Colibri: A Cooperative Lightweight Inter-domain Bandwidth-Reservation Infrastructure

Giacomo Giuliari
ETH Zürich

Dominik Roos
Anapaya Systems

Marc Wyss
ETH Zürich

Juan Angel García-Pardo
ETH Zürich

Markus Legner
ETH Zürich

Adrian Perrig
ETH Zürich, Anapaya Systems

Abstract

Guarantees for traffic traversing the public Internet are hard to come by, as service-level agreements are typically only available for traffic within a single autonomous system or towards direct neighbors. This deficiency leads to unpredictable performance already under normal conditions and can cause outages in the face of network-level distributed-denial-of-service (DDoS) attacks. In this paper, we present an architecture achieving guaranteed bandwidth properties for global inter-domain network traffic. The control plane of our architecture is based on a distributed server infrastructure, while the data plane enables efficient packet forwarding on per-flow stateless routers. Our implementation demonstrates the technical feasibility and scalability of the design.

CCS Concepts

• **Security and privacy** → **Denial-of-service attacks**; • **Computer systems organization** → *Availability*; • **Networks** → *Network protocols.*

Keywords

Bandwidth Reservations, DDoS Attacks, SLOs

ACM Reference Format:
Giacomo Giuliari, Dominik Roos, Marc Wyss, Juan Angel García-Pardo, Markus Legner, and Adrian Perrig. 2021. Colibri: A Cooperative Lightweight Inter-domain Bandwidth-Reservation Infrastructure. In *The 17th International Conference on emerging Networking EXperiments and Technologies (CoNEXT '21), December 7–10, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 15 pages. https://doi.org/10.1145/3485983.3494871

1 Introduction

With the consolidation of computation in data centers, the tension created by co-location of multiple tenants sharing resources has resulted in numerous separation mechanisms to provide resource guarantees. Consequently, service-level objectives (SLOs) have been established at various levels: the CPU provides isolation to prevent information leakage [52, 63]; hypervisors can be configured to guarantee minimal amounts of resources to individual virtual machines [60]; and the local network optimizes communications within the data center [42]. The final frontier is the Internet, raising a fundamental research question: can we provide SLOs in an open Internet—even under the threat of denial-of-service (DoS) attacks—in a scalable manner? While centralized control facilitates SLOs (e.g., inside a data center), there are numerous interacting entities on the heterogeneous Internet.

We consider three important areas that require scalability, given an increase in network size, number of flows under an SLO, and traffic volume: (1) control-plane scalability studies the growth of communication, processing, and storage overhead for admission control and resource allocation; (2) data-plane scalability considers the processing and storage overhead for packet forwarding, including authentication, monitoring, and policing to detect or prevent adversarial actions; and (3) management scalability captures the human effort to configure and administer an autonomous system (AS).

Past efforts cannot handle adversarial settings, where some of the actors behave maliciously, or fall short to achieve scalability in one or multiple of these three areas. Of the many systems proposed to achieve guarantees for global communication, the two archetypal and most widely deployed architectures are *Integrated Services* (IntServ) and *Differentiated Services* (DiffServ). They constitute the two extreme points in the trade-off spectrum between scalability and strength of offered guarantees, and we will therefore use them as representatives of other systems that employ the same ideas and suffer from the same shortcomings (see §8 for a survey).

IntServ provides very strict guarantees on the communication parameters through end-to-end reservations, but is known to scale poorly in all three areas because of the complex decisions that must be made for processing the Resource Reservation Protocol (RSVP) requests and the amount of *per-flow* state that on-path routers have to keep. DiffServ, on the other hand, provides hosts with a way to divide their traffic into a number of *classes* according to the application's requirements, indicated in the IP packet's type of service (ToS) header field [8]. Packet scheduling and prioritization to enforce the desired service level is then delegated to the on-path routers. DiffServ scales well with respect to the three scalability areas, as the only information needed for the traffic differentiation is carried in the IP packet header. Unfortunately, the guarantees provided by DiffServ are weak, as they lack signaling between the entities on the path, which translates to per-hop forwarding strategies that are blind to the state of the downstream networks.

These two examples reveal the apparent trade-off that all existing resource-reservation architectures are forced to make: strong guarantees overload the network precluding Internet-wide scalability, while in light-weight systems such guarantees are sacrificed. Further, neither DiffServ nor IntServ protect against adversarial action:

CoNEXT '21, December 7–10, 2021, Virtual Event, Germany

ACM ISBN 978-1-4503-9098-9/21/12…$15.00
https://doi.org/10.1145/3485983.3494871

an adversary can spoof protocol messages, over-use its reservations, or interfere with communications between other entities.

In this work, we present Colibri, a concrete design and implementation of a *collaborative lightweight inter-domain bandwidth-reservation infrastructure* for the global Internet that overcomes the scalability–quality trade-off and can provide SLOs in the form of worst-case minimum bandwidth guarantees. Colibri is made possible through the confluence of several recently developed technologies: path-aware networking providing path choice and stability [39]; a fair and scalable resource-allocation system [62]; a global symmetric-key distribution system enabling efficient per-packet authentication [43]; an efficient replay-suppression system [32]; and an overuse-flow-detection system [44, 64]. These systems and their role are summarized in Table 1, and described in detail in §2.

Scalability of the system is achieved through hierarchical decomposition at several levels:

- **Topological**: reservations are split between the Internet core and local AS hierarchies, reducing the global coordination effort.
- **Temporal**: intermediate-term AS-to-AS reservations carry most of the computational overhead, which is thus amortized over time. Based on these, short-term reservations are inexpensively set up for host-to-host communications.
- **Monitoring**: stateful reservation monitoring is performed at the edge. At transit and core ASes—where the number of flows is too high to be processed statefully—neighbor-based probabilistic monitoring is used.
- **Policing**: each AS is responsible for preventing its hosts from overusing a reservation; this in turn is enforced by other ASes: per-packet authentication and duplicate suppression allow unambiguously attributing misbehavior to the offending AS and thus punishing it—e.g., by cancelling existing reservations and declining future ones.

The core property we seek in Colibri is to provide minimum bandwidth guarantees between any pair of ASes on a given path, irrespective of distributed-denial-of-service (DDoS) attacks or other allocations. In the common (non-attack) case, allocations will be much higher, but worst-case guarantees enable Internet-scale SLOs. This work is thus the first to present a concrete, scalable system design to solve the open problem of Internet-scale SLOs, bridging the gap between the data center and the end user, and completing the final step towards end-to-end-protected services. Our implementation of Colibri shows that it achieves the *three notions of scalability*—control plane, data plane, and management—and can operate securely in a federated Internet in the presence of adversaries: the control-plane services can process 2000 reservations per second on a single core, while our software router can authenticate and forward 34 *million* packets per second (312 Gbps for 1 KB packets) without specialized equipment. Our implementation of Colibri is now part of SCIONLab [29].[1]

2 Background and Enabling Technologies

Traditionally, inter-domain bandwidth reservation systems were fundamentally limited by the following challenges, each of which needs to be resolved to achieve a viable system: (1) per-flow state in the data plane, (2) lack of path stability, (3) lack of path diversity, (4) flows overusing their reservation, (5) framing attacks by on-path routers through packet replay or alteration, (6) framing attacks by off-path entities using source-address spoofing, and (7) DoS attack on routers' packet-authentication system. In the remainder of this section, we describe the developments in networking that are at the basis of Colibri and allow to overcome both performance and security challenges. Table 1 presents a summary of these points.

[1] https://github.com/netsec-ethz/scion/pull/101

Table 1: Challenges faced by resource-reservation systems, and the technical solutions used in Colibri.

Challenge	Enabling technology	
Per-flow state in the fast path	PCFS	§2.1
Re-convergence changes reservation path Path-hijack invalidates reservation★	Path stability	§2.1
No reservation space available on path On-path adversary★	Path choice	§2.1
Large number of reservations	ISDs and segment types	§2.2
Framing attack with spoofed packets★	Per-packet source auth.	§2.3
Authentication overhead of signatures	Symmetric-key auth.	§2.3
Framing or DoS through packet replay★	Duplicate suppression	§2.3
Over-use of legitimate reservation★	Prob. monitoring	§2.3
Difficult admission decisions	Reservation hierarchy	§3.1

★Adversarial action.

Adversary Model. Regarding attacks, we consider two types of adversaries:

- An *on-path adversary* is a malicious entity residing in or controlling one of the ASes on the forwarding path of the packet. It is able to drop, alter, or replay the packets that traverse the AS, or inject forged packets.
- An *off-path adversary* is not directly located on the forwarding path, but can still attempt to influence the communication by hijacking traffic (therefore becoming an on-path adversary), source-spoofing, creating congestion on the forwarding paths, or by flooding the control plane with bogus announcements, preventing convergence.

We provide an analysis of attacks and how they are prevented in §5.

2.1 Path-Aware Networking

Path-aware networking is embraced by several proposed future Internet architectures such as Platypus [40, 41], PoMo [12], NIRA [68], Pathlets [21], SCION [39, 73], and NEBULA [2]. Their defining trait is that path information such as traversed ASes and measurements are disclosed to end hosts, which obtain some control over the forwarding path of their packets. This is often implemented using *packet-carried forwarding state* (PCFS), where forwarding information is included in the packet header. Path-aware networks provide two essential properties for a resource-reservation architecture: path stability and path choice.

Path Stability. Since routing decisions are decoupled from the dissemination of path information, these networks do not suffer from the long convergence times that affect path-vector protocols such as the Border Gateway Protocol (BGP). Reservation guarantees are hard to achieve and maintain if communication has to wait for re-convergence after every routing event. Moreover, the PCFS protects forwarding from routing attacks attempted by off-path adversaries, preventing the denial of reservations by means of BGP

hijacking or similar attacks. Together, these properties ensure that AS-level paths, and any reservations on them, are stable in time and cannot be affected by off-path entities.

Path Choice. In the current Internet, the routing protocol selects a single forwarding path; if the path is congested there is no reliable way to reroute around the congestion. In path-aware architectures, hosts can choose to use one (or several) of the paths discovered by the routing protocol. This feature not only allows for much more fine-grained routing optimization—as shown by Sobrinho et al. [51]—but enables multiple options for reservation architectures: in case the reservation request cannot be met on the first path, Colibri can attempt to make a reservation on the alternative paths, which increases the probability of a successful reservation. Multiple reservations across multiple paths can also be used, e.g., by a multipath transport protocol.

2.2 Isolation Domains and Path Segments

Colibri builds on SCION as its underlying network architecture. Besides its mature development stage [29], its open-source implementation [45], and its real-world deployment [1, 28, 54], we chose SCION particularly for its hierarchical decomposition of ASes to improve the scalability of the system.

SCION groups ASes into isolation domains (ISDs) [39] and distinguishes between *core ASes* and *non-core ASes*. Core ASes are typically larger entities that manage the ISD's trust roots, and provide connectivity to other ISDs. The routing process is then split into (i) an intra-ISD process discovering up-segments (from non-core ASes to core ASes) and down-segments (from core ASes to non-core ASes), and (ii) an inter-ISD process discovering core-segments between any pair of core ASes. At most one up-, one core-, and one down-segment are then combined by source hosts to construct full end-to-end paths. Thus, SCION tames the (worst-case exponential) complexity of global path discovery by decomposing it into three separate routing sub-problems, which are further simplified by existing customer-provider relationships between ASes. To avoid the inefficiency of hierarchical routing, packets can be routed through *shortcuts* between up- and down-segments. Finally, to uniquely identify an inter-domain connection, SCION uses *interfaces*, which need to be unique within an AS and can be defined by each AS independently. Paths are represented by ingress–egress interface-pairs for each on-path AS.

We also leverage SCION's keying infrastructure to achieve per-packet authentication, as described next.

2.3 Reservation Protection

In order to assign, enforce, and bill reservations correctly without per-flow state on routers, a data packet must carry cryptographically protected information that allows an on-path AS to (i) verify that the packet is sent over a valid reservation, and (ii) attribute the packet to the source. As every packet must be checked individually, efficient cryptographic mechanisms are vital. In addition, control-plane messages must also be efficiently authenticated to avoid additional attack vectors (e.g., signature flooding), and prevent denial-of-capability (DoC) attacks [7]. Recent advances, which we present in the following, enable these properties statelessly and at line rate in path-aware networks.

DRKey. The dynamically-recreatable-key (DRKey) infrastructure allows Colibri to achieve line-rate per-packet authentication of control-plane packets. At a high level, DRKey provides a mechanism for ASes to derive symmetric keys shared with any other AS on the fly, based on pseudo-random functions [43]. In the following, we provide a summary of DRKey's most important components.

An AS A has a secret value K_A, that is used as a key for a pseudo-random function (PRF) to derive an AS-level key used in the communication with another AS B:

$$K_{A\to B} = \mathsf{PRF}_{K_A}(B). \tag{1}$$

While these are symmetric keys, there is an asymmetry between the two ASes, which is indicated by the arrow: the first AS A is able to derive the key on the fly by evaluating the PRF—which is faster than a memory lookup—while the second AS B needs to fetch $K_{A\to B}$ with an explicit request to A's key server, protected by public-key cryptography. As the validity period of these keys is on the order of a day, they can be fetched ahead of time and only need to be infrequently renewed. Thus, DRKey provides a way to perform source authentication: the source computes message-authentication codes (MACs) for all on-path ASes, which can then be checked by on-path ASes without per-source state and using only highly efficient symmetric cryptography. [2] Statelessness is essential to the availability of secure source authentication, as adversaries cannot leverage state exhaustion or state inconsistencies to DDoS the authentication endpoints.

Monitoring and Policing. Even if packets are source-authenticated, the reservation architecture is still susceptible to replay attacks: an on-path adversary can capture an authenticated packet and send it repeatedly at a higher rate than what the reservation allows, thus at the same time causing congestion and framing the honest source. Because of this, an efficient *duplicate-packet-suppression* system with minimal state requirements is needed [32]. Further, to prevent overuse of reservations, additional *monitoring and policing* systems need to be used, i.e., systems the ASes use in order to make sure that flows originating from, traversing, or ending inside their networks do not use more bandwidth than allocated. We integrate such systems into Colibri in §4.8.

Finally, *time synchronization* between ASes is important to schedule start- and end-times of reservations, and to perform duplicate detection and traffic monitoring. Several different systems can be used to achieve this goal [6, 24, 35], and even in adversarial settings [4, 5]. In this paper, we assume that all ASes are synchronized within ±0.1 seconds.

3 Colibri Overview

In this section, we provide a high-level description of the Colibri system. The core ideas are summarized in Fig. 1.

3.1 General Concepts and Intuition

With Colibri, hosts can make short-term bandwidth reservations, similarly to IntServ, to protect traffic end-to-end. However, as central parts of the Internet—the "core"—could potentially serve an enormous number of such end-to-end reservations (EERs), the decision whether or not to allocate resources must be very efficient.

[2] In practice, protocol- and host-specific keys are derived from the key $K_{A\to B}$; we disregard these details in this paper in the interest of readability.

To simplify and speed up this process, Colibri bases the allocation of EERs on a second type of bandwidth reservations called segment reservations (SegRs). SegRs have longer validity periods than EERs and are fewer in number, as they are established between ASes instead of end hosts. To increase efficiency, we offload the burden of computationally expensive operations to SegR allocations; hosts can then use the pre-computed SegRs to quickly set up EERs. Figuratively, SegRs can be thought of as "tubes", which are later filled by EERs. We further restrict the number of SegRs by leveraging the decomposition of paths into segments of the underlying path-aware network architecture (§3.3). Forwarding is not impacted by reservation operations, as they take place in dedicated Colibri services, outside the routers' fast path. End hosts can then request SegRs from their AS (§3.2) and assemble them to cover the complete path to the destination's AS. The resulting SegRs can be used to request EERs (§3.3). This additional level of reservations allows an AS on the path of an EER request to immediately grant or refuse the request based on the available bandwidth in the respective SegR.

To avoid per-flow state in the fast path and to verify the legitimacy of a packet in the data plane, Colibri protects the packet-carried forwarding state (PCFS) through cryptographic tags. The keys to compute these tags, called hop authenticators (HopAuths), are generated with pseudo-random functions by on-path ASes and sent—in encrypted form—to the initiator during the reservation process. At forwarding time, each on-path AS dynamically re-creates the authentication key and verifies the tag, efficiently authenticating the packet and the reservation. Thus, Colibri overcomes another major issue of previously-proposed bandwidth-reservation systems: the need to maintain per-flow state on routers.

3.2 Infrastructure

An AS deploying Colibri needs to perform three main tasks: (i) control the admission procedure, (ii) manage SegRs, and (iii) rate-limit EERs of their end hosts. Tasks (i–ii) are handled by the Colibri service, task (iii) by the Colibri gateway.

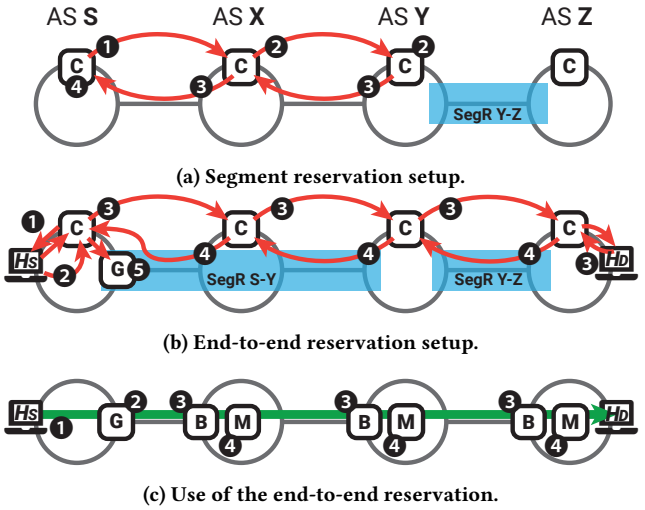
Colibri Service. Every AS runs a *Colibri service* (CServ), which handles all control-plane-related tasks within an AS:

- The CServ requests and renews SegRs according to expected traffic requirements. Since link utilization often exhibits repeating patterns over time, an AS can forecast future requirements and reserve appropriate bandwidth for segments in advance.
- It provides previously established and registered SegRs to end hosts and remote CServs.
- It handles all reservation setup and renewal requests and performs the necessary admission calculations.

To improve scalability, the CServ can be distributed: a central service is used to keep track of SegRs, while the handling of EERs is delegated to sub-services close to or at the border routers. This is described in further detail in Appendix D.

Colibri Gateway. An AS is held accountable by other ASes for misbehavior of its end-host customers. Therefore, it is crucial that all Colibri traffic originating from end hosts within an AS pass through a *Colibri gateway*, which

- performs stateful monitoring and rate limiting for all EERs;
- embeds cryptographic tags into packet headers, allowing border routers to authenticate the source and data in packet headers.



(a) Segment reservation setup.

(b) End-to-end reservation setup.

(c) Use of the end-to-end reservation.

Figure 1: Overview of the Colibri system. C are CServs, G is the Colibri gateway, B are border routers, M are traffic monitors. The description is provided in §3.3 and §3.4.

To ensure consistent processing and prevent attacks by end hosts, every EER needs to be tied to one specific gateway.

End-Host Networking Stack. The path choice (§2.1) enabled by path-aware network architectures requires a more sophisticated end-host networking stack. Colibri modifies the *SCION Daemon* [45] to enable an application to explicitly request and renew EERs.

In principle, any transport protocol can be used with Colibri, as the gateway drops packets if the guaranteed bandwidth is exceeded and thus provides feedback to the congestion-control algorithm. Still, a tighter integration with the transport protocol is necessary to reap the full benefits of Colibri. For example, in QUIC, it is straightforward to disable congestion control and set the sending rate to the reserved bandwidth.

3.3 Control Plane

Colibri's control plane manages the selection, creation, and renewal of (i) SegRs and (ii) EERs, which we discuss individually. Both SegRs and EERs are unidirectional, which reflects traffic demand in the Internet: while some ASes mainly send traffic (e.g., CDNs), others predominantly receive data (e.g., eyeball ASes). Renewals are discussed in detail in §4.2.

Segment Reservations. SegRs are intermediate-term reservations made between two ASes and are valid for approximately five minutes. This duration has been chosen as a compromise between excessive overhead for short intervals and insufficient flexibility for long intervals. To improve the scalability of the SegRs and avoid setting up SegRs between any pair of the currently over 70 000 ASes [10], we leverage the concept of ISDs described in §2.2. Following this structure among ASes, Colibri distinguishes three types of SegRs: up-SegRs (from non-core ASes towards core ASes inside one ISD); down-SegRs (from core ASes towards non-core ASes inside one ISD); and core-SegRs (among core ASes, potentially in different ISDs). SegRs are always initiated by the first AS on the segment. For down-SegRs, the first AS only sets up a SegR upon an explicit request by the last AS.

An AS can decide which SegRs to request based on historical data or traffic predictions. It selects segments generated by the underlying path-aware routing architecture and sends a *segment-reservation setup request* (SegReq) (❶ Fig. 1a) specifying requirements for the reservation such as requested minimum bandwidth. Each AS on the segment can calculate how much bandwidth can be granted; if this is higher than the requested minimum, it records this reservation locally. It then updates the request with the granted amount of bandwidth and forwards it to the next AS on the segment (❷ Fig. 1a). The last AS sends a reply via the same segment to the SegReq initiator (❸ Fig. 1a). If the request was successful (each AS granted more than the minimum amount of bandwidth), each AS locally stores the final amount of bandwidth granted and includes a cryptographic token in the response that allows the request initiator to use the segment for Colibri traffic (❹ Fig. 1a; see §4.5 for details). In case of an unsuccessful request, the ASes clean up their temporary reservations and the initiator can determine the location of potential bottlenecks on the segment. All important information is authenticated using DRKey as discussed in §4.5.

A central mechanism in this procedure is the admission algorithm that determines how much bandwidth can be granted; this will be discussed in §4.7.

End-to-End Reservations. EERs are short-term reservations between two end hosts, with a fixed validity period (16 seconds in our implementation). This interval is chosen by considering the trade-off between the needs to (i) amortize the EER initialization overhead, (ii) minimize the number of idle reservations, and (iii) maximize the flexibility to adapt to changing traffic patterns.

An end host H_S intending to communicate with another host H_D must first obtain one or several SegRs that can be combined to form a connected path between the ASes of H_S and H_D. H_S obtains these reservations from the CServ (❶ Fig. 1b), in a similar way as it would otherwise obtain a path to H_D in the underlying path-aware network architecture (this is described in further detail in Appendix C). It can combine SegRs to obtain a complete path and send an *end-to-end–reservation setup request* (EEReq) (❷ Fig. 1b), which has a similar format as the SegReq described above. Again, each AS on the path decides how much bandwidth can be granted and forwards the request packet (❸ Fig. 1b). In contrast to the SegReq, this decision is simple: the intended bandwidth is granted if there is sufficient available bandwidth in the underlying SegR (see §4.7 for further details). It falls to the AS in which H_S is situated to set limits on the maximum bandwidth that H_S can request. This intra-AS admission policy can be defined by each AS independently.

In addition to all ASes on the path, the destination H_D also needs to grant the request. A response is generated in a similar way as for the SegReq: each AS on the path updates the reservation information and (if the request was successful) includes the necessary HopAuths, in the response (❹ Fig. 1b). The HopAuths are stored at the source AS's Colibri gateway (❺ Fig. 1b), and the rest of the response is forwarded to H_S, who can then start using the EER to send traffic on that path.

The host can base the amount of requested bandwidth on the expected traffic, e.g., the known bitrate of a video stream. In cases of less regular traffic, the host may need to employ heuristics to determine how much bandwidth to request. Both control and data traffic are handled by the end-host networking stack.

3.4 Data Plane

Packet Creation and Forwarding. After setting up an EER, an end host can use the reservation to send traffic to the specified destination. The end host includes the Colibri routing information in the packet header and sends it to the Colibri gateway of its AS (❶ Fig. 1c). There, the AS performs stateful traffic monitoring and uses the HopAuths that were set up during the request process to calculate per-packet message-authentication codes (MACs) for each AS on the path that at the same time (i) provide source authentication, and (ii) prove that the Colibri path was previously authorized by that AS (❷ Fig. 1c). The details are described in §4.5.

On-path routers check the corresponding MAC by recalculating it locally (❸ Fig. 1c). It is crucial to note that they *do not require per-flow state* for this: all necessary keys can be derived on the fly from a single AS-specific secret value. Furthermore, no inter-domain routing-table lookup is necessary as the routing information is already included in the packet header. Thus, after authenticating the packet, the router simply forwards the packet to the destination host or next border router.

Traffic Split. Not all traffic can directly benefit from Colibri EERs: First, reservations are only useful for flows of some minimum size. Second, in some cases the communication requires replies that are however not large enough to need their own reservation—e.g., the acknowledgments for a video stream. Since reservations are unidirectional, these must be sent as best-effort traffic.

Therefore, Colibri reserves a fixed minimum bandwidth (e.g., 20 % of the full link capacity) for best-effort traffic. The remaining bandwidth is split further between Colibri control traffic on SegRs (5 %)—which is used for protected SegRs renewal and EERs establishment, as detailed in §4.4—and traffic over EERs (75 %). Note that no bandwidth is wasted: in case SegRs or EERs are underutilized, the remaining bandwidth can be used for best-effort traffic. In practice, queuing techniques such as *priority queuing* or *class-based weighted fair queuing* [46] can provide this separation on a shared physical infrastructure, see Appendix B for details.

Monitoring and Policing. Using cryptographic MACs in packet headers guarantees that an end host can send traffic only over an authorized reservation. However, a host can over-use a legitimate reservation and exceed the allocated bandwidth. To avoid misuse of reservations, an AS running Colibri performs two tasks: (i) it ensures that its own customers respect the bandwidth of their EERs (❷ Fig. 1c), and (ii) it monitors traffic from other ASes to detect overuse of EERs or SegRs (❹ Fig. 1c). This second task provides an incentive for ASes to comply and perform monitoring properly, as they are held accountable for the behavior of their customers. Further details are provided in §4.8.

4 Architecture Details

4.1 AS Types

For EERs, we distinguish between four types of ASes depending on their position (they correspond to the ASes in Fig. 1): The *source AS* is where H_S is located (AS S); a *transit AS* is an on-path AS in the middle of a SegR (AS X); a *transfer AS* is at the joint of two SegRs and, according to the structure of segments (§2.2), necessarily a core AS (AS Y); the *destination AS* is where H_D is located (AS Z).

4.2 Reservation Versions and Renewal

Reservations are created with short lifetimes to ensure that they do not occupy bandwidth for longer than necessary. If the initiator of a reservation intends to keep using it beyond its expiration time, it can issue a renewal request to extend the reservation, and possibly adjust the bandwidth to shifting traffic demands. Expiration and renewal work slightly differently for EERs and SegRs.

EERs. Since EERs are frequently renewed, it is crucial that they can be renewed without service interruptions. For this purpose, the design allows multiple *versions* of the same EER to exist simultaneously. The initiator can renew the reservation ahead of time to obtain a new version with a later expiration time, which allows for a seamless transition between two versions of an EER. To enhance scalability, CServs can rate-limit the amount of renewal requests for an EER (e.g., to one per second). EERs automatically expire, and there is no mechanism to remove them earlier. The gateway generally uses a single version (the latest one) to send traffic. However, even using multiple versions simultaneously does not increase bandwidth use, as all versions are mapped to the same reservation ID in the traffic monitor (see §4.8).

The initiator of an EER is not the only entity that could be interested in adjusting the reserved bandwidth. An AS on the path may also wish to reduce an EER's bandwidth, e.g., if it receives an increasing number of contending requests. As in the setup procedure, during a renewal request all on-path ASes can specify the amount of bandwidth they are willing to grant, enabling ASes to quickly adapt to changes in demand without interrupting service over existing reservations.

SegRs. Colibri's design ensures that EERs are not affected by a version change of their underlying SegR. In contrast to EERs, only a single version of a SegR can exist at any time and a pending version obtained through a renewal request must be activated explicitly using a separate request. Making this switch explicit allows ASes to precisely control the time to change to a new version and ensure that no over-allocation with EERs can occur.

4.3 Colibri Packets

Packet Format and Header Fields. A Colibri packet traversing AS_0–AS_ℓ has the format

$$\begin{aligned} \textsc{Packet} = (&\textsc{Path} \parallel \textsc{ResInfo} \parallel \textsc{EERInfo} \parallel Ts \parallel \\ &V_0 \parallel \ldots \parallel V_\ell \parallel Payload), \end{aligned} \tag{2a}$$

$$\textsc{Path} = ((In_0, Eg_0) \parallel \ldots \parallel (In_\ell, Eg_\ell)), \tag{2b}$$

$$\textsc{ResInfo} = (SrcAS \parallel ResId \parallel Bw \parallel ExpT \parallel Ver), \tag{2c}$$

$$\textsc{EERInfo} = (SrcHost \parallel DstHost), \tag{2d}$$

where V_i is the hop validation field (HVF) of AS_i, which authenticates parts of the packet header; Path is a list of ingress–egress interface-pairs; *SrcAS* is the source AS; *SrcHost* and *DstHost* are end-host addresses, which are unique inside their AS; *Bw*, *ExpT*, and *Ver* denote the reservation bandwidth, expiration time, and version, respectively; and *Ts* is a high-precision timestamp relative to *ExpT* and uniquely identifies the packet for the particular source. The EERInfo field is only used for EERs data-plane packets.

This packet format is used for all Colibri control- and data-plane traffic. In the case of SegRs, AS_0-AS_ℓ denote the ASes that constitute the particular segment, for EERs they correspond to the ASes on the end-to-end path.

Reservation IDs. The reservation ID (*ResId*) needs to be unique per source AS. Therefore, the CServ increases the *ResId* for every new SegR or EER. Thus, the pair (*SrcAS*, *ResId*) uniquely identifies every SegR and EER globally.

4.4 Control-Plane Messages

The control plane comprises setup and renewal requests for SegRs and EERs. The initiator AS relies on path segments of the underlying path aware Internet architecture for the Path and AS IDs.

SegR Setup and Renewal Procedure. A SegR setup is initiated by the source AS using best-effort traffic, where the packet's payload consists of the Path, the ResInfo, and the minimum acceptable bandwidth. All on-path ASes add further data to the payload, which is used on the backwards path to calculate and agree on the allocation size. A SegR renewal can be made over the existing SegR. Because SegR renewal packets already contain the Path, *SrcAS*, and *ResId*, the source AS only needs to specify the new *Bw* (and new minimum bandwidth), *ExpT*, and *Ver* in the payload. Further, during SegR renewal, on-path ASes can also re-negotiate the bandwidth granted to the *SrcAS*. Thus, they can quickly adapt to shifting traffic demands and policy changes.

EER Setup and Renewal Procedure. Every EER is established over one, two, or three SegRs. For an EER setup, the source AS creates a Colibri packet for the first SegR, and adds the EER Path, the EER ResInfo, the EERInfo, plus the *ResId*s of all segments to the payload. If the EER is intended to be created over more than one SegR, each transfer AS copies the payload of the previous SegR's Colibri packet to a new Colibri packet for the following SegR. This is possible because the transfer AS can look up all the necessary information for the following SegR based on the corresponding *ResId*. This way each AS on the end-to-end path obtains the EER setup information in the payload, based on which they either grant or deny the requested allocation. Additionally, the request is also forwarded using intra-AS communication from the CServ to the destination end host specified in the *DstHost* field, who also has to explicitly accept the EER request.

Similar to SegR renewals, an EER renewal can be made over the existing EER, where the source AS only needs to specify the new EER *Bw*, *ExpT*, and *Ver* in the payload.

4.5 Packet Authentication

Authentication of Control-Plane Messages. To authenticate the payload of control-plane packets, Colibri leverages the DRKey mechanism (§2.3). Thereby, the source AS calculates a MAC over the payload for each on-path AS, using the key $K_{AS_i \to SrcAS}$. AS_i can then efficiently recompute this key on the fly and verify the authenticity of the payload. The same key is used to authenticate the information that AS_i itself adds to the payload.

Segment Reservations. The only packets that are sent over SegRs are *control-plane* packets (SegR renewal and EER setup requests), the payload of which is authenticated using DRKey as discussed above. Nevertheless, it is important that routers can statelessly verify the validity of a SegR, and authenticate forwarding information in the header. Therefore, during every SegR setup and

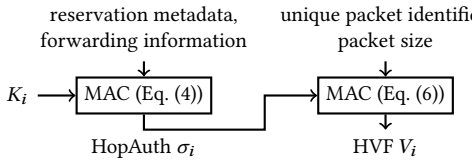


Figure 2: Two-step MAC calculation for the *i*th HVF of a data packet sent on an EER.

renewal, each on-path AS calculates a token in the form of a MAC over reservation metadata and forwarding information, using the AS's secret key K_i, truncated to the first ℓ_{hvf} bytes:

$$V_i^{(\text{S})} = \text{MAC}_{K_i}\left(\textsc{ResInfo} \parallel (In_i, Eg_i)\right)[\![0{:}\ell_{\text{hvf}}]\!]. \tag{3}$$

These tokens are returned to the SegReq initiator and stored in its local database. All the information included in this calculation is explicitly contained in the header of packets sent over this reservation (see Eq. (2)), enabling the router to validate the token on the fly during packet processing without requiring per-reservation state.

While this use of relatively short static MACs (we use $\ell_{\text{hvf}} = 4$) in principle enables the reuse of once-observed or brute-forced tokens [33], this is not problematic in practice due to the short lifetime of reservations. Also, it is not necessary to "chain" the forwarding information of different ASes, which is done in SCION [39] and EPIC [33], to prevent path splicing, as Colibri tokens explicitly include the globally unique combination of *SrcAS* and *ResId* (§4.3).

End-to-End Reservations. The authentication system for data packets on EERs must, simultaneously,

- be highly efficient, to support rates of hundreds of Gbps;
- avoid any per-reservation state on routers; and
- enable each on-path AS to conclusively attribute the packet to a reservation and the source AS.

We achieve these properties through the following two-step authentication mechanism, sketched in Fig. 2.

First, during EER setup or renewal, each on-path AS calculates a HopAuth, akin to the SegR token in Eq. (3) but without truncation:

$$\sigma_i = \text{MAC}_{K_i}\left(\textsc{ResInfo} \parallel \textsc{EERInfo} \parallel (In_i, Eg_i)\right). \tag{4}$$

Similar to SegRs, all the information included in this calculation is an explicit part of the data-plane packet header enabling a router to compute the HopAuth on the fly during packet processing. These HopAuths are returned to the source AS AS_0 over a channel secured through authenticated encryption with associated data (AEAD):

$$AS_i \longrightarrow AS_0: \qquad \text{AEAD}_{K_{AS_i \to AS_0}}(\sigma_i). \tag{5}$$

Each HopAuth σ_i is then a secret, reservation-specific key, which the CServ at the source AS now shares with the on-path AS_i. The resulting set of keys, one for each on-path AS, is subsequently used to calculate and verify per-packet MACs in the data plane as described in §4.6.

4.6 Processing at Gateway and Router

Colibri Gateway. The gateway receives Colibri EER packets from end hosts, where all the Colibri header fields are empty, with the exception of the *ResId* and the *Payload*. The gateway maps the *ResId* of incoming EER packets to the corresponding Path, ResInfo, EERInfo, and the HopAuths, which it obtained earlier during an EER setup or renewal, and performs deterministic traffic monitoring as described in §4.8. It then generates a high-precision timestamp *Ts*, from which it computes the HVFs for all on-path ASes,

$$V_i^{(\text{E})} = \text{MAC}_{\sigma_i}\left(Ts \parallel PktSize\right)[\![0{:}\ell_{\text{hvf}}]\!], \tag{6}$$

where *PktSize* is the size of the packet (including Colibri header). The gateway thus confirms that it has performed the mandatory flow monitoring and authorized this packet. After filling in the missing EER packet contents, the gateway sends the packet to the border router responsible for the egress interface Eg_0.

Router. Upon reception of a Colibri packet, the border router of the *i*th on-path AS validates the packet format, header contents, and packet freshness, and checks whether the reservation has not expired yet. If the packet is a SegR packet, the border router validates the V_i in the packet header by recomputing it using Eq. (3). If the packet is an EER packet, the border router instead computes the authenticator σ_i using Eq. (4), from which it derives the V_i as defined in Eq. (6). In both cases, if this recomputed HVF matches the one specified in the packet, the packet is forwarded to the next entity. For a SegR—and therefore also SegReqs and EEReqs, which can be made over existing SegRs (§3.3)—this entity is the local CServ; for an EER it is the border router of the next AS according to the information in the Path field, except for the border router of the last AS, which forwards the EER packet to the host specified in *DstHost* instead.

4.7 Admission Algorithm

Segment Reservations. As a first step, any two neighboring ASes agree on the bandwidth available for Colibri traffic (the *traffic split* in §3.4) on their inter-domain link and negotiate the pricing model. These typically long-term contractual agreements—in the order of months—are always *bilateral* to facilitate negotiation and billing. Based on these, each AS can define a local traffic matrix that describes the allocation of Colibri traffic between interface pairs.

A fundamental challenge of Colibri is to distribute the bandwidth of an ingress–egress interface-pair in a fair manner. The admission algorithm must ensure that no AS or group of ASes can reserve excessive amounts of bandwidth for SegRs; this property has been dubbed *botnet-size independence* by Basescu et al. [9]. We achieve this property by employing an admission algorithm that guarantees *bounded tube fairness* as described and formally analyzed in a dissertation [62]. At a high level, Colibri's admission algorithm distributes the capacity among competing SegRs proportionally to their *adjusted* bandwidth demand, which is obtained by

(1) limiting the total demand coming from an ingress interface by that interface's capacity;
(2) limiting the total demand between an ingress and an egress interface by the egress interface's capacity; and
(3) limiting the total demand of a particular source AS at a particular egress interface by that interface's capacity.

To perform the SegR admission calculation for a SegReq, the CServ thus needs to look up all existing SegRs that use the same egress interface. As we show in §6, a careful implementation using memoization techniques still achieves the calculation in constant time in the number of existing SegRs, on the order of a millisecond.

End-to-End Reservations. The EER admission depends on the type of AS (§4.1).

- *Source AS*: The source AS has a direct business relationship with the end host. It is free to define which EERs can be provided to an individual end host in this contract. Upon receiving an EEReq, the source AS checks (i) if there is sufficient bandwidth in the first segment reservation, and (ii) whether the request is to be granted under its policy. If these checks are successful, the request is then forwarded along the path.
- *Transit AS*: Transit ASes only need to check if there is sufficient bandwidth in the SegR underlying the EEReq. This is necessary to defend against malicious source ASes, which may forward EEReqs for more bandwidth than available in the SegR.
- *Transfer AS*: A transfer AS needs to check if there is sufficient bandwidth in *both* SegRs it connects. Furthermore, the transfer AS between up- and core-SegR needs to distribute the core-SegR's bandwidth between all up-SegRs in case more EER bandwidth is requested than available in the core-SegR. This is done proportionally to the total of all requested EERs (capped at the up-SegR) that compete for the same core-SegR.
- *Destination AS*: The destination AS follows the same decision process as the source AS.

4.8 Monitoring and Policing

Monitoring and policing in Colibri is split into multiple components, which will be described in this section: deterministic monitoring and traffic shaping at the source AS, probabilistic monitoring at all other ASes, and policing of reservations. Control traffic on SegRs does not need to be processed at line rate, and can therefore be monitored and rate-limited at the CServ.

Deterministic Monitoring at Source AS. Traffic over EERs from end hosts inside an AS is monitored deterministically by the Colibri gateway (in parallel with the calculation of the HVFs). An efficient approach to limit the transmission rate of the flows from customers while still permitting short-term spikes in traffic is the token bucket algorithm [36], which only needs to keep a time stamp and a counter in memory for each flow. When a flow exceeds the maximum transmission rate for longer than the burst threshold, packets are simply dropped.

Probabilistic Monitoring at Other ASes. The probabilistic overuse flow detector (OFD) represents the centerpiece of the monitoring architecture in transit and transfer ASes. The main challenge with monitoring the traffic that originates from other ASes is that it is generally composed of very large number of flows. Therefore, in order to sustain line rate, the OFD must have a limited memory footprint and maximize the use of fast cache. Recent proposals have greatly improved the monitoring precision that is possible with such limited resources [11, 44, 49, 64, 67].

Colibri uses an OFD in each AS to monitor EERs. For each packet, the OFD receives as input (i) the *normalized packet size* (= *total packet size* / *reservation bandwidth*) and (ii) the source AS and the reservation ID, all of which can be read or calculated from the packet. It then tracks bandwidth usage of each EER separately using the pair (*SrcAS*, *ResId*) as a flow label; in particular, it combines packets of all versions of an EER in the same flow. Normalizing the packet size (i) enables monitoring reservations with different bandwidth guarantees using a single OFD and (ii) guarantees that a sender using multiple versions of the same EER can obtain at most the maximum bandwidth of all valid versions but not more. As the total packet size (including headers) is authenticated through the HVFs (6), framing attacks are prevented. Including the header size in the monitoring ensures that malicious source ASes cannot flood the system with packets with very small or no payload.

Due to the probabilistic nature of the OFD, it may report false positives—legitimate flows that appear to be overusing their bandwidth. For this reason, the suspicious EERs are subjected to deterministic monitoring, which inspects the reservation precisely—similar to the monitoring at the source AS—to determine overuse with certainty.

Policing. When a flow is confirmed to be exceeding its EER bandwidth, it can be concluded that the source AS did not perform its monitoring task properly. Typically, the AS that detects the abuse takes two measures: (i) block further traffic over the reservation and (ii) penalize the source AS. Measure (i) is crucial to avoid deteriorating service to legitimate reservations and is achieved by keeping a list of blocked source ASes. As this blocklist is very short—only a tiny share of the 70 000 ASes is expected to misbehave at any point in time—it can be implemented as a simple hash set.

After taking the immediate first step, the border router reports the offense to the local CServ. As misbehavior of the source AS has been established with certainty (due to the cryptographic checks), it is possible for the service to take drastic measures such as completely denying future reservations originating from that AS.

5 DDoS Resilience Analysis

Although a full security analysis is beyond the scope of this paper, we here briefly show how the interplay of cryptography, monitoring, and policing—as presented in the previous sections—protects Colibri reservations from attacks and thus enables worst-case minimum bandwidth guarantees.

5.1 Attacks on Reservation Traffic

Volumetric DDoS Attacks. DDoS attacks against Colibri traffic can be carried out with (i) best-effort traffic, (ii) bogus Colibri traffic, or even (iii) authentic Colibri traffic that is overusing a reservation. The first attack is prevented by traffic isolation (described in Appendix B). As the authentication procedure at border routers uses efficient symmetric cryptography (§4.5), bogus Colibri packets are quickly identified and dropped, countering the second attack. Finally, monitoring quickly identifies and blocks all overusing Colibri reservations (§4.8). In the worst case, an attacker that controls an AS can very briefly cause congestion, but would afterwards be prevented from creating reservations. It is not possible to create congestion with Colibri traffic that respects its reservations, as the admission procedure ensures that the sum of all reservations does not exceed the capacity (§4.7).

Framing DoS Attacks. An adversary could try to turn the monitoring subsystem against benign ASes by (i) spoofing the source AS, or (ii) capturing and replaying legitimate packets to overuse the reserved bandwidth, thus framing the legitimate source. Since overusing ASes are blocked by on-path ASes, this is another form

of DoS. Colibri avoids the first attack thanks to source authentication, and the second by using in-network duplicate suppression at benign ASes (§2.3). All copies of the same packet are thus discarded.

5.2 Attacks on the Admission Algorithm

Another way to degrade the service of target users is to try and obtain an unfairly large share of bandwidth from the reservation process. For SegRs, the admission calculation guarantees that the total amount of bandwidth that any AS or group of ASes can reserve is limited, thus ensuring that a benign AS can *always* obtain a finite minimum bandwidth [62].

The distribution of EER bandwidth is the responsibility of source and destination ASes, as described in §4.7. These are free to define policies on how to allocate the capacity of the SegRs they set up among their customers. As they have direct business relationships with end hosts and control their address space, they can easily define and enforce these rules.

Finally, all on-path ASes check that the total bandwidth of EERs on a particular SegR does not exceed that SegR's capacity. Therefore, a source AS cannot cause over-allocation by requesting an excessive amount of EER bandwidth.

5.3 Attacks on Reservation Setup

As presented so far, Colibri traffic is fully protected against DoS attacks when a reservation is established. The only remaining avenue for malicious actors is to try and prevent legitimate ASes or end hosts to set up Colibri reservations in the first place. In particular, an adversary can attempt to (i) exhaust the resources of the CServ with bogus requests, or (ii) block the reservation setup packet before it reaches the CServ by congesting the network with best-effort traffic. These attacks are known as denial-of-capability (DoC) attacks [7], and Colibri is carefully designed with multiple mechanisms to defend against them.

Efficient Authentication at the CServ. Every control-plane message is authenticated using symmetric cryptography that can be derived efficiently at the CServ. Thus, the CServ can very efficiently filter unauthentic packets and employ per-AS rate limiting.

Protected Control Traffic. As soon as a SegR or EER exists, renewal requests can be sent over this reservation and are thus isolated from flooding attacks with best-effort traffic. Therefore, ASes that want maximum protection against DoC—e.g, towards business-critical destination ASes—can preemptively setup a low-bandwidth, inexpensive SegR to these destinations; should the need arise, the reserved bandwidth can be flexibly increased through renewal requests that are then protected from DoC attacks. Finally, EEReqs are sent as control traffic over existing SegRs and thus also protected from best-effort traffic.

Prioritization of Initial Requests. Since renewals are protected by existing reservations, the only remaining DoC attack surface is the initial SegReq. Besides proactively setting up reservations ahead of time as described above, ASes can use the isolation mechanisms described in Appendix B to forward SegReqs with higher priority than best-effort traffic. As SegReqs are processed, authenticated, and filtered at each AS's CServ, they could only be used to flood the network of neighbor ASes, which is easily detectable.

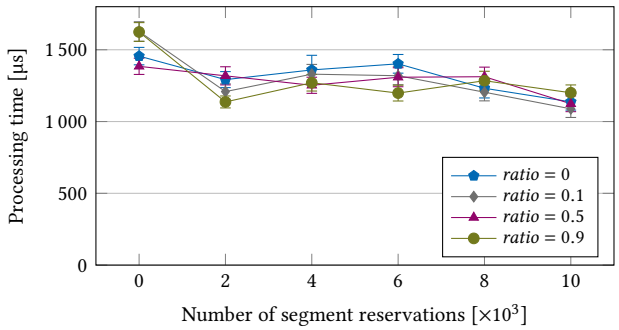


Figure 3: Processing time required for one SegR admission as a function of the number of existing SegRs over the same interface pair, and the ratio of SegRs with the same source as the one being processed (*ratio*).

6 Control-Plane Evaluation

6.1 Implementation and Evaluation Setup

All control-plane operations are centralized in the CServ, which is implemented in Go. The service is in charge of handling admission requests and renewing existing SegRs for the hosting AS. Each new request is served by a separate go routine, and the communication between services across different ASes is implemented via over gRPC calls [55] on top of QUIC. Reservations are stored in a transactional database. The CServ is part of the SCION codebase [45].

We measure the performance of control-plane operations as the time needed to process new reservation setup requests—for both SegRs and EERs—at the CServ. In the case of SegRs, we evaluate the computational overhead of the admission procedure by varying the number of distinct source ASes and the number of reservations per AS present at the moment of the admission; for EER admissions, we vary the number of end hosts and the number of SegRs.

To run the experiments, we load the pre-generated reservations in the service, start the service, and trigger a new setup request. We then observe the time elapsed between the request arriving and the response leaving the service (disregarding propagation delays). Each data point shows the average and standard error of 100 measurements. We run this evaluation on commodity hardware, using a single core on an Intel Xeon 2.8 GHz CPU.

6.2 Results

Figure 3 shows that the time to process SegR admissions is independent of the number of existing SegRs, even when crossing the same interfaces. This result required the careful application of memoization and parallelization, as the admission procedure for a new SegR at a transit AS needs to consider the other SegRs present on the pair of interfaces requested (§4.7). Should the need arise for the CServ to process more than 800 SegReqs per second (1/1250 µs), these computations can be scaled out to multiple cores, and even distributed across multiple CServ replicas (see Appendix D). The high scalability of the admission procedure suggests that Colibri's control plane will be able to scale to large, highly-interconnected networks like today's Internet, with consequently many SegRs.

In Fig. 4, we see that the overhead for the admission computation of EERs is independent of both the number of existing EERs over the same SegR and the number of SegRs. This is as expected:

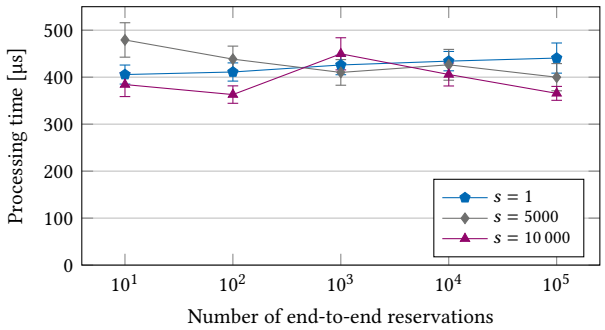


Figure 4: Processing time required for one EER admission in a transit AS, as a function of the number of existing EERs sharing the same SegR and active SegRs sharing the same source AS (s).

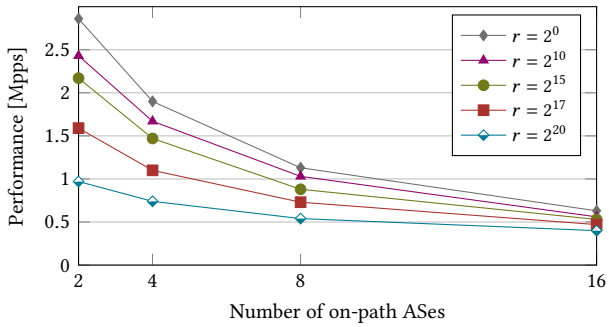


Figure 5: Forwarding performance of the gateway as a function of the number of on-path ASes and the number of existing reservations (r), using one CPU core.

EERs admission at transit ASes only requires checking whether there is sufficient available bandwidth in the underlying SegR (§4.7). With standard techniques, these checks can be performed in constant time, and—similar to SegR admission—EERs admission can parallelized over multiple cores, and distributed over multiple subservices. In our implementation, a single core can process more than 2000 requests per second.

7 Data-Plane Evaluation

In the data plane, we want to evaluate (i) the ability of gateways and border routers to forward EER traffic at high rates and (ii) the effectiveness of the EER bandwidth reservations in protecting traffic from extreme congestion.

7.1 Implementation and Evaluation Setup

We implement both the gateway and the border router as specified in §4.6, using Intel DPDK [18]. At the gateway we deploy DPDK's ready-to-use *rte_hash* hash table, where the key is the *ResId* and the value is a pointer to the reservation data. Note that we consider the duplicate-suppression system to be a separate component, which we do not include in the evaluation of the border router. For the calculation and verification of the HopAuths and HVFs, we use the AES-128 block cipher in CBC mode through native hardware-accelerated instructions, which are available on all modern CPUs.

Speedtest. To evaluate the gateway and the border router, we run them on a commodity machine with an Intel Xeon 2.1 GHz CPU. This machine is connected by three 40 Gbps bidirectional Ethernet links to our Colibri packet generator (Spirent SPT-N4U), which also serves as performance monitor.

For the border router, we evaluate how many packets per second it can handle depending on the number of cores dedicated to packet processing. For the gateway, we additionally evaluate the impact of the path length (longer paths require more authenticators to be calculated) and the number of existing reservations that the gateway needs to keep track of. While the border router is stateless, the gateway needs to store information about every reservation starting at its AS. As caching can greatly influence the data access speed, we evaluate the gateway in the worst-case scenario, packets arrive with random reservation IDs (out of the set of valid ones).

To test the limits of the gateway and border router, and avoid possible bottlenecks on the Ethernet links, we conduct the measurements using zero-payload Colibri packets. This is unproblematic as for both the gateway and the border router the packet-processing time is independent of payload size (see Appendix E).

Data-Plane Protection. In this second experiment, we measure Colibri's SLOs, and how effectively they are enforced at border routers. Authenticated Colibri traffic has to be protected against three different availability threats:

(1) *Best-effort cross-traffic:* as Colibri is intended to complement best-effort traffic on the underlying network, it has to be protected against congestion generated by co-existing best-effort flows.
(2) *Unauthentic Colibri packets:* an adversary can send Colibri packets without authorization, and replace the authentication tags with random strings hoping to overwhelm the authentication process on the router.
(3) *Authentic Colibri packets at unintended rates:* a faulty or malicious AS may not monitor Colibri flows originating in its network and thus send more traffic than it has reserved.

To evaluate the data-plane protection against those threats, we split our measurements into three phases, during which we send different mixtures of best-effort and authentic and unauthentic Colibri traffic over the three input ports, where the packets are all destined to the same output port.

7.2 Results

Speedtest. The forwarding performance of the gateway is illustrated in Fig. 5. As expected, the performance decreases with the number of on-path ASes and with the number of existing reservations. Even in particularly extreme situations, where there are over a million reservations and where every reservation is made on a path consisting of 16 ASes,[3] one CPU core of the gateway can forward 0.4 Mpps (million packets per second). The performance reduction observed when using a higher number of reservations is mostly due to an increase in cache misses. In such cases the Colibri gateway could be further sped up by adding more cache memory, or by using multiple gateways, each handling only a fraction of all reservations. By using random reservation IDs, we evaluated the

[3]The current average path length in the Internet is 4–5 AS hops [15, 61].

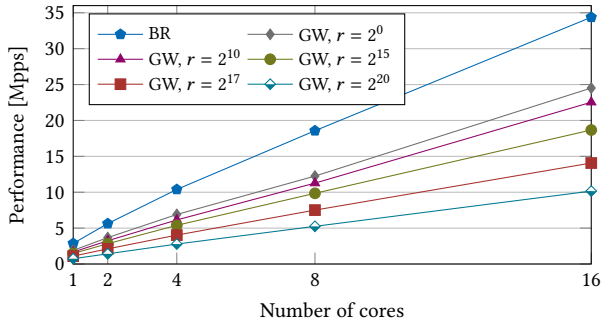


Figure 6: Forwarding performance of the gateway (GW) and border router (BR) as a function of the number of used cores. The gateway is evaluated for paths consisting of four ASes and various numbers of existing reservations (r).

Table 2: Measurement results in Gbps for different phases. Reservations 1 and 2 have bandwidth guarantees of 0.4 Gbps and 0.8 Gbps, respectively.

		Inputs			
	Traffic class	1	2	3	Output
phase 1	Reservation 1	0.400	—	—	0.400
	Reservation 2	—	0.800	—	0.800
	Best effort	—	39.200	40.000	38.669
phase 2	Reservation 1	0.400	—	—	0.400
	Reservation 2	—	0.800	—	0.800
	Best effort	—	39.200	20.000	38.643
	Colibri unauth.	—	—	20.000	—
phase 3	Reservation 1	40.000	—	—	0.400
	Reservation 2	—	0.800	—	0.800
	Best effort	—	39.200	20.000	38.608
	Colibri unauth.	—	—	20.000	—

gateway with worst-case inputs—for realistic traffic patterns the performance would therefore further increase.

Figure 6 shows the scalability of the border router and the gateway. We observe that for both components, the performance is almost perfectly linear in the number of cores dedicated to packet processing. For realistic parameters such as paths consisting of four ASes[3] and around 32 000 existing reservations, the gateway can forward 18.7 Mpps using 16 CPU cores. For packets with a payload of 1000 B, this corresponds to a throughput of 170 Gbps. The border router achieves even higher rates of 34.4 Mpps with 16 cores, which corresponds to 312 Gbps when using 1000 B packet payloads.

Data-Plane Protection. Through three phases, we show that the border router is able to effectively protect the Colibri reservations from best-effort cross traffic (phase 1), unauthentic Colibri attacks (phase 2), and overuse of the provided bandwidth guarantees (phase 3), see Table 2.

Phase 1 shows that congestion in the best-effort traffic does not impact the Colibri bandwidth due to the implemented traffic prioritization. Phase 2 demonstrates that unauthentic Colibri traffic is filtered effectively due to the cryptographic checks. In phase 3, we simulate a state where reservations 1 and 2 were flagged by the probabilistic flow monitor as suspicious (see §4.8) and are being deterministically monitored by means of the token-bucket algorithm. The overusing reservation 1 is limited to the guaranteed bandwidth of 0.8 Gbps without impacting the well-behaved reservation 2. Overall, the border router is able to process 120 Gbps of input (the maximum of our hardware) to fill the 40 Gbps output link under adversarial conditions without impacting the provided bandwidth guarantees of honest flows.

8 Related Work

Resource Allocation. Multiple queuing protocols [37, 47] aim to approximate the fair bandwidth allocation at each router. They rely on the flow identifiers being reported correctly by the end hosts and routers. The PCE architecture [57, 58] aims to provide inter-AS support and allows resource allocations in GMPLS networks. Unfortunately, all on-path ASes must cooperate and trust each other. RCS [14] is a mechanism to calculate fair traffic shares without congestion control. DiffServ [8] enables traffic prioritization based on multiple traffic classes but it does not provide any guarantees, and its application is restricted to intra-domain settings. All of these mechanisms lack security mechanisms and are thus not applicable to the global Internet. Route Bazaar [16] uses a decentralized public ledger to automate the negotiation of trustworthy SLOs on a path-aware Internet. This architecture, however, does not specify how such SLOs are created and enforced, as it is focused on their dissemination, billing, and the detection of misbehavior. Colibri's highly secure SLOs are then complementary to Route Bazaar's flexible SLO negotiation framework.

RSVP. RSVP [13, 72] is a protocol to signal bandwidth reservations. As it was designed without any security considerations, reservations may not be protected during DDoS attacks. RSVP forces routers, even with aggregation [19], to keep a large amount of state in the fast path.

Capability-Based Mechanisms. Many systems [3, 9, 22, 30, 31, 34, 38, 66, 69] try to isolate legitimate flows from illegitimate and malicious flows by issuing *network capabilities*, i.e., access tokens for on-path entities. Privileged channels—similar to Colibri's EERs—may only be used in case the packet carries the token. These schemes require defense mechanisms to protect against DoC attacks [7] and against collusion attacks [26, 53]. GLWP [65] also achieves inter-domain reservations, but only provides a small, inflexible amount of bandwidth to each AS.

Comparison with SIBRA. The Scalable Internet Bandwidth Reservation Architecture (SIBRA) [9] is an ancestor of Colibri, and they are both designed to operate on top of SCION. Therefore, they share many design principles. However, Colibri makes numerous substantial improvements.

Regarding security and performance, the major difference between the two systems is in the authentication mechanisms. SIBRA's hop-field authenticators are at the same time more complex (e.g., because of *chaining*) and provide weaker security guarantees: they do not allow an on-path AS to conclusively attribute a packet to the source, and cannot prevent the replay of authenticators. In Colibri, this problem is solved by the two-step calculation of HVFs (Eq. (6)). Further, monitoring is more efficient in Colibri, as it uses a single traffic monitor to track all bandwidth classes simultaneously and, thanks to source authentication, policing is much simpler than in SIBRA. Then, the lack of management scalability in SIBRA, where core reservations are static contracts, is addressed in Colibri by supporting the renegotiation of core reservations.

Finally, Colibri is implemented in a real system, and thus our evaluation evaluation accounts for all processing steps (including header updates, interactions with the NIC, etc.), in contrast to SIBRA, where only the parsing and MAC verification was evaluated.

Other DDoS-Defense Mechanisms. There exists a vast literature on DDoS-defense mechanisms; Zargar et al. published a taxonomy and extensive survey in 2013 [71]. While it is impossible to compare Colibri to all of them, a shortcoming shared by most mechanisms is that they are *reactive* and are thus not able to provide any availability guarantees. Furthermore, filtering techniques [48, and references therein] are vulnerable to source-address spoofing—unless cryptographic mechanisms (e.g., in the form of capabilities) are deployed in conjunction—and it has been shown that cloud-based traffic scrubbing can often be circumvented [59]. More recently, routing-based defenses have been proposed [50] but later shown to be ineffective in practice [56].

9 Discussion & Conclusion

The absence of global bandwidth-reservation architectures can be attributed to many factors—excessive overhead, intricate management, negligible benefits over best-effort—that amount to a general lack of incentives for deployment [27]. By addressing these concerns, Colibri contributes to the Internet ecosystem.

Low Overhead. Protecting performance-sensitive (e.g., low-latency) traffic is one of the main benefits of bandwidth reservation systems. However, if a system's overhead creates similar or worse effects as congestion, as in many past proposals, this benefit is negated. Colibri, on the contrary, is efficient in the data plane and can process multi-Gbps traffic in software (§7). Moreover, best-effort traffic coexists with Colibri reservations and can scavenge for unused bandwidth, thus avoiding network resource waste.

Simple Management. Coordinating reservation admission and billing in a distributed, global setting poses many challenges. A transit ISP does not have enough information to properly allocate bandwidth to flows that are not terminating at its direct customers. Further, billing such reservations is difficult, as costs are shared among many entities.

End-to-end information on flows is however not necessary when granting Colibri reservations, as the admission decision depends on neighbor-based policies. The admission procedure is fully automated and built into the Colibri control plane; as we show in §6, the system is fast and efficient in handing vast amounts of reservations. Additionally, thanks to the locality of policies, billing can be implemented with scalable neighbor-to-neighbor settlements, similarly to today's AS peering agreements.

A Profitable Service. Historically, overprovisioning has been the primary means with which ISPs maintain SLOs [23]. As the Internet remained successfully operational so far, the common belief is that this measure is largely sufficient. However, congestion is widely observable both at the edges and in the Internet core [17], and routing optimization is also known to generate unexpected congestion [25]. Further, recent attacks have been able to fill even large inter-domain links [20, 70]. Colibri reservations enforce fair sharing during peak traffic, mitigating the impact of these events, and enhancing the survivability and profitability of ASes. These characteristics add value to the service proposition of an ISP, and will therefore be drivers for deployment.

Secure Operation. Past systems did not consider adversarial actions, and could therefore be abused. Colibri employs a combination of light-weight cryptography, monitoring, and policing that make it the first system to achieve worst-case SLOs in a distributed, adversarial setting.

In summary, global network bandwidth guarantees in a public Internet have been, so far, only a distant mirage. Now, building on top of modern network architectures—providing per-packet source authentication, path-aware networking, and multi-path communication—we can finally construct a scalable global bandwidth-reservation system: Colibri. This is an important step towards providing strong Internet-wide SLOs and thus increasing the efficiency, resilience, and profitability of inter-domain communication.

Acknowledgements

We would like to thank Tom Anderson for suggesting the interpretation of bandwidth reservations as SLOs; Simon Leinen for the illuminating discussions on Internet-scale QoS; Tobias Klenze, Jonghoon Kwon, Joel Wanner, and Samuel Hitz for their feedback on the manuscript; and our shepherd Ignacio Castro and anonymous reviewers for their insightful comments. We gratefully acknowledge support from ETH Zurich, and from the Zurich Information Security and Privacy Center (ZISC).

References

[1] Anapaya Systems. 2021. SCION-Internet: The New Way To Connect. https://www.anapaya.net/scion-the-new-way-to-connect.

[2] Tom Anderson, Ken Birman, Robert Broberg, Matthew Caesar, Douglas Comer, Chase Cotton, Michael J. Freedman, Andreas Haeberlen, Zachary G. Ives, Arvind Krishnamurthy, William Lehr, Boon Thau Loo, David Mazières, Antonio Nicolosi, Jonathan M. Smith, Ion Stoica, Robbert van Renesse, Michael Walfish, Hakim Weatherspoon, and Christopher S. Yoo. 2013. The NEBULA Future Internet Architecture. In *The Future Internet*. Springer. https://doi.org/10.1007/978-3-642-38082-2_2

[3] Tom Anderson, Timothy Roscoe, and David Wetherall. 2004. Preventing Internet denial-of-service with capabilities. *ACM SIGCOMM Computer Communication Review (CCR)* 34, 1 (2004). https://doi.org/10.1145/972374.972382

[4] R. Annessi, J. Fabini, and T. Zseby. 2017. It's about Time: Securing Broadcast Time Synchronization with Data Origin Authentication. In *IEEE International Conference on Computer Communication and Networks (ICCCN)*. https://doi.org/10.1109/ICCCN.2017.8038418

[5] Robert Annessi, Joachim Fabini, and Tanja Zseby. 2017. SecureTime: Secure Multicast Time Synchronization. https://arxiv.org/abs/1705.10669.

[6] Luis Arceo-Miquel, Yuriy S Shmaliy, and Oscar Ibarra-Manzano. 2009. Optimal synchronization of local clocks by GPS 1PPS signals using predictive FIR filters. *IEEE Transactions on Instrumentation and Measurement* 58, 6 (2009).

[7] Katerina Argyraki and David Cheriton. 2005. Network capabilities: The good, the bad and the ugly. In *ACM Workshop on Hot Topics in Networks (HotNets)*.

[8] Fred Baker, David L. Black, Kathleen Nichols, and Steven L. Blake. 1998. *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*. RFC 2474.

[9] Cristina Basescu, Raphael M. Reischuk, Pawel Szalachowski, Adrian Perrig, Yao Zhang, Hsu-Chun Hsiao, Ayumu Kubota, and Jumpei Urakawa. 2016. SIBRA: Scalable Internet Bandwidth Reservation Architecture. In *Symposium on Network and Distributed Systems Security (NDSS)*. https://doi.org/10.14722/ndss.2016.23132

[10] Tony Bates, Philip Smith, and Geoff Huston. 2021. CIDR Report. https://www.cidr-report.org/as2.0/.

[11] R. Ben Basat, X. Chen, G. Einziger, R. Friedman, and Y. Kassner. 2019. Randomized Admission Policy for Efficient Top-k, Frequency, and Volume Estimation. *IEEE/ACM Transactions on Networking (ToN)* 27, 4 (2019). https://doi.org/10.1109/TNET.2019.2918929

[12] Bobby Bhattacharjee, Ken Calvert, Jim Griffioen, Neil Spring, and James P. G. Sterbenz. 2006. Postmodern Internetwork Architecture. *NSF Nets FIND Initiative* (2006).

[13] Robert T. Braden, Lixia Zhang, Steven Berson, Shai Herzog, and Sugih Jamin. 1997. *Resource ReSerVation Protocol (RSVP) – Version 1 Functional Specification*. RFC 2205.

[14] Lloyd Brown, Ganesh Ananthanarayanan, Ethan Katz-Bassett, Arvind Krishnamurthy, Sylvia Ratnasamy, Michael Schapira, and Scott Shenker. 2020. On the Future of Congestion Control for the Public Internet. In *ACM Workshop on Hot Topics in Networks (HotNets)*. https://doi.org/10.1145/3422604.3425939

[15] Timm Böttger, Gianni Antichi, Eder L. Fernandes, Roberto di Lallo, Marc Bruyere, Steve Uhlig, Gareth Tyson, and Ignacio Castro. 2019. Shaping the Internet: 10 Years of IXP Growth. https://arxiv.org/abs/1810.10963v3.

[16] Ignacio Castro, Aurojit Panda, Barath Raghavan, Scott Shenker, and Sergey Gorinsky. 2015. Route Bazaar: Automatic Interdomain Contract Negotiation. In *USENIX Workshop on Hot Topics in Operating Systems*.

[17] Amogh Dhamdhere, Kc Claffy, David D. Clark, Alexander Gamero-Garrido, Matthew Luckie, Ricky K. P. Mok, Gautam Akiwate, Kabir Gogia, Vaibhav Bajpai, and Alex C. Snoeren. 2018. Inferring persistent interdomain congestion. In *ACM SIGCOMM Conference*. https://doi.org/10.1145/3230543.3230549

[18] DPDK Project. 2021. Data Plane Development Kit. https://dpdk.org.

[19] François Le Faucheur, Fred Baker, Dr. Bruce S. Davie, and Carol Iturralde. 2001. *Aggregation of RSVP for IPv4 and IPv6 Reservations*. RFC 3175.

[20] Nick Galov. 2021. 39 Jaw-Dropping DDoS Statistics to Keep in Mind for 2021. https://hostingtribunal.com/blog/ddos-statistics/.

[21] P. Brighten Godfrey, Igor Ganichev, Scott Shenker, and Ion Stoica. 2009. Pathlet Routing. In *ACM SIGCOMM Conference*.

[22] Hsu-Chun Hsiao, Tiffany Hyun-Jin Kim, Sangjae Yoo, Xin Zhang, Soo Bum Lee, Virgil Gligor, and Adrian Perrig. 2013. STRIDE: Sanctuary Trail – Refuge from Internet DDoS Entrapment.. In *ACM Symposium on Information, Computer, and Communications Security (ASIACCS)*. https://doi.org/10.1145/2484313.2484367

[23] Geoff Huston. 2012. The QoS Emperor's Wardrobe. https://labs.ripe.net/Members/gih/the-qos-emperors-wardrobe.

[24] IEEE. 2019. IEEE 1588-2019 – IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems.

[25] Min Suk Kang and Virgil D. Gligor. 2014. Routing Bottlenecks in the Internet. In *ACM Conference on Computer and Communications Security (CCS)*. https://doi.org/10.1145/2660267.2660299

[26] Min Suk Kang, Soo Bum Lee, and V. D. Gligor. 2013. The Crossfire Attack. In *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/sp.2013.19

[27] Kalevi Kilkki and Benjamin Finley. 2019. In Search of Lost QoS. https://arxiv.org/abs/1901.06867.

[28] Cyrill Krähnebühl, Seyedali Tabeiaghdaei, Christelle Gloor, Jonghoon Kwon, Adrian Perrig, David Hausheer, and Dominik Roos. 2021. Deployment and Scalability of an Inter-Domain Multi-Path Routing Infrastructure. In *Conference on Emerging Networking Experiments and Technologies (CoNEXT)*.

[29] Jonghoon Kwon, Juan A. García-Pardo, Markus Legner, François Wirz, Matthias Frei, David Hausheer, and Adrian Perrig. 2020. SCIONLab: A Next-Generation Internet Testbed. In *IEEE Conference on Network Protocols (ICNP)*.

[30] Soo Bum Lee and Virgil D. Gligor. 2010. FLoc : Dependable Link Access for Legitimate Traffic in Flooding Attacks. In *IEEE International Conference on Distributed Computing Systems*. https://doi.org/10.1109/icdcs.2010.78

[31] Soo Bum Lee, Min Suk Kang, and Virgil D. Gligor. 2013. CoDef: Collaborative defense against large-scale link-flooding attacks. In *Conference on Emerging Networking Experiments and Technologies (CoNEXT)*. https://doi.org/10.1145/2535372.2535398

[32] Taeho Lee, Christos Pappas, Adrian Perrig, Virgil Gligor, and Yih-Chun Hu. 2017. The Case for In-Network Replay Suppression. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. https://doi.org/10.1145/3052973.3052988

[33] Markus Legner, Tobias Klenze, Marc Wyss, Christoph Sprenger, and Adrian Perrig. 2020. EPIC: Every Packet Is Checked in the Data Plane of a Path-Aware Internet. In *USENIX Security Symposium (USENIX Security)*.

[34] Zhuotao Liu, Hao Jin, Yih-Chun Hu, and Michael Bailey. 2016. MiddlePolice: Toward enforcing destination-defined policies in the middle of the Internet. In *ACM Conference on Computer and Communications Security (CCS)*.

[35] Jim Martin, Jack Burbank, William Kasch, and Professor David L. Mills. 2010. *Network Time Protocol Version 4: Protocol and Algorithms Specification*. RFC 5905.

[36] Deepankar Medhi and Karthikeyan Ramasamy. 2007. *Network Routing: Algorithms, Protocols, and Architectures*. Morgan Kaufmann Publishers.

[37] Rong Pan, B. Prabhakar, and K. Psounis. 2000. CHOKe – a stateless active queue management scheme for approximating fair bandwidth allocation. In *IEEE Conference on Computer Communications (INFOCOM)*. https://doi.org/10.1109/infcom.2000.832269

[38] Bryan Parno, Dan Wendlandt, Elaine Shi, Adrian Perrig, Bruce Maggs, and Yih-Chun Hu. 2007. Portcullis: Protecting Connection Setup from Denial-of-Capability Attacks. In *ACM SIGCOMM Conference*. https://doi.org/10.1145/1282380.1282413

[39] Adrian Perrig, Pawel Szalachowski, Raphael M. Reischuk, and Laurent Chuat. 2017. *SCION: A Secure Internet Architecture*. Springer. https://doi.org/10.1007/978-3-319-67080-5

[40] Barath Raghavan and Alex C. Snoeren. 2004. A system for authenticated policy-compliant routing. *ACM SIGCOMM Computer Communication Review (CCR)* 34, 4 (2004). https://doi.org/10.1145/1030194.1015487

[41] Barath Raghavan, Patric Verkaik, and Alex C. Snoeren. 2009. Secure and Policy-Compliant Source Routing. *IEEE/ACM Transactions on Networking (ToN)* 17, 3 (2009). https://doi.org/10.1109/tnet.2008.2007949

[42] Henrique Rodrigues, Jose Renato Santos, Yoshio Turner, Paolo Soares, and Dorgival O Guedes. 2011. Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks. In *Conference of I/O virtualization (WIOV)*.

[43] Benjamin Rothenberger, Dominik Roos, Markus Legner, and Adrian Perrig. 2020. PISKES: Pragmatic Internet-Scale Key-Establishment System. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. https://doi.org/10.1145/3320269.3384743

[44] Simon Scherrer, Che-Yu Wu, Yu-Hsi Chiang, Benjamin Rothenberger, Daniele Asoni, Arish Sateesan, Jo Vliegen, Nele Mentens, Hsu-Chun Hsiao, and Adrian Perrig. 2021. Low-Rate Overuse Flow Tracer (LOFT): An Efficient and Scalable Algorithm for Detecting Overuse Flows. In *Symposium on Reliable Distributed Systems (SRDS)*.

[45] SCION Project. 2021. SCION Open-Source Implementation. https://github.com/scionproto/scion.

[46] Chuck Semeria. 2001. *Supporting differentiated service classes: queue scheduling disciplines*. Technical Report. Juniper Networks.

[47] M. Shreedhar and G. Varghese. 1996. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on Networking (ToN)* 4, 3 (1996). https://doi.org/10.1109/90.502236

[48] Devkishen Sisodia, Jun Li, and Lei Jiao. 2020. In-Network Filtering of Distributed Denial-of-Service Traffic with Near-Optimal Rule Selection. In *ACM Asia Conference on Computer and Communications Security (ASIACCS)*. https://doi.org/10.1145/3320269.3384755

[49] Vibhaalakshmi Sivaraman, Srinivas Narayana, Ori Rottenstreich, S. Muthukrishnan, and Jennifer Rexford. 2017. Heavy-Hitter Detection Entirely in the Data Plane. In *Symposium on SDN Research (SOSR)*. https://doi.org/10.1145/3050220.3063772

[50] Jared M Smith and Max Schuchard. 2018. Routing Around Congestion: Defeating DDoS Attacks and Adverse Network Conditions via Reactive BGP Routing. In *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/sp.2018.00032

[51] João Luís Sobrinho and Miguel Alves Ferreira. 2020. Routing on Multiple Optimality Criteria. In *ACM SIGCOMM Conference*. https://doi.org/10.1145/3387514.3405864

[52] C. Song, H. Moon, M. Alam, I. Yun, B. Lee, T. Kim, W. Lee, and Y. Paek. 2016. HDFI: Hardware-Assisted Data-Flow Isolation. In *IEEE Symposium on Security and Privacy (SP)*. https://doi.org/10.1109/SP.2016.9

[53] Ahren Studer and Adrian Perrig. 2009. The Coremelt Attack. In *Computer Security – ESORICS*. Springer. https://doi.org/10.1007/978-3-642-04444-1_3

[54] Swisscom AG. 2021. Enhancing WAN connectivity and services for Swiss organisations with the next-generation internet. https://www.swisscom.ch/scion.

[55] The gRPC Authors and The Linux Foundation. 2021. gRPC: A high performance, open source universal RPC framework. https://grpc.io/.

[56] Muoi Tran, Min Suk Kang, Hsu-Chun Hsiao, Wei-Hsuan Chiang, Shu-Po Tung, and Yu-Su Wang. 2019. On the Feasibility of Rerouting-based DDoS Defenses. In *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/SP.2019.00055

[57] JP Vasseur, Adrian Farrel, and Gerald Ash. 2006. *A Path Computation Element (PCE)-Based Architecture*. RFC 4655.

[58] JP Vasseur and Jean-Louis Le Roux. 2009. *Path Computation Element (PCE) Communication Protocol (PCEP)*. RFC 5440.

[59] Thomas Vissers, Tom Van Goethem, Wouter Joosen, and Nick Nikiforakis. 2015. Maneuvering around clouds: Bypassing cloud-based security providers. In *ACM Conference on Computer and Communications Security (CCS)*.

[60] VMware. 2020. vSphere Resource Management. https://docs.vmware.com/en/VMware-vSphere/7.0/vsphere-esxi-vcenter-server-701-resource-management-guide.pdf.

[61] Cun Wang, Zhengmin Li, Xiaohong Huang, and Pei Zhang. 2016. Inferring the average AS path length of the Internet. In *IEEE International Conference on Network Infrastructure and Digital Content (IC-NIDC)*. https://doi.org/10.1109/icnidc.2016.7974603

[62] Thilo Weghorn. 2019. *Qualitative and Quantitative Guarantees for Access Control*. Ph.D. Dissertation. ETH Zürich. https://doi.org/10.3929/ethz-b-000397549

[63] Emmett Witchel, Josh Cates, and Krste Asanović. 2002. Mondrian Memory Protection. *SIGPLAN Notices* 37, 10 (2002). https://doi.org/10.1145/605432.605429

[64] Hao Wu, Hsu-Chun Hsiao, and Yih-Chun Hu. 2014. Efficient Large Flow Detection over Arbitrary Windows. In *ACM Internet Measurement Conference (IMC)*. https://doi.org/10.1145/2663716.2663724

[65] Marc Wyss, Giacomo Giuliari, Markus Legner, and Adrian Perrig. 2021. Secure and Scalable QoS for Critical Applications. In *IEEE/ACM International Symposium on Quality of Service (IWQoS)*.

[66] A. Yaar, A. Perrig, and D. Song. 2004. SIFF: a stateless Internet flow filter to mitigate DDoS flooding attacks. In *IEEE Symposium on Security and Privacy (S&P)*. https://doi.org/10.1109/secpri.2004.1301320

[67] T. Yang, H. Zhang, J. Li, J. Gong, S. Uhlig, S. Chen, and X. Li. 2019. HeavyKeeper: An Accurate Algorithm for Finding Top-k Elephant Flows. *IEEE/ACM Transactions on Networking (ToN)* 27, 5 (2019). https://doi.org/10.1109/TNET.2019.2933868
[68] Xiaowei Yang, David Clark, and Arthur W. Berger. 2007. NIRA: A New Inter-Domain Routing Architecture. *IEEE/ACM Transactions on Networking (ToN)* (2007).
[69] Xiaowei Yang, David Wetherall, and Thomas Anderson. 2005. A DoS-limiting network architecture. In *ACM SIGCOMM Conference*. https://doi.org/10.1145/1080091.1080120
[70] Omer Yoachimik and Vivek Ganti. 2020. Network-layer DDoS attack trends for Q3 2020. https://blog.cloudflare.com/network-layer-ddos-attack-trends-for-q3-2020/.
[71] S. T. Zargar, J. Joshi, and D. Tipper. 2013. A Survey of Defense Mechanisms Against Distributed Denial of Service (DDoS) Flooding Attacks. *IEEE Communications Surveys & Tutorials* 15, 4 (2013). https://doi.org/10.1109/SURV.2013.031413.00127
[72] Lixia Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. 2002. RSVP: a new resource reservation protocol. *IEEE Communications Magazine* 40, 5 (2002). https://doi.org/10.1109/mcom.2002.1006981
[73] Xin Zhang, Hsu-Chun Hsiao, Geoffrey Hasker, Haowen Chan, Adrian Perrig, and David Andersen. 2011. SCION: Scalability, Control, and Isolation On Next-Generation Networks. In *IEEE Symposium on Security and Privacy (S&P)*.

A List of Acronyms

AEAD	authenticated encryption with associated data
AS	autonomous system
BGP	the Border Gateway Protocol
CBWFQ	class-based weighted fair queuing
CDN	content-distribution network
CServ	Colibri service
DoC	denial-of-capability
DDoS	distributed-denial-of-service
DoS	denial-of-service
DRKey	dynamically-recreatable-key
EER	end-to-end reservation
EEReq	end-to-end–reservation setup request
GMPLS	generalized multi-protocol label switching
HopAuth	hop authenticator
HVF	hop validation field
ISD	isolation domain
ISP	Internet service provider
MAC	message-authentication code
OFD	overuse flow detector
PCFS	packet-carried forwarding state
PRF	pseudo-random function
QoS	quality of service
RSVP	the Resource Reservation Protocol
SLA	service-level agreement
SLO	service-level objective
SegR	segment reservation
SegReq	segment-reservation setup request
ToS	type of service
WAN	wide-area network

B Traffic Isolation

Colibri is intended to coexist with best-effort traffic of the underlying network architecture, sharing the same physical infrastructure. This raises the challenge of protecting Colibri traffic from congestion and attacks in the best-effort traffic class, while, at the same time, ensuring that unused Colibri bandwidth is available for best-effort traffic.

We achieve this separation by defining three different traffic classes—best-effort, Colibri control, and Colibri data traffic—and using queuing techniques such as *priority queuing*[4] or *class-based weighted fair queuing* [46], which are available on most modern networking equipment. It is crucial that priority is given to Colibri traffic not only at border routers, but also at switches and routers in each AS's internal network. This requires encoding the traffic class in the header of the intra-domain networking protocol in use. For example, in an IP network, the traffic class can be encoded using DiffServ and the DSCP field [8]. To defend against malicious hosts in an AS's network, all traffic should pass through a gateway that sets this field to the correct value.

C Dissemination of Segment Reservations

To reduce the latency of setting up a bandwidth reservation, end hosts need to efficiently obtain the SegRs that are necessary to construct their desired EERs. For this purpose, Colibri uses a hierarchical caching approach at CServs: hosts can then fetch SegRs from their local AS with minimal latency.

Once a SegR is established, the initiator can choose to share it publicly by registering it at its CServ along with a whitelist of ASes that are allowed to use the SegR to create EERs. An end host can then query its local CServ for SegRs to the intended destination, which looks up SegRs in its database and contacts remote CServs if necessary to fetch additional SegRs that together form a complete path. These additional SegRs are then also cached at the local CServ.

With this hierarchical caching approach, queries can be answered with low latency. However, it is not directly observable when the version in a remote SegR is switched, which may lead to end hosts initiating EEReqs with an outdated underlying reservation. This is not an issue, as the remote CServ can indicate expiry of the SegR during setup of the EER, allowing the end host to retry with the new version of the SegR. As this reply also passes through the source AS's CServ, its cache can be invalidated.

D Distributed Colibri Service

For an AS that receives a very large amount of reservations (e.g., an AS located in the Internet core), a CServ that is deployed on a single machine will be subject to high load and can potentially become the bottleneck. If this is the case, the AS can leverage the hierarchical structure of reservations to distribute the load efficiently.

While admission for SegRs requires a complete view of all SegRs passing through the AS, EEReqs can be handled using knowledge of only a specific subset of the reservations in the AS. Concretely, the decision of an AS to admit an EER depends only on the state of the adjacent SegRs that are used in the requested reservation. While this is a single SegR in most cases, a transfer AS (located at the intersection of path segments) may need to consider both the underlying incoming and the outgoing SegRs in the admission process. Even in this more complex case, the decision can be split into two separate problems: (i) admission based on the incoming SegR, and (ii) admission based on the outgoing SegR.

[4] As the CServ ensures that the total bandwidth of all active reservations does not exceed the available bandwidth reserved for Colibri, strict priority queuing can be used without risking starvation of best-effort traffic.

These observations enable a decomposition of the CServ into three types of sub-services: *the coordinator sub-service*, which handles all SegReq; (ii) *the ingress sub-service*, which handles EEReqs that use a given ingress interface; and (iii) *the egress sub-service*, which handles EEReqs that use a given egress interface and is only necessary at transfer ASes. In case a load balancer is used to distribute requests across sub-services on the same interface, it must assign the requests such that all EEReqs based on the same underlying SegR are processed by the same sub-service. This enables the admission decisions to be parallelized easily. Depending on the expected load, the AS can deploy the sub-service instances on dedicated machines or routers.

E Additional Evaluation Results

We also explored the performance of the gateway and the border router with respect to differently sized Colibri packets. Because the size of the packet headers together with a payload of 1500 B exceeds the Ethernet maximum transmission unit, we enabled jumbo frame support in our test network.

For this experiment, we initialize the gateway with 2^{15} pre-existing reservations; the border router does not maintain state on reservations. We observed that for both components, forwarding is not influenced by the payload size. The border router and the gateway can forward 3 Mpps and 1.5 Mpps respectively, independently of the payload size.