in technologies such as L4S [8] highlights this priority.

In this paper, we evaluate the potential benefits of PAN for enhancing QoS for video conferencing applications. We propose Polaris[1], a path optimization framework for end hosts. Polaris includes a feedback system and a feedback-driven path selection strategy. We study the impact of Polaris on the end-to-end performance of video conferencing traffic. To the best of our knowledge, our paper is the first to present the design and evaluation of a feedback-driven end-to-end path optimization system for PAN end hosts. Specifically, our work makes the following main contributions:

- We introduce a novel feedback mechanism that provides end hosts with real-time insights into network conditions, including available bandwidth on candidate paths.
- Our system incorporates feedback-driven path selection strategies, enabling endpoints to optimize routing decisions and seamlessly switch paths according to network conditions and traffic profiles.
- We conduct extensive simulations to evaluate the effectiveness of Polaris, showing significant improvements in end-to-end performance metrics for video conferencing applications compared to current Internet mechanisms.

In the following sections, we present the motivation of Polaris (§II) followed by the system design (§III) and evaluation setup details (§IV). Finally, we report the evaluation results (§V) and summarize related work (§VI).

This work does not raise any ethical issues.

II. Motivation and Background

To motivate our work, we explore the limitations of traditional inter-domain routing including the lack of adaptability to traffic requirements and congestion. We then highlight the potential of PAN for addressing these limitations and the relevance of video conferencing as a motivating use case.

a) ***Agnostic to traffic requirements****:* The traditional design of the network layer of a host is largely agnostic to the specific traffic requirements, relying on broad, standardized protocols to facilitate data transmission. This separation constrains the network layer from adapting dynamically to the diverse requirements of various applications. In contrast, the application layer possesses knowledge of its own needs, enabling it to adapt its behavior based on feedback from the network or destination. This capability can improve path selection and potentially enhance performance [9], especially for bandwidth-demanding, bandwidth-sensitive, and latency-sensitive applications such as video conferencing [10], [11].

b) ***Agnostic to congestion****:* Inter-domain path selection in today's Internet, primarily governed by BGP, is largely agnostic to network congestion. BGP does not consider current traffic load or congestion levels when determining the best paths [4], [5]. BGP's routing decisions are based on predefined criteria such as path length, origin of the route, local preference, and Multi-Exit Discriminator (MED), mostly intended to ensure routing stability, adherence to AS relationships, and compliance with financial and policy agreements [12]. Although mechanisms such as Equal-Cost Multi-Path (ECMP) and traffic engineering via specific prefix announcements attempt to optimize load distribution within or across ASes, they do not respond to real-time congestion levels.

These methods rely on static metrics and administrator-defined policies, rather than dynamically adapting to current network congestion. As a result, while BGP is typically effective for maintaining policy compliance and overall routing across the global Internet, it does not dynamically adapt to congestion, which can lead to suboptimal routing and congestion on specific paths [4]. Intra-AS protocols such as OSPF can impact BGP path selection within an AS by providing congestion information through weights that reflect network conditions. However, this influence is limited to scenarios where the IGP cost is a tie-breaker. An intra-AS protocol only provides information about the intra-AS links of the current AS. However, the overall performance of the end-to-end path depends on congestion across both intra-AS and inter-AS links along the entire route [13]. BGP remains agnostic to congestion in the rest of the network, as BGP update messages do not carry performance information.

c) ***Myopic view of the network****:* Intermediary nodes, compared to end hosts, might appear to have a better view of the network's congestion state, but this is a misconception. While individual routers possess the most accurate knowledge about their queues' state, they remain unaware of the queue state of downstream routers unless this information is explicitly communicated back to them. However, if routers convey queue state information back towards the source, end hosts can eventually aggregate this backpropagated information and make informed decisions for path selection.

d) ***Emergence and deployment of PANs****:* PANs present new opportunities for optimizing path selection, potentially enhancing network performance and user experience. PANs allow the source to embed a forwarding path into the packet header or leverage packet-carried forwarding state (PCFS). Several PAN architectures have been proposed in recent years, including Platypus [14], NIRA [15], Pathlets [16], Segment Routing [17], and SCION [18].

We instantiated Polaris in SCION as the PAN for our implementation, because it is already deployed and operational in real-world production networks with a mature implemen-

[1]We named our system after Polaris, the North Star, symbolizing guidance and optimization in navigating paths.

tation. Additionally, multiple SCION simulators are publicly available [19], [20], making it a suitable foundation for the implementation of our path selection method.

e) ***Beyond over-provisioning****:* In today's Internet, over-provisioning is a crucial strategy to prevent congestion by allocating more resources than necessary. Over-provisioning comes with associated costs, such as higher CAPEX for purchasing extra equipment [21]. Dynamic resource allocation techniques can help optimize resource utilization, potentially reducing the need for over-provisioning or enhancing an ISP's revenue through increased network utilization and efficiency.

f) ***Economic considerations****:* In inter-domain path selection, business considerations often outweigh performance factors like QoS and load balancing [22]. ASes announce paths based on business relationships, typically avoiding transit between providers to minimize costs. PAN, as it allows end hosts to select their end-to-end paths, poses a challenge for intermediate ASes in terms of cost optimization. However, path choices remain constrained by AS-announced options, aligning with economic incentives. Routers can also implicitly influence selection through traffic shaping. To further support AS traffic engineering, Polaris includes an explicit feedback mechanism that allows ASes to prioritize their traffic engineering decisions over end hosts' path optimization needs when necessary.

g) ***Video conferencing****:* Video conferencing traffic is highly sensitive to congestion and requires a consistent sending rate to maintain a seamless user experience. Video conferencing applications that use a transport layer protocol without congestion control, such as UDP, employ alternative congestion control strategies. These applications use algorithms such as Google Congestion Control (GCC) [23], [24] in conjunction with protocols such as RTP (Real-time Transport Protocol) and RTCP (RTP Control Protocol) [25].

GCC is the default congestion control algorithm of WebRTC [26], a fundamental technology in all major web browsers [27], used to manage network bandwidth for real-time applications. GCC dynamically adjusts the bitrate of the video stream in response to real-time network conditions, such as changes in available bandwidth, latency, and packet loss. It relies on RTP for media streaming and RTCP for periodic transmission of control packets, which include key metrics such as packet loss, jitter, and round-trip time.

When congestion occurs on an active path, congestion control mechanisms may reduce the sending rate to alleviate the congestion. However, for video conferencing applications, this rate reduction can lead to degraded video quality or even disconnections. To address this challenge, switching to a new path, where the sending rate can be maintained or even increased, can enhance the user experience. In this paper, we measure receiving rate, loss, latency, and jitter as QoS metrics for video conferencing traffic.

III. System Design

In this section, we outline the design of Polaris, which includes a feedback system, a path-aware congestion control, and a path selection mechanism.

A. Feedback system

To switch to a path with higher available bandwidth, an end host needs to compare its active path with alternatives. However, while the congestion controller monitors the active path, it lacks visibility into the available bandwidth on alternative paths. A lightweight probe can measure latency on these paths, but bandwidth remains unknown. To address this, we design a feedback system that conveys available bandwidth information of alternative paths to the end host. This system relies on two types of messages: the Polaris probe, referred to as *P-probe*, for providing frequent estimates of the available bandwidth to the end host, and the Polaris Congestion Alert, referred to as *P-CA*, to alert the end host about a congestion situation.

1) ***Polaris Probe****:* We propose the P-probe message, which enables the end host to receive an estimate of its bandwidth share on a path. This allows the end host to compare its share on the current path with its estimated share on the alternative paths.

Each router in the network is aware of its link capacity, can keep track of the number of flows traversing it, and can monitor its current queue length. Furthermore, routers know their queue management mechanisms and how capacity is allocated among different flows, including any policies that prioritize certain flows based on their destinations. If routers backpropagate this information to the sending host, the host can gain insight into its bandwidth share along the path. However, sending this information individually to each sender would incur significant overhead. To address this, we propose the P-probe mechanism.

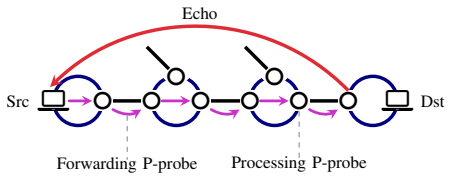

Fig. 1: Overview of Polaris's operation on a path of four ASes (blue circles). The source sends a P-probe along the path, the border routers ○ on the path update the probe and the last border router echoes it back to the sender.

As shown Figure 1, a P-probe message carries the information to the last border router on the path, where it is then echoed back to the source. In PAN, the path included in the packet header enables the last router to identify itself as such. Each border router along the path calculates its bandwidth share for a new flow on the corresponding link and updates certain fields in the probe if needed.

Only border routers contribute to this feedback system since the path is inter-domain and includes only the ingress and egress interfaces of the AS, excluding intra-AS routers. This is acceptable because the ingress border router can estimate a flow's bandwidth share between the ingress and egress points based on internal traffic engineering and monitoring mechanisms. Additionally, inter-domain links are more prone to congestion than intra-AS links.

The P-probe message includes a 16-bit field for the bandwidth share on the path's bottleneck. If a router's calculated share is lower than the current Bottleneck Share, the router should update the Bottleneck Share field with its own value and identify itself as the bottleneck in the P-probe. To efficiently use the 16-bit field, the estimated value (in Mb/s) is encoded in IEEE-754 half-precision floating point format. P-probe also includes a 16-bit field for the cumulative queuing delay (in μs), encoded in the same format. Each border router updates the cumulative queuing delay field by adding the queue waiting time of the current packet. The details of the router processing and the layout of the Polaris messages are depicted in Figures 2 and 3, respectively.

P-probe is an extension to the network layer protocol such as the IPv6 or SCION hop-by-hop extension header. The hop-by-hop extension header is used to carry optional information that may be examined and processed by every node along a packet's delivery path.

The ***NextHdr*** field specifies the type of the next header, and ***ExtLen*** is the length of the extension header. The rest of the packet is ***Options***, which is a variable-length field. We incorporate the following fields in the Options field, inspired by the SCION Control Message Protocol (SCMP) message [28]. SCMP is analogous to ICMP in the current Internet, providing functionalities for network diagnostics (similar to Ping and Traceroute) and transmitting error messages related to packet processing and network-layer issues to end hosts.

The ***Type*** field is set to indicate that it is a P-probe. The ***Code*** field shows the traffic class (e.g., video). The ***Checksum*** field is used to detect data corruption. The ***RequestID*** and ***SeqNum*** fields are a 16-bit identifier and a 16-bit sequence number, respectively, to aid in matching the echo message with the probe message. The ***AS Identifier*** and ***InterfaceID*** fields are the 64-bit AS identifier and 16-bit Interface ID of the bottleneck Interface, respectively. Optionally, routers can replace these two fields with zeros to conceal the bottleneck location. The ***BottleneckShare*** field is the estimated bottleneck bandwidth share for a data flow with the same source and destination as the P-probe. The ***CumQDelay*** is the summation of the estimated waiting time of the current packet in the queue of each on-path border router.

We do not specify how routers must compute their P-probe bottleneck share. In our implementation, a router estimates the potential share of a flow as link bandwidth divided by active flows plus one (to account for the new flow). This estimation is approximate, as flows may join or leave the path dynamically. Therefore, each end host considers a margin when making decisions based on this estimation.

A router is naturally incentivized to report truthfully—it risks congestion if it overstates the share and potential revenue loss if it understates it.

Each border router keeps a hash table of flow tuples *(src_host_addr, src_port, dst_host_addr, dst_port)* with timestamps. Every τ (default: 1s), entries older than τ are removed, and the number of active flows is the table size. A larger τ reduces processing overhead but may decrease accuracy.

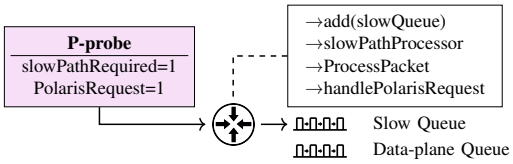

Fig. 2: Detailed view of router P-probe processing.

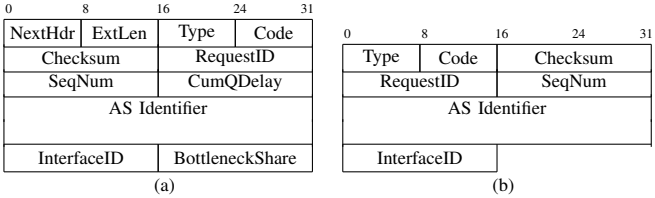

Fig. 3: Packet layouts of a) a P-probe and b) a P-CA notification.

While per-data packet update is computationally expensive for routers, there exist novel probabilistic sketches providing efficient estimates of flow counts [29].

*2) **Polaris Congestion Alert**:* We propose the P-CA message, which aims to enable an on-path border router to inform the sender directly in case of congestion. For a P-probe, when the router queue length is substantial and there is a high probability of dropping a packet, the border router will drop the P-probe and send a P-CA back to the sender.

Figure 3b shows the layout of the P-CA message. The ***Type*** field is set to indicate it is a P-CA. The ***Code*** field is reserved for future additions. The ***RequestID*** and ***SeqNum*** fields are set to the same values as those in the P-probe. ***AS Identifier*** and ***InterfaceID*** fields identify the P-CA originator.

B. Congestion Control

We employ GCC to manage congestion control on the active path. GCC is designed for a single working path; when integrating it into Polaris, we must ensure smooth transitions during path switches to prevent performance degradation and maintain fairness with other flows (§ III-C).

C. Path Selection Strategy

In large-scale PANs, an end host will likely have numerous path options to the destination. Probing all paths is inefficient, so the end host can select a subset of paths as available paths.

P-probes are sent every 250ms, matching that of GCC, which effectively captures network dynamics with minimal overhead. Path switching is evaluated at a lower frequency, with a minimum interval of 10s, referred to as waiting time, between consecutive changes. This value is comparable to the turbulence period after path switching, allowing flows to reach their fair share instead of hastily switching to another path, thus reducing instability from frequent switches.

The sender maintains candidacy information for each available path. This information changes every time the congestion controller adjusts the sending rate (R_s in Mb/s). A path is added to the set of potential candidate paths if the following conditions are met:

- The estimated bandwidth share ($\hat{R}$ in Mb/s) must be from a P-probe sent after $t_{last\_path\_change} + RTT$, with

no P-CA received in the interim. This ensures potential changes in $\hat{R}$ due to the last path change are accounted for.
- The estimated bandwidth on that path must be larger than $\alpha \cdot R_s$, where α is a factor greater than 1 (1.5 by default), providing a margin for the estimation accuracy and some inertia against unnecessary path changes.

A path that consistently meets these criteria over the last 5s (half of the waiting time) is considered a final candidate, and the current working path is always included as a candidate. If more than 10s (waiting time) has elapsed since the last path consideration, the sender selects a path uniformly at random among final candidates. This helps prevent a scenario where multiple end hosts switch to the same path simultaneously, thus maintaining stability. If there is a P-CA for the active path, an immediate path switch consideration is triggered, although it does not necessarily result in a path change. If the active path fails and an alternative is available, a path switch occurs immediately.

To ensure a smooth transition when switching paths, we update GCC by calling `OnRouteChange()` and providing $\hat{R}$ as the new target rate. The target rate is an indication of the potential bandwidth the flow may ultimately get, with a larger target enabling a faster slow start. This resets the internal state to facilitate adaptation to the new path's conditions. Starting with $\hat{R}$, rather than a minimal rate, may cause higher initial losses but will stabilize as other flows back off, and the rate converges to the actual fair share by adjusting according to GCC. This approach prevents a sudden decline in data throughput after a path change, without compromising fairness.

D. Virtual Traffic Shaping

In the feedback system of Polaris, a border router can flexibly adjust the value it inserts into the P-probe, allowing for customized bandwidth allocation based on its policies or traffic engineering requirements. When it needs to limit bandwidth, the router updates the 'BottleneckShare' field in the P-probe to indicate the reduced bandwidth allocation instead of the actual share. Additionally, a router can proactively send a P-CA signal when it has to constrain bandwidth, rather than waiting for congestion to occur.

E. Partial Deployment

To enable partial deployment of our system, we set the highest-order 2 bits of the Option Type identifier to 00, ensuring that routers that do not recognize the option will simply skip it and continue processing and forwarding the packet [30]. However, the resulting bottleneck share measurement may be less accurate, as it will rely only on the routers that recognize and update the probe. This behavior is analogous to cases where a router chooses not to update the probe.

F. Security

A thorough analysis of security considerations is beyond this paper's scope; however, we briefly discuss one potential attack scenario. An adversary could spoof a reply to a probe, misleading end hosts and redirecting traffic to cause denial-of-service. In PANs, such attacks are prevented by authenticating control messages that trigger far-reaching decisions. Specifically, in SCION, SCMP messages are authenticated using DRKey and SPAO (SCION Packet Authenticator Option) [31].

G. Overhead and scalability

Each end host sends a 20-byte P-probe every 250ms, consuming 80 B/s per host, with the same for replies. Routers process each probe packet in $\mathcal{O}(1)$ time, including reading, writing, comparison, summation, and division. The same applies to queue length tracking for delay recording. A probabilistic sketch for flow counting ensures the scalability in the fair share estimation process.

IV. Evaluation Setup

We aim to evaluate the end-to-end performance of Polaris compared to traditional path selection methods. To achieve this, we implement Polaris alongside alternative methods in a simulated environment, using an inter-AS topology, randomly generated intra-AS topology per AS, and traffic flows between hosts. Additionally, we develop a Mixed Integer Linear Programming (MILP) model to validate the simulation results. This section provides an overview of our evaluation methodology.

A. Topology

1) Inter-AS: We employ a toy topology consisting of 4 ASes and 12 links with 10MB/s bandwidth. As shown in Figure 4, there are multiple links between each pair of ASes. In this topology, the degree of each AS indicates the number of border routers in each AS. We consider all the feasible paths between source and destination as available paths.

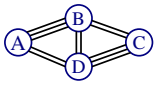

Fig. 4: AS-level view of the inter-AS topology used in the evaluation, with 4 ASes and 12 inter-domain 10MB/s links, where ASes B and D have two peering links between themselves and are providers for ASes A and C.

2) Intra-AS: We generate the intra-AS topologies using the Watts-Strogatz model [32], which generates random graphs with small-world properties such as short average path length and high clustering. We consider a full-mesh iBGP topology.

B. Traffic Flows

Our simulation includes three types of traffic flows: Polaris traffic flows, TCP-Cubic traffic flows, and GCC traffic flows. The latter two represent typical network flows, where end-hosts cannot select or switch paths during transmission.

Our target traffic is Polaris traffic flows, for which we measure the quality metrics. The inclusion of TCP-Cubic and GCC traffic flows allows us to ensure the fairness of Polaris traffic when coexisting with other types of traffic and evaluate Polaris's performance in a mixed traffic environment.

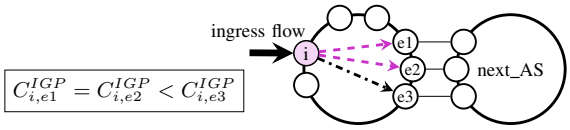

Fig. 5: BGP Load Balancing (LB): ECMP LB uses two equal-cost links (- -), while Full-diversity LB distributes the traffic over all three links (- - , -·).

C. Alternative Path Selection Methods

We compare Polaris with two other alternatives: A naïve path selection scheme in PANs and BGP load balancing.

1) Naïve Path Selection: In the naïve load balancing scheme (denoted as Naïve PAN), each end host in a PAN chooses among *all* available paths randomly (without weighting them) and keeps sending on that path for the rest of the session.

2) BGP Load Balancing: Since ECMP is a common load-balancing method in BGP, we also compare Polaris against this alternative path selection method. ECMP halts the tie-breaking process after evaluating the IGP cost for the NEXT_HOP attribute. BGP implementations typically include a configuration option that sets the maximum number of equal paths allowed in the routing table (e.g., six paths). For ECMP selection, paths must have identical AS_PATH lengths and often identical AS_SEQUENCE segments. If an eBGP path is present, all iBGP paths are excluded from the ECMP candidate set. For iBGP-learned routes, the final decision usually depends on the IGP metric of the NEXT_HOP attribute, unless tunneling mechanisms like MPLS are used. Even with multiple paths in the routing table, a BGP speaker advertises only a single best path to its peers, chosen by standard tie-breaking rules. We simulate BGP load balancing using the following methods:

- **Full-Diversity Flow-Level Load Balancing:** In this scenario, the IGP cost is assumed to be the same for all ingress-egress pairs. In each AS on the best BGP AS-level path, the BGP load balancer assigns a flow, based on the hash of its source and destination host tuple, to one of the egress interfaces, as shown in Figure 5.
- **ECMP Flow-Level Load Balancing:** Here, in each AS on the best BGP path, the egresses with the minimum IGP cost from the ingress interface are listed (Figure 5, e1 and e2). The ECMP load balancer then assigns a flow based on its hash to one of the equal-cost egress interfaces.

D. Simulation Implementation

We implemented Polaris in C++ (with ∼1.5k lines of code) within the ns-3 SCION network simulator [19]. The simulator includes GCC senders and receivers as shown in Figure 6. The GCC sender is implemented based on real instances of WebRTC [2] and adapted to fit the simulation environment. The GCC receiver records and sends back the arrival times of all received packets to the sender. We also developed Python scripts to generate the input topology file, as well as to create the intra-AS topology and the IGP cost matrix for each AS, which are necessary for BGP load balancing. All code will be released as open-source.

[2]The `GoogCcNetworkController` module from the WebRTC code repository of the Chromium browser [33].

E. Theoretical Model

To validate the simulation results, we propose the MILP in Table II to solve the global path selection problem. The constants and variables are listed in Table I. Equation (1) is the objective of the MILP. The goal is to maximize all the flow rates within the constraints of a linear objective. The term $\sum_{f,t} x_{f,t}$ maximizes the summation of the flows. To avoid cases where the overall flow rate is maximized, while some flows starve, we first maximize the minimum flow rate at each time slot ($\sum_t d_t$) and then the summation of all flow rates.

Equation (2) ensures that each flow has exactly one active path during its active time. Equation (3) ensures that non-target flows select the first path in the list. When path p is used by flow f, Equation (4) ensures that all links on path p are assigned to this flow at that time. Equations (5) to (7) are the linear form of $z_{f,t,e} = a_{f,t,e} \cdot x_{t,e}$, which set the same bandwidth ($x_{t,e}$) for flow f on its active links. Equation (8) ensures that the sum of flows over a link does not exceed its capacity. Equation (9) constrains the rate of each flow to its application limit. Equations (10) and (11) set $g_{f,t} = 1$ for the application-limited flows. Equation (12) constrains the rate of the flows on a link to at most be equal to the bottleneck share of that link. Equation (13) constrains the rate of the flows bottlenecked at link e to at least be equal to its bottleneck share. In combination with Equation (12), the bandwidth of these flows is set equal to the bottleneck share. Equation (14) ensures that a flow, if not application-limited, has exactly one bottleneck link on its path. Equation (15) prevents a flow from having a non-active link as its bottleneck. Equations (16) and (17) set the value of variable d_t as the smallest flow share at time t among all flows.

Our MILP ensures that each non-application-limited flow has a single bottleneck link. We prove that the MILP cor-

Indices:	
$f \in F$: flow; $e \in E$: link; $p \in P_f$: path; $t \in T$: time slot	
Constants	
$\delta_{f,p,e}$	1 if link e belongs to path p of flow f, 0 o.w.
λ_f	Application limited rate of flow f
τ_f	Duration of flow f
ζ_f	Starting time of flow f
μ_f	0 if f is a target flow, 1 otherwise
$\nu_{f,t}$	1 if flow f is active at time t, 0 otherwise
γ_e	Capacity of link e
M	Big positive integer
ϵ	Small positive value
Variables	
$x_{f,t}$	Rate of flow f at time t
$y_{f,t,p}$	1 if flow f is using path p at time t, 0 otherwise
$z_{f,t,e}$	Rate of flow f at time t on link e
$a_{f,t,e}$	1 if flow f is using link e at time t, 0 otherwise
$b_{t,e}$	Fair share on link e at time t
$c_{f,t,e}$	1 if link e is the bottleneck of flow f at time t, 0 otherwise
$g_{f,t}$	1 if flow f is application-limited at time t, 0 otherwise
d_t	Minimum flow rate at time t

TABLE I: Definitions of the MILP constants and variables.

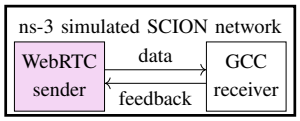

Fig. 6: Overview of the implementation, featuring a real WebRTC sender instance and a GCC receiver within an ns-3 simulated SCION network.

rectly identifies the actual bottleneck link, using a proof by contradiction. Suppose that the MILP selects link e as the bottleneck of flow f at time t, but the actual bottleneck is link e' ($e' \neq e$). We put aside the time index t since it remains consistent throughout this proof. Let $\hat{b}_e$ denote the fair share on link e if it is selected as the bottleneck of flow f, and b_e denote the fair share of link e when it is not selected as the bottleneck of flow f. According to the assumption, $\hat{b}_{e'} < \hat{b}_e$. Given that e is selected as the bottleneck by the MILP, $b_e = \hat{b}_e$, and the bandwidth of flow f on link e ($z_{f,e}$) equals the fair share on link e ($z_{f,e} = b_e = \hat{b}_e$), according to Equations (12) and (13). However, $b_{e'}$ is not necessarily equal to $\hat{b}_{e'}$, as link e' is not selected as the bottleneck. According to Equations (5) to (7), a flow maintains the same rate across different links in its path ($z_{f,e} = x_f = z_{f,e'}$). Then, we derive $\hat{b}_{e'}$ by subtracting the bandwidth taken by flows not bottlenecked at link e' (α) from its capacity and dividing it by the number of flows bottlenecked at e' (β_n), including flow f:

$$\hat{b}_{e'} = \frac{\gamma_e - \alpha}{\beta_n} = \frac{\gamma_e - \alpha - \hat{b}_{e'}}{\beta_n - 1} \tag{18}$$

We continue by deriving $b_{e'}$, for the case in which flow f is not among the bottlenecked flows on link e':

$$b_{e'} = \frac{\gamma_e - \alpha - z_{f,e'}}{\beta_n - 1} < \frac{\gamma_e - \alpha - \hat{b}_{e'}}{\beta_n - 1} \overset{\text{Equation (18)}}{=} \hat{b}_{e'} \tag{19}$$

$$\Rightarrow b_{e'} < \hat{b}_e < \hat{b}_{e'} = z_{f,e'} \Rightarrow b_{e'} < z_{f,e'} \tag{20}$$

This deduction contradicts Equation Equation (12).

$\max \quad \sum_t d_t + \epsilon \cdot \sum_{f,t} x_{f,t}$	(1)
$\sum_p y_{f,t,p} = \nu_{f,t} \quad \forall f,t$	(2)
$\mu_f \cdot (y_{f,t,0} - \nu_{f,t}) = 0 \quad \forall f,t$	(3)
$a_{f,t,e} = \sum_p \delta_{f,e,p} \cdot y_{f,t,p} \quad \forall f,t,e$	(4)
$z_{f,t,e} \leq M \cdot a_{f,t,e} \quad \forall f,t,e$	(5)
$z_{f,t,e} \geq x_{f,t} - (1 - a_{f,t,e}) \cdot M \quad \forall f,t,e$	(6)
$z_{f,t,e} \leq x_{f,t} \quad \forall f,t,e$	(7)
$\sum_f z_{f,t,e} \leq \gamma_e \quad \forall t,e$	(8)
$x_{f,t} \leq \lambda_f \quad \forall f,t$	(9)
$g_{f,t} \geq 1 - M \cdot (\lambda_{f,t} - x_{f,t}) \quad \forall f,t$	(10)
$(\lambda_{f,t} - x_{f,t}) \leq M(1 - g_{f,t}) \quad \forall f,t$	(11)
$z_{f,t,e} \leq b_{t,e} \quad \forall f,t,e$	(12)
$z_{f,t,e} + (1 - c_{f,t,e}) \cdot M \geq b_{t,e} \quad \forall f,t,e$	(13)
$\sum_e c_{f,t,e} + g_{f,t} = \nu_{f,t} \quad \forall f,t$	(14)
$c_{f,t,e} \leq a_{f,t,e} \quad \forall f,t,e$	(15)
$d_t \cdot (1 - g_{f,t}) \cdot \nu_{f,t} \leq x_{f,t} \quad \forall f,t$	(16)
$d_t \leq \gamma_f \quad \forall f,t$	(17)

TABLE II: The MILP formulation.

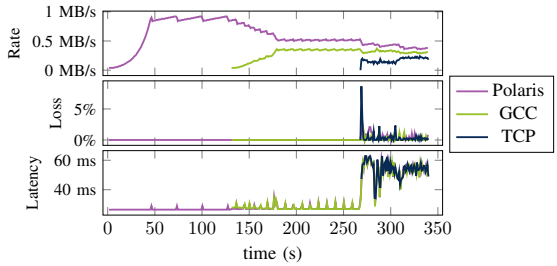

Fig. 7: Competing Polaris, GCC and TCP flows reach a stable state, and the throughput of each flow converges to the fair share on a 1 MB/s link.

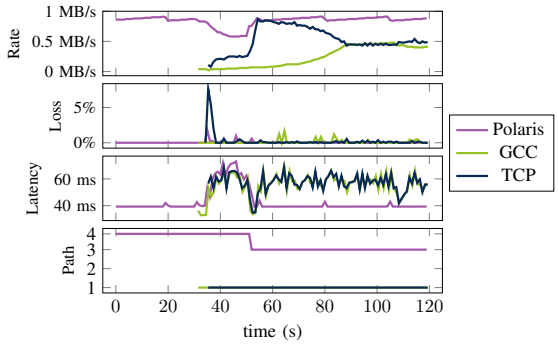

Fig. 8: Once new flows join, Polaris detects that the active path is no longer optimal and switches to a path with a higher bandwidth share.

1) MILP Implementation: We also used the Python API of Gurobi [34] to solve the proposed MILP formulation of the path selection problem. To compute the global solution, we limit input paths to k for a fixed-size variable matrix, selecting the k shortest paths when more are available. All code will be released as open-source. With a bounded or moderately growing k, the MILP complexity is $\mathcal{O}(|F| \cdot |E| \cdot |T|)$. Its flow-based design significantly reduces complexity, allowing it to run efficiently on the toy topology. However, while a flow-based formulation effectively calculates each flow's bandwidth, it cannot capture loss or queuing latency.

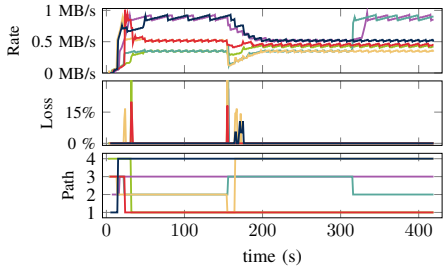

Fig. 9: Six Polaris flows, each represented by a different color, start by spreading out across four paths. The router on path 2 sends out P-CAs from 150 to 300 seconds, prompting the flows to migrate to different paths.

V. Evaluation Results

A. Metrics

We report the evaluation result using the following metrics:

- ***Cumulative Average Receiving Rate*** is the average receiving rate over time for the target flows (excluding

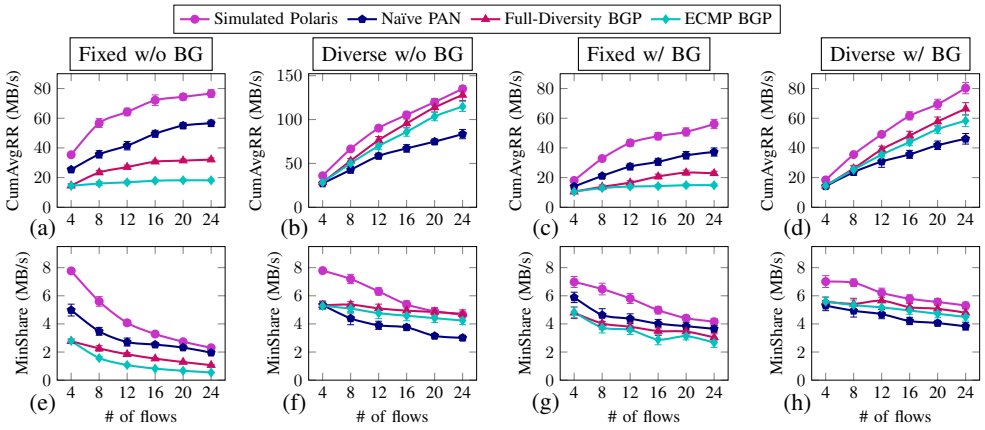

Fig. 10: ns-3 simulation results showing the Cumulative Average Receiving Rate (a-d) and the Minimum Share (e-h) vs. number of flows for the path selection methods Polaris, Naïve PAN, Full-Diversity BGP, and ECMP BGP with Fixed A-C source-destination pairs and Diverse all-to-all source-destination pairs.

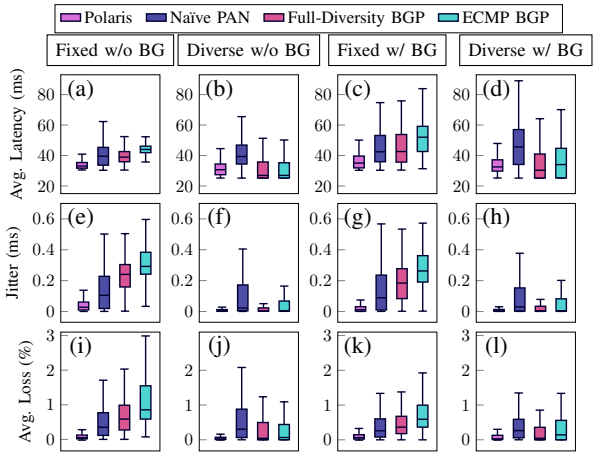

Fig. 11: Latency, jitter, and loss distribution for the different path selection methods *simulated* for 24 flows in the Fixed and Diverse source-destination scenarios with and without background traffic.

the background traffic flows), aggregated over the flows ($CumAvgRR = \sum_f \frac{1}{\tau_f} \cdot \sum_t x_{f,t}$).

- ***Minimum Share*** refers to the lowest receiving rate experienced by any flow. To avoid capturing transient low rates, we divide the time into fixed intervals of 120s and calculate the average receiving rate of each flow for each interval. The minimum of these average rates across all intervals is reported as the minimum share.
- ***Loss, Latency, and Jitter***: Packet loss and latency are recorded for each flow over 1s intervals. Jitter is calculated as the standard deviation of inter-arrival time differences between consecutive packets in the jitter buffer.

B. Flow generation Settings

The flows are defined as tuples in the format *(flow-ID, source, destination, application-limited rate, duration, starting time, flow type)*, where *flow type* specifies whether a flow is a target flow or a background flow. The starting time for each flow is selected randomly from the time slots within the experiment time. Each time slot lasts 120s, and the experiment time comprises 5 time slots. The duration of each flow is chosen randomly, constrained to be between a minimum of 2 time slots and the remaining time. Flows' source and destination are selected from two scenarios: either fixed (node A as the source and node C as the destination, denoted as *Fixed*), or randomly selected among all pairs of distinct nodes (denoted as *Diverse*). Additionally, we consider two different scenarios regarding background traffic: one with no background traffic on the links (denoted as w/o BG) and one with background traffic (denoted as w/ BG). The number of background traffic flows is half of all flows. These background flows are GCC flows, and their path selection follows BGP-ECMP. None of the flows are application-limited. We generate flows for 20 iterations and report the average of CumAvgRR and MinShare with 95% confidence interval as error bar in Figures 10 and 12. The distributions of loss, latency, and jitter are presented across flows and iterations using boxplots in Figure 11.

C. Simulation results

In Figures 7 and 8, we observe the fairness between TCP, GCC, and Polaris flows on the same link and over multiple paths, respectively. Additionally, we see how GCC maintains small queues before a loss-based TCP-Cubic flow joins. In Figure 8, we observe that the Polaris flow switches to an idle path once other flows start on the same path.

Figure 9 demonstrates how the P-CA message triggers a path change. When path 1 begins responding with P-CA to P-probes for a certain period, the Polaris flows on path 1 switch to new paths, with one flow returning after this period.

The simulation results for different metrics are shown in Figures 10 and 11. We observe that Polaris outperforms all other approaches in every scenario. In the Diverse scenario, ECMP and Full-Diversity BGP perform similarly, indicating that increasing interface-level diversity in BGP does not necessarily improve performance. The Naïve PAN approach exhibits different performance in the Fixed and Diverse scenarios. In the Fixed scenario, Naïve PAN outperforms BGP due to its greater number of path options. However, in the Diverse scenario, Naïve PAN demonstrates that simply having more

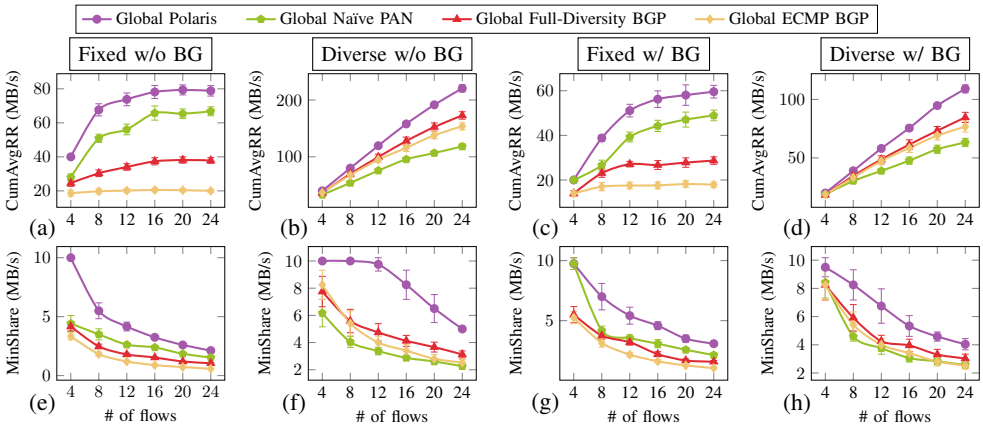

Fig. 12: Global MILP solution for the Cumulative Avg. Receiving Rate (top) and the Minimum Share (bottom) for the different path selection methods.

path options can actually hinder performance when end hosts choose paths naïvely. In Figure 11, we observe that Polaris outperforms all other approaches in terms of loss and jitter across all scenarios. Additionally, Polaris maintains latency comparable to BGP in the Diverse scenarios and lower than BGP in the Fixed scenarios, despite using paths that might potentially be longer than the shortest path. We observe that Naïve PAN, despite having the same diverse path options as Polaris, performs poorly due to its naïve path selection.

Figure 12 shows the global MILP solutions. We observe the same trend as in the simulation results, with Polaris consistently outperforming other methods in all scenarios, validating our simulation results.

VI. Related Work

One common mechanism for enhancing QoS in today's Internet is ECMP routing [35], [36], widely supported by major vendors [37]–[39]. At the inter-domain level, AS Path Manipulation techniques provide coarse-grained load balancing. Methods such as AS Path Prepending [40], route filtering [41], and community tags [42] influence routing decisions, while selective announcements and prefix de-aggregation modify path visibility [43], [44]. Additionally, BGP attributes like Local_Pref and MED affect path selection, though MED can lead to route oscillation and is often ignored [44], [45].

To increase path diversity, BGP extensions such as Add-Path [46] and BGP-XM [47] enable AS-level multipath routing, improving load balancing. However, these approaches introduce the overhead of maintaining multiple BGP sessions [28].

Transport-layer solutions such as Multipath TCP [2] and QUIC [3] allow end-hosts to use multiple interfaces (e.g., Wi-Fi and cellular) to balance load over a few first-hop options. Recent proposals leverage multiple prefixes for the same destination to enhance path diversity without modifying network routers. Examples of these systems are Painter [48] and Tango [49], with Tango significantly reducing one-way delay by bypassing default BGP routes.

Another approach to enhancing QoS through path optimization is application-aware traffic engineering, where in-band signaling enables applications to request specific network resources based on their requirements [50], [51].

The limitations of router-driven path selection have led to mechanisms that empower end hosts to choose inter-domain paths. Source routing [52] allows senders to specify routes but lacks scalability due to its reliance on full topology awareness. In contrast, path-aware networking (PAN) [15], [18], [41] offers a scalable alternative by enabling end-hosts to choose from a set of control-plane-managed paths [31].

To fully realize PAN's potential, a scalable and fine-grained inter-domain path optimization mechanism is needed. In this paper, we propose Polaris to address this requirement.

In Polaris, we use probes to determine the fair share of available but unused paths. While probing for path performance is common, most systems either measure only latency (gradient) or only the bottleneck capacity of paths, without accounting for current congestion or the fair share available to new flows [53], [54]. Moreover, systems with complex measurements [53] are suitable for coarse-grained evaluations over longer periods but are not useful for real-time applications.

VII. Conclusion

This paper presents Polaris, a feedback-driven path optimization system designed to enhance QoS for bandwidth-intensive and latency-sensitive traffic. Leveraging the source-based path selection in PANs, Polaris enables endpoints to optimize their experience by dynamically selecting paths based on real-time network conditions. Extensive simulations demonstrate that Polaris outperforms ECMP, achieving an average of 42% improvement in receiving rate and a 81% reduction in median loss, in coexistence with background traffic and diverse source-destination pairs for the flows.

In conclusion, Polaris demonstrates the potential of PANs to improve QoS for network applications, prompting further research to study different approaches for network state collection and distribution, as well as different approaches for enhancing end-to-end performance.

References

[1] B. Trammell, "Current open questions in path-aware networking," *IRTF, RFC 9217*, 2022.

[2] C. Paasch and O. Bonaventure, "Multipath TCP," *Communications of the ACM*, vol. 57, no. 4, pp. 51–57, 2014.

[3] Q. De Coninck and O. Bonaventure, "Multipath QUIC: Design and evaluation," in *Proceedings of the 13th international conference on emerging networking experiments and technologies*, 2017, pp. 160–166.

[4] B. Schlinker, H. Kim, T. Cui *et al.*, "Engineering egress with edge fabric: Steering oceans of content to the world," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 418–431.

[5] N. Feamster, J. Borkenhagen, and J. Rexford, "Guidelines for interdomain traffic engineering," *ACM SIGCOMM Computer Communication Review*, vol. 33, no. 5, pp. 19–30, 2003.

[6] S. Scherrer, A. Perrig, and S. Schmid, "The value of information in selfish routing," in *Structural Information and Communication Complexity: 27th International Colloquium, SIROCCO 2020, Paderborn, Germany, June 29–July 1, 2020, Proceedings 27*. Springer, 2020, pp. 366–384.

[7] J. Eo, Z. Niu, W. Cheng *et al.*, "OpenNetLab: Open platform for RL-based congestion control for real-time communications," in *Proceedings of the 6th Asia-Pacific Workshop on Networking*, 2022, pp. 70–75.

[8] D. Schepper *et al.*, "RFC 9330: Low latency, low loss, and scalable throughput (L4S) internet service: Architecture," 2023.

[9] W. Li, J. Liu, S. Wang, T. Zhang, S. Zou, J. Hu, W. Jiang, and J. Huang, "Survey on traffic management in data center network: from link layer to application layer," *IEEE Access*, vol. 9, pp. 38 427–38 456, 2021.

[10] K. Tsioutas and G. Xylomenos, "Audio delay in web conference tools," in *Web Audio Conference (WAC)*, 2022, pp. 1–6.

[11] S. F. Lindström, M. Wetterberg, and N. Carlsson, "Cloud gaming: A QoE study of fast-paced single-player and multiplayer gaming," in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*. IEEE, 2020, pp. 34–45.

[12] Y. Rekhter, T. Li, and S. Hares, "RFC 4271: A border gateway protocol 4 (BGP-4)," 2006.

[13] M. Luckie, A. Dhamdhere, D. Clark, B. Huffaker, and K. Claffy, "Challenges in inferring internet interdomain congestion," in *Proceedings of the 2014 Conference on Internet Measurement Conference*, pp. 15–22.

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

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

[16] P. B. Godfrey *et al.*, "Pathlet routing," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 4, pp. 111–122, 2009.

[17] C. Filsfils, S. Previdi, L. Ginsberg, B. Decraene, S. Litkowski, and R. Shakir, "Segment routing architecture," 2018.

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

[19] S. Tabaeiaghdaei, "ns-3-scion: A simulator for SCION integration with ns-3," https://gitlab.com/SeyedaliTaba/ns-3-scion, 2022, accessed: 2024.

[20] "SCION Seed emulator," https://perma.cc/M8VQ-S89D.

[21] M. Menth, R. Martin, and J. Charzinski, "Capacity overprovisioning for networks with resilience requirements," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 87–98, 2006.

[22] P. Gill, M. Schapira, and S. Goldberg, "A survey of interdomain routing policies," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 1, pp. 28–34, 2013.

[23] G. Carlucci, L. De Cicco, S. Holmer, and S. Mascolo, "Analysis and design of the google congestion control for web real-time communication (webrtc)," in *Proceedings of the 7th International Conference on Multimedia Systems*, 2016, pp. 1–12.

[24] S. Holmer, H. Lundin, G. Carlucci *et al.*, "A Google Congestion Control Algorithm for Real-Time Communication," Internet Engineering Task Force, Internet-Draft draft-ietf-rmcat-gcc-02, Jul. 2016.

[25] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson, "Real-time transport protocol," in *MCNC 2nd Packet Video Workshop*, vol. 2, 2003.

[26] Google, "Real-time communication for the web," 2024. [Online]. Available: https://webrtc.org/

[27] MDN contributors, "RTCPeerConnection: Browser compatibility," 2024. [Online]. Available: https://perma.cc/G2S8-MFV8

[28] C. Krähenbühl, S. Tabaeiaghdaei, C. Gloor *et al.*, "Deployment and scalability of an inter-domain multi-path routing infrastructure," in *Proceedings of the 17th CoNEXT*, 2021, pp. 126–140.

[29] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005.

[30] S. Deering and R. Hinden, "Rfc 8200: Internet protocol, version 6 (ipv6) specification," 2017.

[31] L. Chuat, M. Legner, D. Basin, D. Hausheer, S. Hitz, P. Müller, and A. Perrig, "The complete guide to SCION," *Information Security and Cryptography*, 2022.

[32] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," *nature*, vol. 393, no. 6684, pp. 440–442, 1998.

[33] "WebRTC code repository," https://perma.cc/WUW9-ZMFU.

[34] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2024. [Online]. Available: https://www.gurobi.com

[35] C. Hopps, "Analysis of an equal-cost multi-path algorithm," 2000.

[36] N. Katta, M. Hira, C. Kim, A. Sivaraman, and J. Rexford, "Hula: Scalable load balancing using programmable data planes," in *Proceedings of the Symposium on SDN Research*, 2016, pp. 1–12.

[37] Juniper Networks, "Load balancing for a BGP session," February 2024, [Online] Available: https://perma.cc/7RK8-FUNY.

[38] Cisco Systems, "BGP best path selection algorithm," July 2023, [Online] Available: https://perma.cc/6NCD-YU3W.

[39] Huawei Technologies, "Example for configuring BGP load balancing," August 2023, [Online] Available: https://perma.cc/WK4C-XPH2.

[40] P. Marcos, L. Prehn, L. Leal, A. Dainotti *et al.*, "AS-Path prepending: there is no rose without a thorn," in *Proceedings of the ACM Internet Measurement Conference*, 2020, pp. 506–520.

[41] W. Xu and J. Rexford, "Miro: Multi-path interdomain routing," in *Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications*, pp. 171–182.

[42] F. Streibelt, F. Lichtblau, R. Beverly *et al.*, "BGP communities: Even more worms in the routing can," in *Proceedings of the Internet Measurement Conference 2018*, 2018, pp. 279–292.

[43] P. Amaral, E. Silva, L. Bernardo, and P. Pinto, "Inter-domain traffic engineering using an AS-level multipath routing architecture," in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–6.

[44] B. Quoitin, C. Pelsser, L. Swinnen, O. Bonaventure, and S. Uhlig, "Interdomain traffic engineering with BGP," *IEEE Communications magazine*, vol. 41, no. 5, pp. 122–128, 2003.

[45] T. G. Griffin and G. Wilfong, "Analysis of the MED oscillation problem in BGP," in *10th IEEE International Conference on Network Protocols, 2002. Proceedings.* IEEE, 2002, pp. 90–99.

[46] A. Retana, "Advertisement of multiple paths in BGP: Implementation report," 2015.

[47] J. M. Camacho, A. García-Martínez, M. Bagnulo, and F. Valera, "BGP-XM: BGP extended multipath for transit autonomous systems," *Computer Networks*, vol. 57, no. 4, pp. 954–975, 2013.

[48] T. Koch, S. Yu, S. Agarwal, E. Katz-Bassett, and R. Beckett, "Painter: Ingress traffic engineering and routing for enterprise cloud networks," in *Proceedings of the ACM SIGCOMM 2023 Conference*, pp. 360–377.

[49] H. Birge-Lee, M. Apostolaki, and J. Rexford, "It takes two to tango: cooperative edge-to-edge routing," in *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, 2022, pp. 174–180.

[50] T. Miyasaka, Y. Hei, and T. Kitahara, "Networkapi: An in-band signalling application-aware traffic engineering using srv6 and ip anycast," in *Proceedings of the Workshop on Network Application Integration/CoDesign*, 2020, pp. 8–13.

[51] S. H. Mortazavi *et al.*, "Earlybird: Automating application signalling for network application integration in datacenters," in *Proceedings of the ACM SIGCOMM Workshop on Network-Application Integration*, 2022, pp. 40–45.

[52] S. A. Jyothi, M. Dong, and P. B. Godfrey, "Towards a flexible data center fabric with source routing," in *Proceedings of the 1st ACM SIGCOMM Symposium on Software Defined Networking Research*, 2015, pp. 1–8.

[53] H. V. Madhyastha *et al.*, "iplane: An information plane for distributed services," in *Proceedings of the 7th symposium on Operating systems design and implementation*, 2006, pp. 367–380.

[54] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, 2003, pp. 39–44.