Network Working Group
Internet-Draft
Intended status: Informational
Expires: 8 January 2026

C. de Kater
N. Rustignoli
SCION Association
S. Hitz
Anapaya Systems
7 July 2025

SCION Control Plane
draft-dekater-scion-controlplane-09

Abstract

This document describes the Control Plane of the path-aware, inter-domain network architecture SCION (Scalability, Control, and Isolation On Next-generation networks). A fundamental characteristic of SCION is that it gives path control to SCION-capable endpoints that can choose between multiple path options, thereby enabling the optimization of network paths. The Control Plane is responsible for discovering these paths and making them available to the endpoints.

The SCION Control Plane creates and securely disseminates path segments between SCION Autonomous Systems (AS) which can then be combined into forwarding paths to transmit packets in the data plane. This document describes mechanisms of path exploration through beaconing and path registration. In addition, it describes how Endpoints construct end-to-end paths from a set of possible path segments through a path lookup process.

This document contains new approaches to secure path aware networking. It is not an Internet Standard, has not received any formal review of the IETF, nor was the work developed through the rough consensus process. The approaches offered in this work are offered to the community for its consideration in the further evolution of the Internet.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://scionassociation.github.io/scion-cp_I-D/draft-dekater-scion-controlplane.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-dekater-scion-controlplane/.

Source for this draft and an issue tracker can be found at https://github.com/scionassociation/scion-cp_I-D.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

SCION is a path-aware internetworking routing architecture as described in [RFC9217]. It allows endpoints and applications to select paths across the network to use for traffic, based on trustworthy path properties. SCION is an inter-domain network architecture and is therefore not concerned with intra-domain forwarding.

SCION has been developed with the following goals:

Availability - to provide highly available communication that can send traffic over paths with optimal or required characteristics, quickly handle inter-domain link or router failures (both on the last hop or anywhere along the path) and provide continuity in the presence of adversaries.

Security - to introduce a new approach to inter-domain path security that leverages path awareness in combination with a unique trust model. The goal is to provide higher levels of trustworthiness in routing information to prevent traffic hijacking, and enable users to decide where their data travels based on routing information that can be unambiguously attributed to an AS, ensuring that packets are only forwarded along authorized path segments. A particular use case is to enable geofencing. Security properties are further discussed in Section 7.1.

Scalability - to improve the scalability of the inter-domain control plane and data plane, avoiding existing limitations related to convergence and forwarding table size. The advertising of path segments is separated into a beaconing process within each Isolation Domain (ISD) and between ISDs which incurs minimal overhead and resource requirements on routers.

SCION relies on three main components:

PKI - To achieve scalability and trust, SCION organizes existing ASes into logical groups of independent routing planes called _Isolation Domains (ISDs)_. All ASes in an ISD agree on a set of trust roots called the _Trust Root Configuration (TRC)_ which is a collection of signed root certificates in X.509 v3 format [RFC5280]. The ISD is governed by a set of _core ASes_ which typically manage the trust roots and provide connectivity to other ISDs. This is the basis of the public key infrastructure used for the authentication of messages used by the SCION Control Plane.

Control Plane - performs inter-domain routing by discovering and securely disseminating path information between ASes. The core ASes use Path-segment Construction Beacons (PCBs) to explore intra-ISD paths, or to explore paths across different ISDs.

Data Plane - carries out secure packet forwarding between SCION-enabled ASes over paths selected by endpoints. A SCION border router reuses existing intra-domain infrastructure to communicate to other SCION routers or SCION endpoints within its AS.

This document describes the SCION Control Plane component. It should be read in conjunction with the other components [I-D.dekater-scion-pki] and [I-D.dekater-scion-dataplane].

The SCION architecture was initially developed outside of the IETF by ETH Zurich with significant contributions from Anapaya Systems. It is deployed in the Swiss finance sector to provide resilient connectivity between financial institutions. The aim of this document is to document the existing protocol specification as

deployed, to encourage interoperability among implementations, and to introduce new concepts that can potentially be further improved to address particular problems with the current Internet architecture. This document is not an Internet Standards Track specification; it does not have IETF consensus and it is published for informational purposes.

Note (to be removed before publication): this document, together with the other components [I-D.dekater-scion-pki] and [I-D.dekater-scion-dataplane], deprecates [I-D.dekater-panrg-scion-overview].

1.1. Terminology

Autonomous System (AS): An Autonomous System is a network under a common administrative control. For example, the network of an Internet service provider, company, or university can constitute an AS.

Beaconing: The Control Plane process where an AS discovers paths to other ASes.

Control Plane: The SCION Control Plane is responsible for the propagation and discovery of network paths, i.e., for the exchange of routing information between SCION nodes. The Control Plane thus determines where traffic can be sent and deals with questions such as how paths are discovered, which paths exist, how they are disseminated to endpoints, etc. Within a SCION AS, such functionalities are carried out by the Control Service whereas packet forwarding is a task carried out by the data plane.

Control Service: The Control Service is the main control plane infrastructure component within a SCION AS. It is responsible for the path exploration and registration processes that take place within the Control Plane.

Core AS: Each Isolation Domain (ISD) is administered by a set of distinguished autonomous systems (ASes) called core ASes, which are responsible for initiating the path-discovery and -construction process (in SCION called "beaconing"). Each ISD MUST have at least one Core AS.

Endpoint: An endpoint is the start or the end of a SCION path, as defined in [RFC9473].

Forwarding Path: A forwarding path is a complete end-to-end path between two SCION endpoints which is used to transmit packets in the data plane. It can be created with a combination of up to three path segments (an up segment, a core segment, and a down segment).

Hop Field (HF): As they traverse the network, Path Segment Construction Beacons (PCBs) accumulate cryptographically protected AS-level path information in the form of Hop Fields. In the data plane, Hop Fields are used for packet forwarding: they contain the incoming and outgoing Interface IDs of the ASes on the forwarding path.

Info Field (INF): Each Path Segment Construction Beacon (PCB) contains a single Info field, which provides basic information about the PCB. Together with Hop Fields (HFs), these are used to create forwarding paths.

Isolation Domain (ISD): In SCION, Autonomous Systems (ASes) are organized into logical groups called Isolation Domains or ISDs. Each ISD consists of ASes that span an area with a uniform trust environment (e.g. a common jurisdiction). A possible model is for ISDs to be formed along national boundaries or federations of nations.

Leaf AS: An AS at the "edge" of an ISD, with no other downstream ASes.

Message Authentication Code (MAC). In the rest of this document, "MAC" always refers to "Message Authentication Code" and never to "Medium Access Control". When "Medium Access Control address" is implied, the phrase "Link Layer Address" is used.

Path Segment: Path segments are derived from Path Segment Construction Beacons (PCBs). A path segment can be (1) an up segment (i.e. a path between a non-core AS and a core AS in the same ISD), (2) a down segment (i.e. the same as an up segment, but in the opposite direction), or (3) a core segment (i.e. a path between core ASes). Up to three path segments can be used to create a forwarding path.

Path Segment Construction Beacon (PCB): Core AS control planes generate PCBs to explore paths within their isolation domain (ISD) and among different ISDs. ASes further propagate selected PCBs to their neighboring ASes. These PCBs traverse each AS accumulating information, including Hop Fields (HFs) which can subsequently be used for traffic forwarding.

SCION Control Message Protocol (SCMP): A signaling protocol analogous to the Internet Control Message Protocol (ICMP). This is described in Section 6.

Trust Root Configuration (TRC): A Trust Root Configuration or TRC is a signed collection of certificates pertaining to an isolation domain (ISD). TRCs also contain ISD-specific policies.

1.2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

1.3. Paths and Links

SCION routers and endpoints connect to each other via links. A link refers to a physical or logical connection between two SCION nodes (e.g., router or endpoint). A SCION path between two endpoints traverses one or more links.

In SCION, Autonomous Systems (ASes) are organized into logical groups called Isolation Domains or ISDs. Each ISD consists of ASes that are part of a uniform trust environment (i.e. a common jurisdiction) and is administered by a set of distinguished ASes called core ASes.

SCION distinguishes three types of links between ASes: (1) core links, (2) parent-child links, and (3) peering links.

* _Core_ links connect two core ASes, which are either within the same or in different ISDs. Core links can exist for various reasons, including provider-customer (where the customer pays the provider for traffic) and peering relationships.

* _Parent-child_ links create a hierarchy between the parent and the child AS within the same ISD. ASes with a parent-child link typically have a provider-customer relationship.

* _Peering_ links exist between ASes with a (settlement-free or paid) peering relationship. They can be established between any two ASes (core or non-core) and can cross ISD boundaries.

SCION paths are comprised of at most three path segments: an up segment, traversing links from child to parent, then a core segment consisting of core links, followed by a down segment traversing links from parent to child. Each path segment is established over one or more links.

SCION paths are always "valley free" whereby a child AS does not carry transit traffic from a parent AS to another parent AS. These paths can contain at most one peering link which can be used as shortcut between two path segments containing two peer ASes.

Figure 1 shows the three types of links for one small ISD with two core ASes A and C, and four non-core ASes D,E,F, and G.

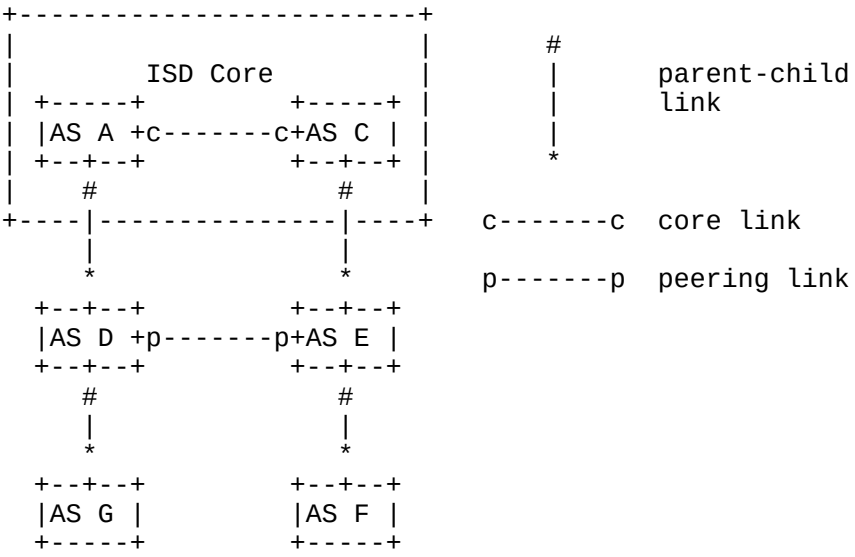


Figure 1: The three types of SCION links in one ISD. Each node in the figure is a SCION AS.

Each link connecting SCION routers is bi-directional and is identified by its corresponding egress and ingress Interface IDs. An Interface ID is a 16-bit identifier as described in [I-D.dekater-scion-dataplane] in section Terminology. It is required to be unique within each AS and can therefore be chosen without any need for coordination between ASes.

1.4. Routing

SCION provides path-aware inter-domain routing between ASes across the Internet. The SCION Control Plane is responsible for discovering these inter-domain paths and making them available to the endpoints within the ASes.

SCION inter-domain routing operates on two levels: within a ISD which is called _intra_-ISD routing, and between ISD which is called _inter_-ISD routing. Both levels use _Path Segment Construction Beacons (PCBs)_ to explore network paths. A PCB is initiated by a core AS and then disseminated either within an ISD to explore intra-ISD paths, or among core ASes to explore core paths across different ISDs.

The PCBs accumulate cryptographically protected path and forwarding information at an AS level and store this information in the form of _Hop Fields_. Endpoints use information from these Hop Fields to create end-to-end forwarding paths for data packets that carry this information in their headers. This also supports multi-path communication among endpoints.

The creation of an end-to-end forwarding path consists of the following processes:

1. _Path exploration (or beaconing)_: This is the process where an AS discovers paths to other ASes. See also Section 2.

2. _Path registration_: This is the process where an AS selects a few PCBs, according to defined policies, turns the selected PCBs into path segments, and adds these path segments to the relevant path infrastructure, thus making them available to other ASes. See also Section 4.

3. _Path resolution_: This is the process of actually creating an end-to-end forwarding path from the source endpoint to the destination. For this, an endpoint performs (a) a path lookup step to obtain path segments, and (b) a path combination step to combine the forwarding path from the segments. This last step takes place in the data plane. See also Section 5.

All processes operate concurrently.

The *Control Service* is responsible for the path exploration and registration processes in the Control Plane. It is the main control plane infrastructure component within each SCION AS and has the following tasks:

* Generating, receiving, and propagating PCBs. Periodically, the Control Service of a core AS generates a set of PCBs, which are forwarded to the child ASes or neighboring core ASes. In the latter case, the PCBs are sent over policy compliant paths to discover multiple paths between any pair of core ASes.

* Selecting and registering the set of path segments via which the AS wants to be reached.

* Managing certificates and keys to secure inter-AS communication. Each PCB contains signatures of all on-path ASes and each time the Control Service of an AS receives a PCB, it validates the PCB's authenticity. When the Control Service lacks an intermediate certificate, it can query the Control Service of the neighboring AS that sent the PCB through the API described in Figure 30.

Note: The Control Service of an AS is decoupled from SCION border routers. The Control Service of a specific AS is part of the Control Plane, is responsible for _finding and registering suitable paths_, and can be deployed anywhere inside the AS. Border routers are deployed at the edge of an AS and their main tasks are to _forward data packets_.

1.4.1. Path Segments

SCION distinguishes the following types of path segments:

* A path segment from a non-core AS to a core AS is an _up segment_.

* A path segment from a core AS to a non-core AS is a _down segment_.

* A path segment between core ASes is a _core segment_.

Each path segment starts and/or ends at a core AS.

All path segments may be reversed: a core segment can be used bidirectionally, an up segment can be converted into a down segment, and a down segment can be converted into an up segment depending on the direction of the end-to-end path. This means that all path segments can be used to send data traffic in both directions.

The cryptographic protection of PCBs and path segments is based on the Control Plane PKI. The signatures are structured such that the entire message sequence constituting the path segment can be authenticated by anyone with access to this PKI.

For fast validation of the path information carried in individual packets during packet forwarding, symmetric key cryptography is used and the Hop Fields contain a MAC. These MACs are structured to allow verification of the sequence of hops, but in contrast to the PCBs can only be validated by the border routers of the respective AS.

1.5. Addressing

Inter-domain SCION routing is based on an <ISD, AS> tuple. Although a complete SCION address is composed of the <ISD, AS, endpoint address> 3-tuple, the endpoint address is not used for inter-domain routing or forwarding. The endpoint address is of variable length, does not need to be globally unique, and can thus be an IPv4, IPv6, or link layer address.

Note: As a consequence of the fact that SCION relies on existing routing protocols (e.g. IS-IS, OSPF, SR) and communication fabric (e.g. IP, MPLS) for intra-domain forwarding, existing internal routers do not need to be changed to support SCION.

1.5.1. ISD Numbers

An ISD number is the 16-bit global identifier for an ISD and MUST be globally unique.

The following table gives an overview of the ISD number allocation:

ISD	Description
0	The wildcard ISD
1 - 15	Reserved for documentation and sample code (analogous to [RFC5398]).
16 - 63	Private use (analogous to [RFC6996]) - can be used for testing and private deployments
64 - 4094	Public ISDs - should be allocated in ascending order, without gaps and "vanity" numbers
4095 - 65535	Unallocated

Table 1: ISD number allocations

ISD numbers are currently allocated by Anapaya, a provider of SCION-based networking software and solutions (see [ISD-AS-assignments-Anapaya]). This function is being transitioned to the SCION Association ([ISD-AS-assignments]).

1.5.2. SCION AS Numbers

A SCION AS number is the 48-bit identifier for an AS. Although they play a similar role, there is no relationship between SCION AS numbers and BGP ASNs as defined by [RFC4271]. For historical reasons some SCION Autonomous Systems use a SCION AS number where the first 16 bits are 0 and the remaining 32 bits are identical to their BGP ASN, but there is no technical requirement for this.

1.5.2.1. Wildcard Addressing

SCION endpoints use wildcard AS 0 to designate any core AS, e.g. to place requests for core segments or down segments during path lookup. These wildcard addresses are of the form I-0, to designate any AS in ISD I. For more information, see Section 5.2.1.

1.5.2.2. SCION AS numbers

AS	Size	Description
0	1	The wildcard AS
1 - 4294967295	~4.3 billion	Public SCION AS numbers
1:0:0 - 1:ffff:ffff	~4.3 billion	Unallocated
2:0:0 - 2:ffff:ffff	~4.3 billion	Public SCION AS numbers
3:0:0 - feff:ffff:ffff	~280 trillion	Unallocated
ff00:0:0 - ff00:0:ffff	65536	Reserved for documentation and sample code (analogous to [RFC5398])
ff00:1:0 - ffa9:ffff:ffff	~730 billion	Unallocated
ffaa:0:0 - ffaa:ff:ffff	~16.8 million	Reserved for private use (analogous to [RFC6996]) -

		these numbers can be used for testing and private deployments
ffaa:100:0 - ffff:ffff:fffe	~369 billion	Unallocated
ffff:ffff:ffff	1	Reserved

Table 2: AS number allocations

1.5.3. Text Representation

1.5.3.1. ISD numbers

The text representation of SCION ISD numbers MUST be its decimal ASCII representation.

1.5.3.2. AS numbers

The text representation of SCION AS numbers is as follows:

* SCION AS numbers in the lower 32-bit range MUST be printed as decimal by implementations. Implementations may parse AS numbers in the lower 32-bit range in hexadecimal notation too (e.g., a program may accept AS number '0:1:f' for AS 65551).

* SCION AS numbers in the higher 32-bit range MUST be printed using big-endian hexadecimal notation in 3 groups of 4, in the range 1:0:0 to ffff:ffff:ffff. Leading zeros in each group are omitted, with the exception that one zero MUST be notated if a group is entirely zeros (e.g., 1:0:1). The :: zero-compression feature of IPv6 MUST NOT be used.

* A range of AS numbers can be shortened with a notation similar to the one used for CIDR IP ranges ([RFC4632]). In such case, hexadecimal notation MUST be used. For example, the range of the lowest 32-bit AS numbers (0-4294967295) can be represented as 0:0:0/16.

1.5.3.3. <ISD, AS> tuples

The text representation of SCION addresses MUST be <ISD>-<AS>, where <ISD> is the text representation of the ISD number, <AS> is the text representation of the AS number, and - is the literal ASCII character 0x2D.

For example, the text representation of AS number ff00:0:1 in ISD number 15 is 15-ff00:0:1.

1.6. Bootstrapping ability

A secure and reliable routing architecture has to be designed specifically to avoid circular dependencies during network bootstrapping. One goal is that the SCION network can start up even after large outages or attacks, in addition to avoiding cascades of outages caused by fragile interdependencies. This section lists the concepts SCION uses to prevent circular dependencies:

* Neighbor-based path discovery: Path discovery in SCION is performed by the beaconing mechanism. In order to participate in this process, an AS only needs to be aware of its direct neighbors. As long as no path segments are available, communicating with the neighboring ASes is possible with the one-hop path type which does not rely on any path information. SCION uses these _one-hop paths_ to propagate PCBs to neighboring ASes to which no forwarding path is available yet. The One-Hop Path Type is described in more detail in [I-D.dekater-scion-dataplane].

* Path reversal: In SCION, every path is reversible. That is, the receiver of a packet can reverse the path in the packet header in order to produce a reply packet without having to perform a path lookup. Such a packet follows the original packet's path backwards.

* Availability of certificates: In SCION, every entity is required to be in possession of all cryptographic material (including the ISD's TRC and certificates) that is needed to verify any message it sends. This (together with the path reversal) means that the receiver of a message can always obtain all this material by contacting the sender. This avoids circular dependencies between the PKI and connectivity.

Note: For a detailed description of a TRC and more information on the availability of certificates and TRCs, see [I-D.dekater-scion-pki].

1.7. Resistance to partitioning

Partitioning occurs when a network splits into two because of link failures. Partitioning of the global SCION inter-domain network is much less likely to happen, thanks to its path awareness that exposes multiple paths between SCION ASes. Even during failures there is no special failure mode required as SCION-enabled ASes can always switch to already known paths that use other links.

Recovering from a partitioned network is also seamless as only coarse time synchronization between the partitions is required to resume normal operation and move forward with updates of the cryptographic material. Section 3.2 further describes the impact of time synchronization and path discovery time.

1.8. Communication Protocol

All communication between the Control Services in different ASes is expressed in terms of gRPC remote procedure calls (for details, see [gRPC]). Service interfaces and messages are defined in the Protocol Buffer "proto3" interface definition language (for details, see [proto3]).

The RPC messages are transported via the [Connect]’s rpc protocol; a gRPC-like protocol that carries messages over HTTP/3 (see [RFC9114])). HTTP3 traffic uses QUIC/UDP ([RFC9000]) as a transport layer. In the case of SCION, UDP relies on the data plane.

Appendix "Full Control Service gRPC API" provides the entire Control Service API definition in protobuf format.

Appendix "Use of the SCION Data Plane" provides details about the establishment of the underlying QUIC connections through the data plane.

2. Path Exploration or Beaconing

2.1. Introduction and Overview

Path Exploration is the process where an AS discovers paths to other ASes. In SCION, this process is referred to as _beaconing_ and this section provides a detailed explanation of this.

In SCION, the _Control Service_ of each AS is responsible for the beaconing process. The Control Service generates, receives, and propagates the _Path Segment Construction Beacons (PCBs)_ on a regular basis, to iteratively construct path segments.

PCBs contain inter-domain topology and authentication information, and can also include additional metadata that helps with path management and selection. The beaconing process itself is divided into routing processes on two levels, where _inter-ISD_ or core beaconing is based on the (selective) sending of PCBs without a defined direction, and _intra-ISD_ beaconing on top-to-bottom propagation. Beaconing is initiated by core ASes, therefore each ISD MUST have at least one core AS.

* _Inter-ISD or core beaconing_ is the process of constructing path segments between core ASes in the same or in different ISDs. During core beaconing, the Control Service of a core AS either initiates PCBs or propagates PCBs received from neighboring core ASes to other neighboring core ASes. PCBs are periodically sent over policy compliant paths to discover multiple paths between any pair of core ASes.

* _Intra-ISD beaconing_ creates path segments from core ASes to non-core ASes. For this, the Control Services of core ASes create PCBs and sends them to the non-core child ASes (typically customer ASes) at regular intervals. The Control Service of a non-core child AS receives these PCBs and forwards them to its child ASes, and so on until the PCB reaches an AS without any children (leaf AS). As a result, all ASes within an ISD receive path segments to reach the core ASes of their ISD and register reciprocal segments with the Control Service of the associated core ASes.

On its way, a PCB accumulates cryptographically protected path and forwarding information per traversed AS. At every AS, metadata as well as information about the AS's ingress and egress interfaces is added to the PCB. The full PCB message format is described in Section 2.2. PCBs are used to construct path segments. ASes register them to make them available to other ASes, as described in Section 4.

2.1.1. Peering Links

PCBs do not traverse peering links. Instead, peering links are announced along with a regular path in a PCB. If both ASes at either end of a peering link have registered path segments that include this specific peering link, then it is possible to use this peering link during segment combination to create the end-to-end path.

2.1.2. Appending Entries to a PCB

Every propagation interval (as configured by the AS), the Control Service:

* selects the best combinations of PCBs and interfaces connecting to a neighboring AS (i.e. a child AS or a core AS). This is described in Section 2.3.3.

* propagates each selected PCB to the selected egress interface(s) associated with it. This is described in Section 2.3.5.

For every selected PCB and egress interface combination, the AS appends an _AS entry_ to the selected PCB. This includes a Hop Field that specifies the ingress and egress interface for the packet forwarding through this AS, in the beaconing direction. The AS entry can also contain peer entries.

2.1.3. PCB Propagation - Illustrated Examples

The following three figures show how intra-ISD PCB propagation works, from the ISD's core AS down to child ASes. Interface identifiers of each AS are numbered with integer values while ASes are described with an upper case letter for the sake of illustration.

In Figure 2 below, core AS X sends the two different PCBs "a" and "b" via two different links to child AS Y: PCB "a" leaves core AS X via egress interface "2", whereas PCB "b" is sent over egress interface "1". Core AS X adds the respective egress information to the PCBs when sending them off, as can be seen in the figure (the entries "_Core - Out:2_" and "_Core - Out:1_", respectively).

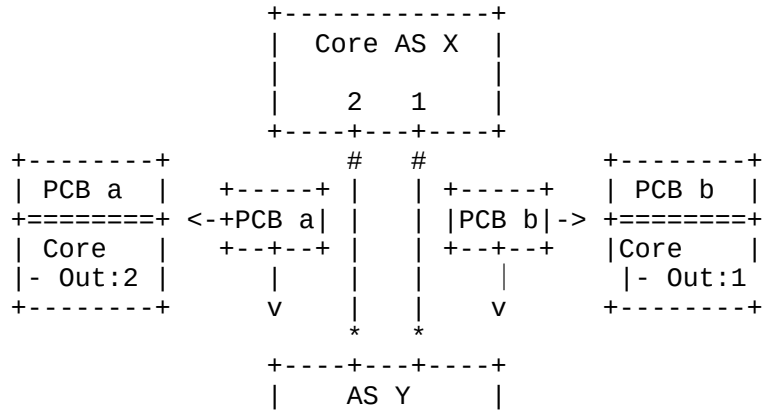


Figure 2: Intra-ISD PCB propagation from the ISD core to child ASes - Part 1

AS Y receives the two PCBs "a" and "b" through two different (ingress) interfaces, namely "2" and "3", respectively (see Figure 3 below). Additionally, AS Y forwards to AS Z four PCBs that were previously sent by core AS X. For this, AS Y uses the two different (egress) links "5" and "6". AS Y appends the corresponding ingress and egress interface information to the four PCBs. As can be seen in the figure, AS Y also has two peering links to its neighboring peers V and W, through the interfaces "1" and "4", respectively - AS Y includes this information in the PCBs, as well. Thus, each forwarded PCB cumulates path information on its way "down" from core AS X.

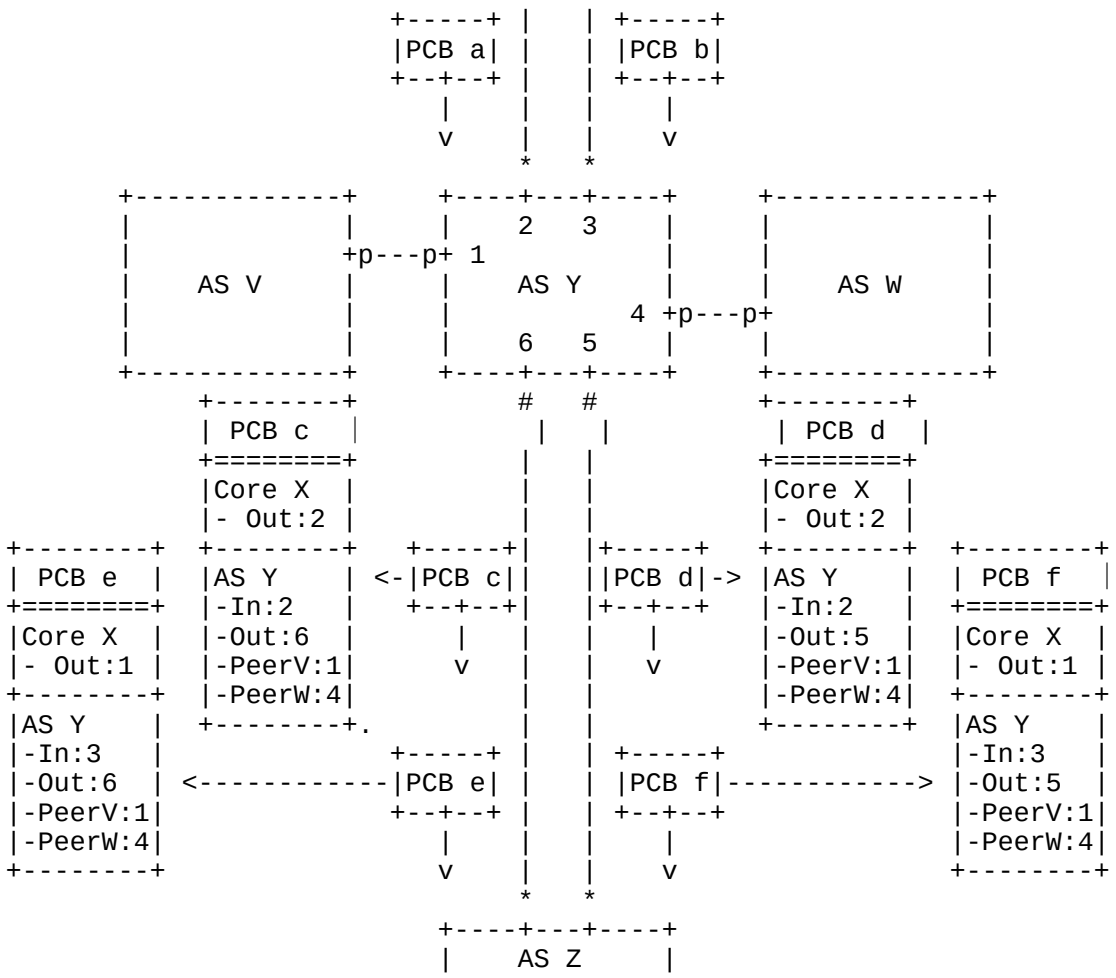


Figure 3: Intra-ISD PCB propagation from the ISD core to child ASes - Part 2

The following figure shows how the four PCBs "c", "d", "e", and "f", coming from AS Y, are received by AS Z over two different links: PCBs "c" and "e" reach AS Z over ingress interface "5", whereas PCBs "d" and "f" enter AS Z via ingress interface "1". Additionally, AS Z propagates PCBs "g", "h", "i", and "j" further down, all over the same link (egress interface "3"). AS Z extends the PCBs with the relevant information, so that each of these PCBs now includes AS hop entries from core AS X, AS Y, and AS Z.

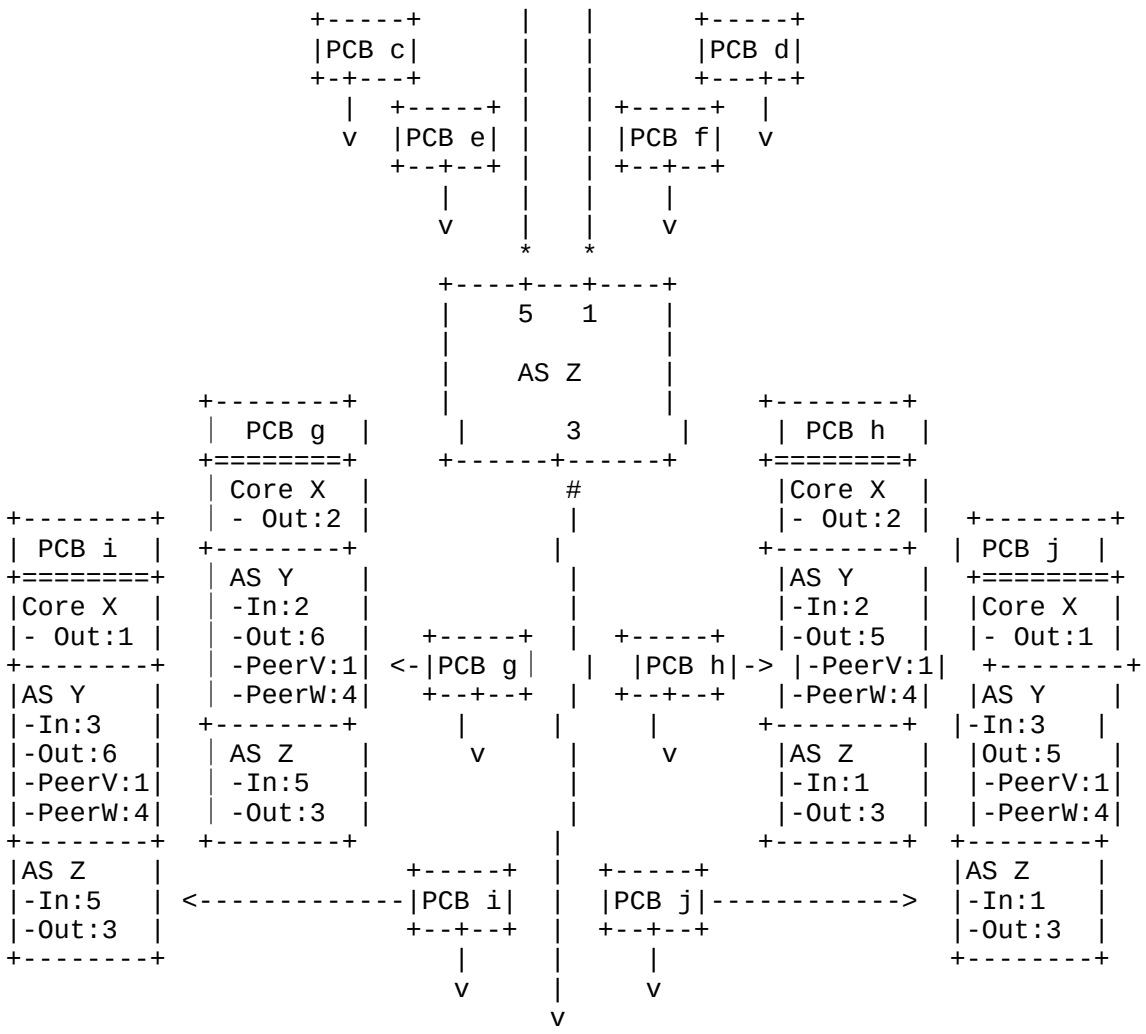


Figure 4: Intra-ISD PCB propagation from the ISD core to child ASes - Part 3

Based on the figures above, one could say that a PCB represents a single path segment. However, there is a difference between a PCB and a registered path segment as a PCB is a so-called "travelling path segment" that accumulates AS entries when traversing the Internet. A registered path segment is instead a "snapshot" of a travelling PCB at a given time T and from the vantage point of a particular AS A. This is illustrated by Figure 5. This figure shows several possible path segments to reach AS Z, based on the PCBs "g", "h", "i", and "j" from Figure 4 above. It is up to AS Z to use all of these path segments or just a selection of them.

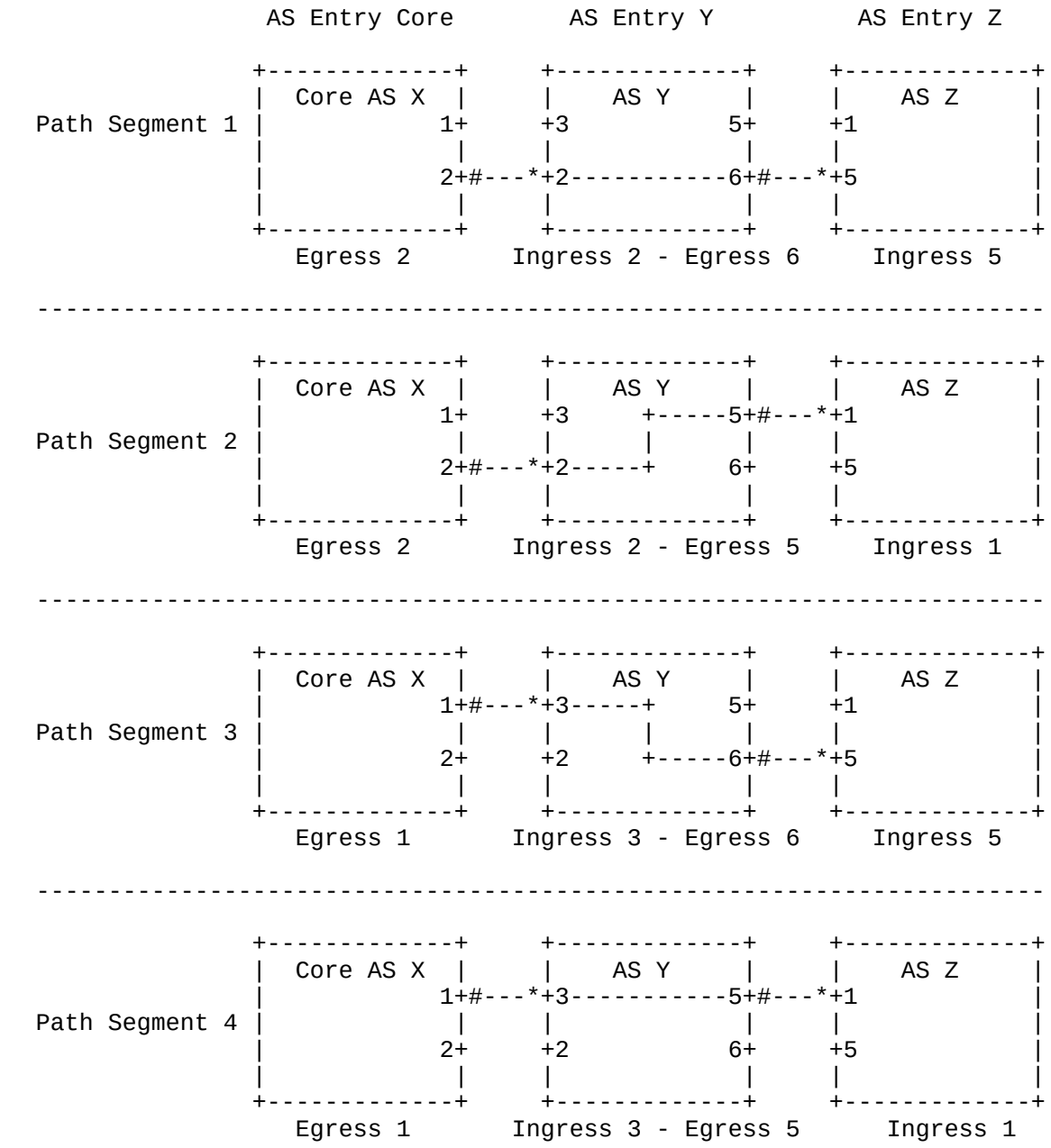
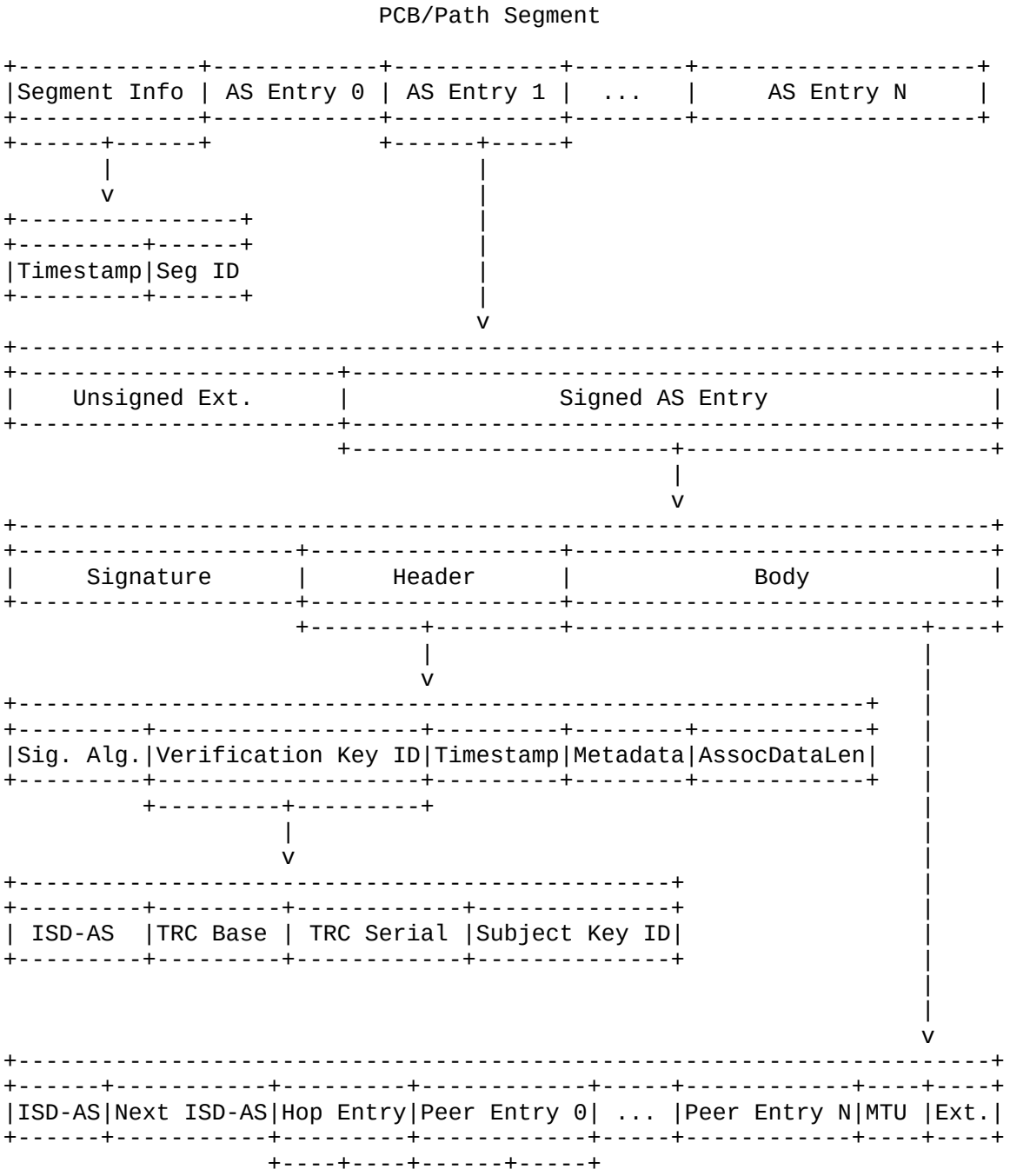


Figure 5: Possible up- or down segments for AS Z

2.2. Path-Segment Construction Beacons (PCBs)

2.2.1. PCB Message Format



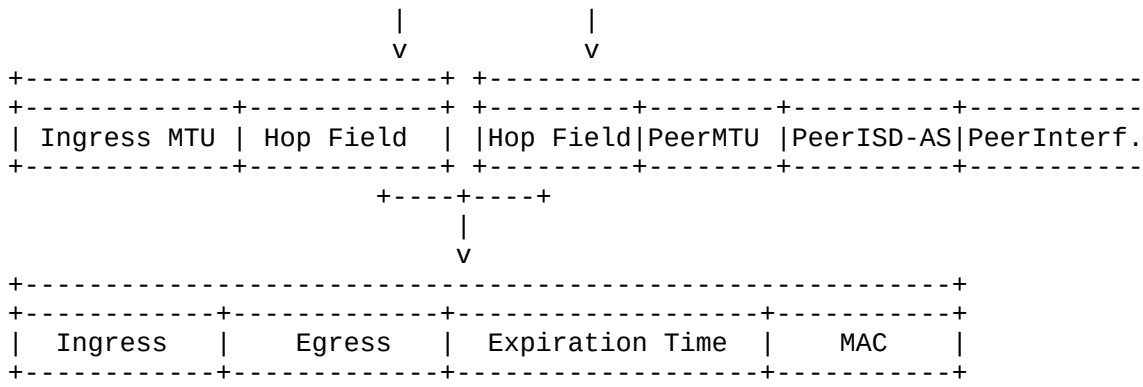


Figure 6: Top-down composition of a PCB

Note: For a full example of a PCB in the Protobuf message format, please see Figure 28.

2.2.1.1. PCB Top-Level Message Format

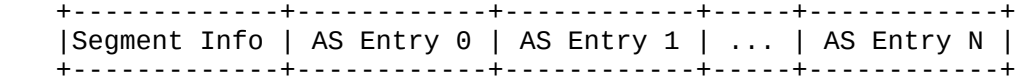


Figure 7: PCB Top-Level Message Format

Each PCB MUST consists of at least:

* An information field with an identifier and a timestamp.

* Entries of all ASes on the path segment represented by this PCB.

The following code block defines the PCB at the top level in Protobuf message format.

```
message PathSegment {
    bytes segment_info = 1;
    repeated ASEntry as_entries = 2;
}
```

* segment_info: This field is used as input for the PCB signature. It is the encoded version of the component SegmentInformation, which provides basic information about the PCB. The SegmentInformation component is specified in detail in Section 2.2.1.2.

* as_entries: Contains the ASEntry component of all ASes on the path segment represented by this PCB.

  - ASEntry: The ASEntry component contains the complete path information of a specific AS that is part of the path segment represented by the PCB. The ASEntry component is specified in detail in Section 2.2.1.3.

2.2.1.2. Segment Information

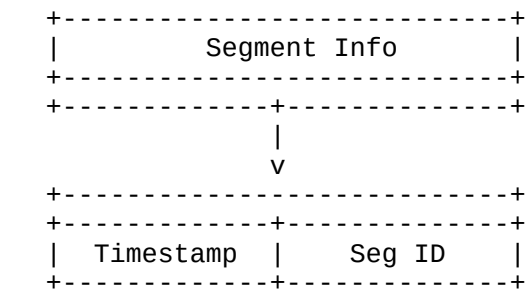


Figure 8: Segment Information Component

Each PCB MUST include an information component with basic information about the PCB.

In the Protobuf message format, the information component of a PCB is called the SegmentInformation message. The following code block shows the Protobuf message definition for the SegmentInformation message.

```
message SegmentInformation {
    int64 timestamp = 1;
    uint32 segment_id = 2;
}
```

* timestamp: The 32-bit timestamp indicates the creation time of this PCB. It is set by the originating core AS and the expiration time of each Hop Field in the PCB is computed relative to this timestamp. The timestamp is encoded as the number of seconds elapsed since the POSIX Epoch (1970-01-01 00:00:00 UTC).

* segment_id: The 16-bit identifier of this PCB and the corresponding path segment. The segment ID is REQUIRED for the computation of the message authentication code (MAC) of an AS's Hop Field. The MAC is used for Hop Field verification in the data plane and the originating core AS MUST fill this field with a cryptographically random number.

Note: See Section 2.2.1.6 for more information on the Hop Field message format. [I-D.dekater-scion-dataplane] provides a detailed description of the computation of the MAC and the verification of the Hop Field in the data plane.

2.2.1.3. AS Entry

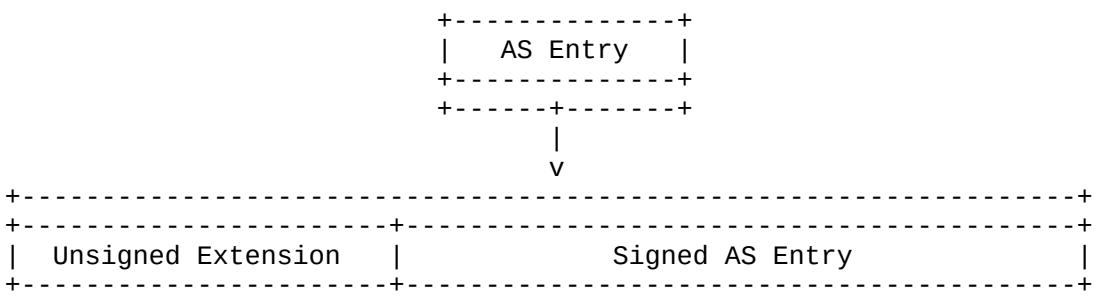


Figure 9: AS Entry

Each PCB MUST also contain the entries of all ASes included in the corresponding path segment. This means that the originating core AS MUST add its AS entry to each PCB it creates, and each traversed AS MUST attach its AS entry to the PCB.

One AS entry contains the complete hop information for this specific AS in this specific path segment. It consists of a signed and an unsigned component.

The code block below defines an AS entry ASEntry in Protobuf message format.

```
message ASEntry {
    SignedMessage signed = 1;
    PathSegmentUnsignedExtensions unsigned = 2;
}
```

It includes the following components:

* SignedMessage: The signed component of an AS entry. For the specification of this part of the AS entry, see Section 2.2.1.4 below.

* PathSegmentUnsignedExtensions: The unsigned and thus unprotected part of the AS entry. These are extensions with metadata that need no explicit protection.

2.2.1.4. AS Entry Signed Component

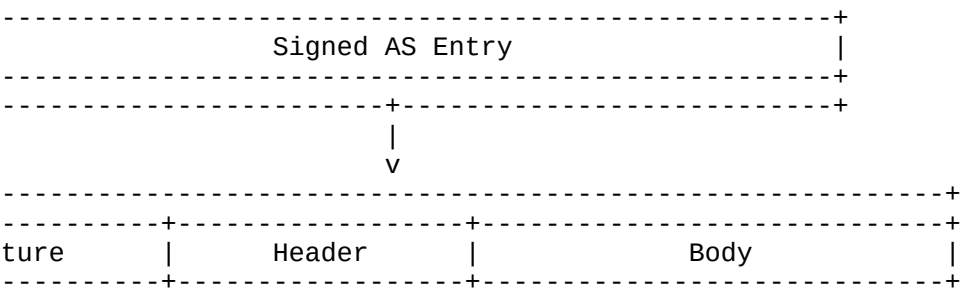


Figure 10: AS Entry Signed Component

Each AS entry of a PCB MUST include a signed component as well as a signature computed over the signed component. Each AS entry MUST be signed with the Control Plane AS Certificate (see [I-D.dekater-scion-pki]).

The signed component of an AS entry MUST include the following elements:

* a header,

* a body, and

* a signature.

In the Protobuf message format implementation, the signed component of an AS entry is specified by the SignedMessage. It consists of a header-and-body part (header_and_body) and a raw signature (signature). See also the code block below.

```
message SignedMessage {
    bytes header_and_body = 1;
    bytes signature = 2;
}
```

The following code block shows the low level representation of the HeaderAndBodyInternal message used for signature computation input. This message SHOULD NOT be used by external code.

```
message HeaderAndBodyInternal {
    // Encoded header suitable for signature computation.
    bytes header = 1;
    // Raw payload suitable for signature computation.
    bytes body = 2;
}
```

* For the specification of the signed header, see Section 2.2.1.4.1.

* For the specification of the signed body, see Section 2.2.1.4.2.

* For the specification of the signature field, see Section 2.2.1.4.3.

2.2.1.4.1. AS Entry Signed Header

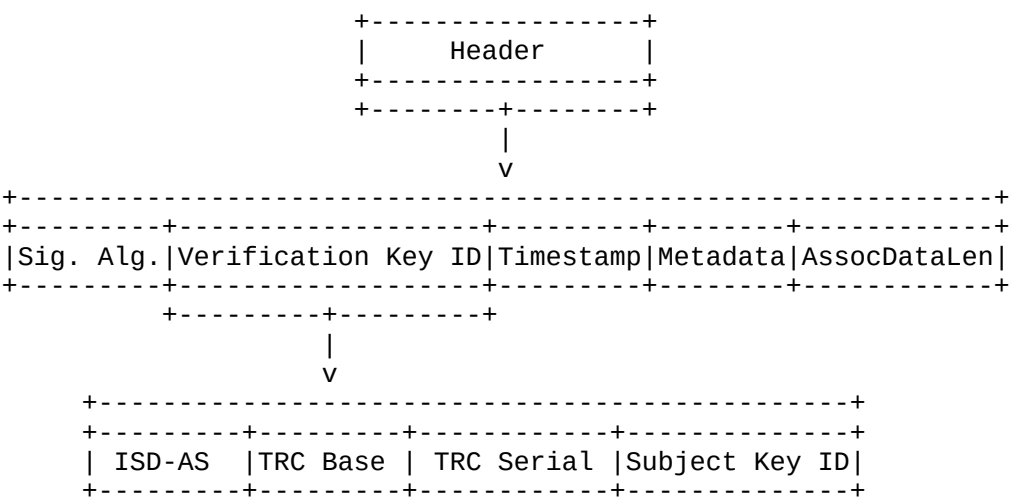


Figure 11: AS Entry Signed Header

The header part defines metadata that is relevant to the computation and verification of the signature. It MUST include at least the following metadata:

* The algorithm to compute the signature

* The subjectKeyIdentifier of the public key to be used to verify the signature (see [I-D.dekater-scion-pki])

* The ISD-AS number of the AS

The following code block defines the signed header of an AS entry in Protobuf message format (called the Header message).

```
message Header {
    SignatureAlgorithm signature_algorithm = 1;
    bytes verification_key_id = 2;
    // Optional
    google.protobuf.Timestamp timestamp = 3;
    // Optional
    bytes metadata = 4;
    int32 associated_data_length = 5;
}

message VerificationKeyID {
    uint64 isd_as = 1;
    bytes subject_key_id = 2;
    uint64 trc_base = 3;
    uint64 trc_serial = 4;
}
```

* signature_algorithm: Specifies the algorithm to compute the signature.

* verification_key_id: Contains information that is relevant to signing and verifying PCBs and other control-plane messages. It includes the following fields (see also the above code block):

  - isd_as: The ISD-AS number of the current AS.

  - subject_key_id: Refers to the certificate that contains the public key needed to verify this PCB’s signature.

  - trc_base: Defines the _base_ number of the latest Trust Root Configuration (TRC) available to the signer at the time of the signature creation.

  - trc_serial: Defines the _serial_ number of the latest TRC available to the signer at the time of the signature creation.

Note: For more information on signing and verifying control plane messages (such as PCBs), see ’Signing and Verifying Control Plane Messages’ in [I-D.dekater-scion-pki]. For more information on the TRC base and serial number, see ’Trust Root Configuration Specification’ in [I-D.dekater-scion-pki].

* timestamp: Defines the signature creation timestamp. This field is OPTIONAL.

* metadata: Can be used to include arbitrary per-protocol metadata. This field is OPTIONAL.

* associated_data_length: Specifies the length of associated data that is covered by the signature, but is not included in the header and body. The value of this field is zero, if no associated data is covered by the signature.

2.2.1.4.2. AS Entry Signed Body

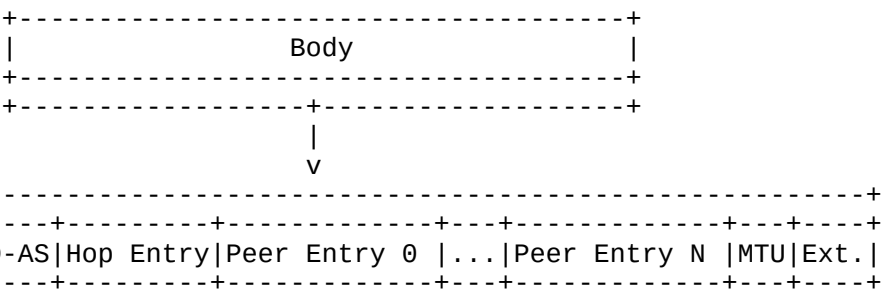


Figure 12: AS Entry Signed Body

The body of an AS entry MUST consist of the signed component ASEntrySignedBody of all ASes in the path segment represented by the PCB, up until and including the current AS.

The following code block defines the signed body of one AS entry in Protobuf message format (called the ASEntrySignedBody message).

```
message ASEntrySignedBody {
    uint64 isd_as = 1;
    uint64 next_isd_as = 2;
    HopEntry hop_entry = 3;
    repeated PeerEntry peer_entries = 4;
    uint32 mtu = 5;
    PathSegmentExtensions extensions = 6;
}
```

* isd_as: The ISD-AS number of the AS that created this AS entry.

* next_isd_as: The ISD-AS number of the downstream AS to which the PCB MUST be forwarded. The presence of this field prevents path hijacking attacks, as further discussed in Section 7.3.1.

* hop_entry: The hop entry (HopEntry) with the information required to forward this PCB through the current AS to the next AS. This information is used in the data plane. For a specification of the hop entry, see Section 2.2.1.5.

* peer_entries: The list of optional peer entries (PeerEntry). For a specification of one peer entry, see Section 2.2.1.7.

* mtu: The maximum transmission unit (MTU) that is supported by all intra-domain links within the current AS. This value is set by the control service when adding the AS entry to the beacon. How the control service obtains this information is implementation dependent. Current practice is to make it a configuration item.

* extensions: List of (signed) extensions (optional). PCB extensions defined here are part of the signed AS entry. This field SHOULD therefore only contain extensions that include important metadata for which cryptographic protection is required. For more information on PCB extensions, see Section 2.2.2.

2.2.1.4.3. AS Entry Signature

Each AS entry MUST be signed with the AS certificate's private key K_i. The certificate MUST have a validity period fully containing that of the segment being verified, regardless of current time. The signature Sig_i of an AS entry ASE_i is computed over the AS entry's signed component.

This is the input for the computation of the signature:

* The signed header and body of the current AS (header_and_body).

* The segment_info component of the current AS. This is the encoded version of the SegmentInformation component containing basic information about the path segment represented by the PCB. For the specification of SegmentInformation, see Section 2.2.1.2.

* The signed header_and_body/signature combination of each previous AS on this specific path segment.

The signature Sig_i of an AS entry ASE_i is now computed as follows:

```
Sig_i = K_i( SegInfo || ASE_0^(signed) || Sig_0 || ... ||
ASE_(i-1)^(signed) || Sig_(i-1) || ASE_i^(signed) )
```

The signature metadata minimally contains the ISD-AS number of the signing entity and the key identifier of the public key to be used to verify the message. For more information on signing and verifying control plane messages, see 'Signing and Verifying Control Plane Messages' in [I-D.dekater-scion-pki].

The following code block shows how the signature input is defined in the SCION Protobuf implementation ("ps" stands for path segment). Note that the signature has a nested structure.

```
input(ps, i) = signed.header_and_body || associated_data(ps, i)

associated_data(ps, i) = ps.segment_info ||
                         ps.as_entries[1].signed.header_and_body ||
                         ps.as_entries[1].signed.signature ||
                         ...
                         ps.as_entries[i-1].signed.header_and_body ||
                         ps.as_entries[i-1].signed.signature
```

2.2.1.5. Hop Entry

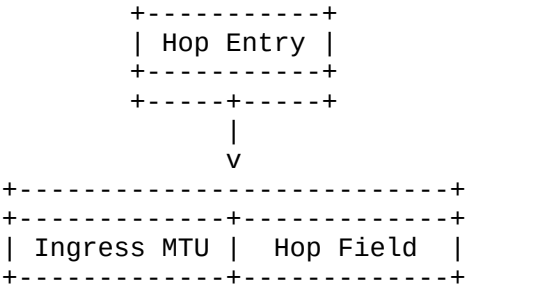


Figure 13: Hop Entry

Each body of an AS entry MUST contain exactly one hop entry component. The hop entry component specifies forwarding information which the data plane requires to create the hop through the current AS (in the direction of the beaconing).

The following code block defines the hop entry component HopEntry in Protobuf message format:

```
message HopEntry {
    HopField hop_field = 1;
    uint32 ingress_mtu = 2;
}
```

* hop_field: Contains the authenticated information about the ingress and egress interfaces in the direction of beaconing. Routers need this information to forward packets through the current AS. For further specifications, see Section 2.2.1.6.

* ingress_mtu: Specifies the maximum transmission unit (MTU) of the ingress interface (in beaconing direction) of the hop being described. The MTU of paths constructed from the containing beacon is necessarily less than or equal to this value. How the control service obtains the MTU of an inter-AS link is implementation dependent. It may be discovered or configured. Current practice to make it a configuration item.

In this description, MTU and packet size are to be understood in the same sense as in [RFC1122]. That is, exclusive of any layer 2 framing or packet encapsulation (for links using an underlay network).

2.2.1.6. Hop Field

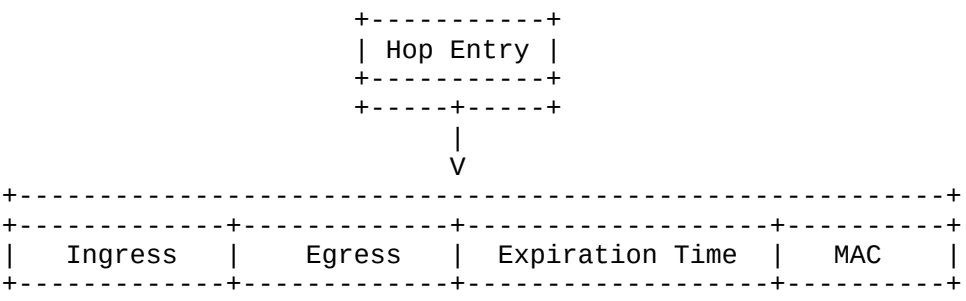


Figure 14: Hop Field

The Hop Field, part of both hop entries and peer entries, is used directly in the data plane for packet forwarding and specifies the incoming and outgoing interfaces of the ASes on the forwarding path. This information is authenticated with a Message Authentication Code (MAC) which is used by the Control Service of an AS to authenticate path segments with its border routers during packet forwarding.

The algorithm used to compute the Hop Field MAC is an AS-specific choice, although the Control Services and border routers within an AS MUST use the same algorithm. Implementations MUST also support the Default Hop Field MAC algorithm. See [I-D.dekater-scion-dataplane] section "Authorizing Segments through Chained MACs") for more information including configuration. Endpoints do not compute MACs.

The following code block defines the Hop Field component HopField in Protobuf message format:

```
message HopField {
    uint64 ingress = 1;
    uint64 egress = 2;
    uint32 exp_time = 3;
    bytes mac = 4;
}
```

* ingress: The 16-bit ingress interface identifier (in the direction of the path construction, that is, in the direction of beaconing through the current AS).

Note: For the core AS that initiates the PCB, the ingress interface identifier MUST be set to the "unspecified" value (see [I-D.dekater-scion-dataplane] section Terminology).

* egress: The 16-bit egress interface identifier (in the direction of beaconing).

* exp_time: The 8-bit encoded expiration time of the Hop Field, indicating its validity. This field expresses a duration in seconds according to the formula: duration = (1 + exp_time) * (24*60*60/256). The minimum duration is therefore 337.5 s. This duration is relative to the PCB creation timestamp set in the PCB's segment information component (see also Section 2.2.1.2). Therefore, the absolute expiration time of the Hop Field is the sum of these two values.

* mac: The message authentication code (MAC) used in the data plane to verify the Hop Field, as described in [I-D.dekater-scion-dataplane].

2.2.1.7. Peer Entry

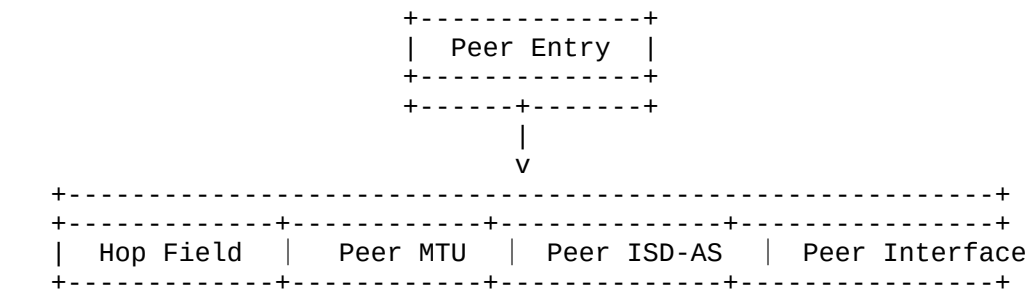


Figure 15: Peer Entry

By means of a peer entry, an AS can announce that it has a peering link to another AS. A peer entry is an optional component of a PCB - it is only included if there is a peering link to a peer AS.

The following code block defines the peer entry component PeerEntry in Protobuf message format:

```
message PeerEntry {
    uint64 peer_isd_as = 1;
    uint64 peer_interface = 2;
    uint32 peer_mtu = 3;
    HopField hop_field = 4;
}
```

* peer_isd_as: The ISD-AS number of the peer AS. This number is used to match peering segments during path construction.

* peer_interface: The 16-bit interface identifier of the peering link on the peer AS side. This identifier is used to match peering segments during path construction.

* peer_mtu: Specifies the maximum transmission unit (MTU) of the peering link being described. The MTU of paths via such link is necessarily less than or equal to this value. How the control service obtains the MTU of an inter-AS link is implementation dependent. It may be discovered or configured, but current practice is to make it a configuration item.

* hop_field: Contains the authenticated information about the ingress and egress interfaces in the current AS (coming from the peering link, in the direction of beaconing - see also Figure 7). The data plane needs this information to forward packets through the current AS. For further specifications, see Section 2.2.1.6.

In this description, MTU and packet size are to be understood in the same sense as in [RFC1122]. That is, exclusive of any layer 2 framing or packet encapsulation (for links using an underlay network).

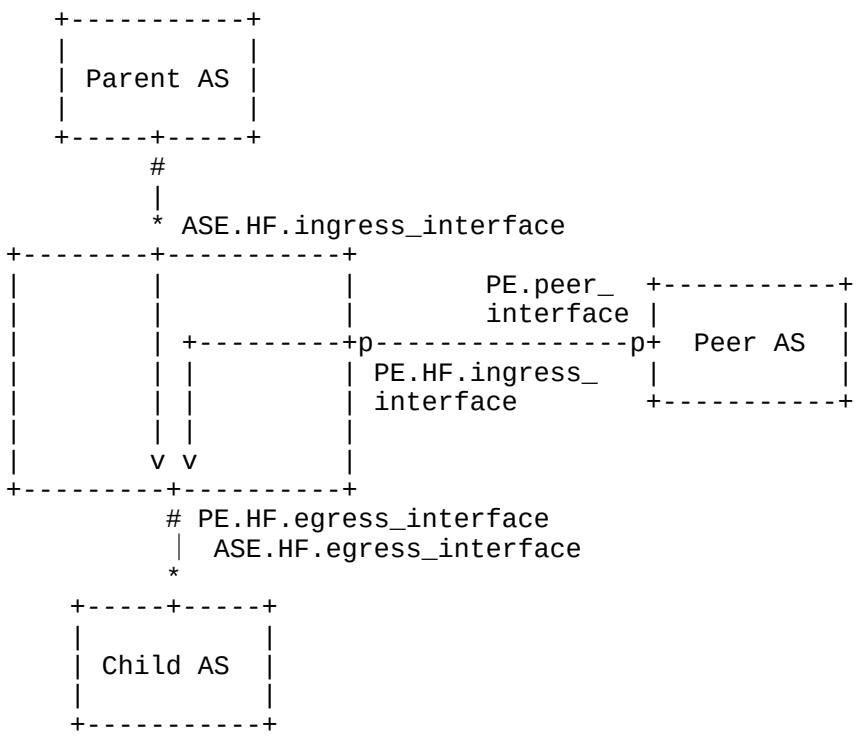


Figure 16: Peer entry information, in the direction of beaconing

2.2.2. PCB Extensions

In addition to basic routing information such a hop entries and peer entries, PCBs can be used to communicate additional metadata in extensions. Extensions can be signed and unsigned: signed extensions are protected by the AS signature, whereas unsigned extensions are not.

In Protobuf, extensions are specified as follows:

* Unsigned extensions PathSegmentUnsignedExtensions are part of the AS entry component (the ASEntry message, see also Section 2.2.1.3).

* Signed extensions PathSegmentExtensions are part of the signed body component of an AS entry (the ASEntrySignedBody message, see also Section 2.2.1.4.2).

Note: SCION also supports so-called "detachable extensions". The detachable extension is part of a PCB's unsigned extensions, but a cryptographic hash of the detachable extension data is added to the signed extensions. Thus, a PCB with a detachable extension can be signed and verified without actually including the detachable extension in the signature. This prevents a possible processing overhead caused by large cryptographically-protected extensions.

2.2.3. PCB Validity

To be valid (that is, usable to construct a valid path), a PCB MUST:

* Contain valid AS Entry signatures (Section 2.2.1.4.3).

* Have a timestamp (Section 2.2.1.2) that is not in the future.

* Contain only unexpired hops (Section 2.2.1.6).

For the purpose of validation, a timestamp is considered "future" if it is later than the current time at the point of validation plus an allowance for differences between the validator's and originator's clock. As an allowance, it is recommended to use the granularity of the hopfield expiration time (that is 337.5 seconds, see Section 2.2.1.6).

For the purpose of validation, a hop is considered expired if its absolute expiration time, calculated as defined in Section 2.2.1.6, is later than the current time at the point of validation.

2.2.4. Configuration

For the purpose of constructing and propagating path segments, an AS Control Service must be configured with links to neighboring ASes. Such information may be conveyed to the Control Service in an out-of-band fashion (e.g in a configuration file). For each link, these values MUST be configured:

* Local Interface ID. This MUST be unique within each AS.

* Neighbor type (core, parent, child, peer), depending on link type (see Section 1.3). Link type depends on mutual agreements between the organizations operating the ASes at each end of each link.

* Neighbor ISD-AS number

* Neighbor interface underlay address

The maximum MTU supported by all intra-AS links MAY also be configured.

The AS SHOULD adopt a PCB selection policy that does not accidentally isolate the AS from the network, i.e. such that it does not block connectivity to parent providers and ensures downstream connectivity for children. For more details, see Section 2.3.3.

2.3. Propagation of PCBs

This section describes how PCBs are received, selected and further propagated in the path exploration process.

2.3.1. Reception of PCBs

Upon receiving a PCB, the Control Service of an AS performs the following checks:

1. PCB validity: It verifies the validity of the PCB (see Section 2.2.3) and invalid PCBs MUST be discarded. The PCB contains the version numbers of the TRC(s) and certificate(s) that MUST be used to verify its signatures which enables the Control Service to check whether it has the relevant TRC(s) and certificate(s). If not, they can be requested from the Control Service of the sending AS through the API described in Figure 30.

2. Loop avoidance: If it is a core AS, the Control Service MUST check whether the PCB includes duplicate hop entries created by the core AS itself or by other ASes. If so, the PCB MUST be discarded in order to avoid loops. This step is necessary because core beaconing is based on propagating PCBs to all AS neighbors. Additionally, core ASes SHOULD discard PCBs that were propagated at any point by a non-core AS. Ultimately, core ASes MAY make a policy decision not to propagate beacons containing path segments that traverse the same ISD more than once as this can be legitimate, e.g. if the ISD spans a large geographical area, a path transiting another ISD may constitute a shortcut.

3. Incoming Interface: the last ISD-AS entry in a received PCB (in its AS Entry Signed Body) MUST coincide with the ISD-AS neighbor of the interface where the PCB was received. If not, the PCB MUST be discarded.

4. Continuity: when a PCB contains two or more AS entries, the receiver Control Service MUST check every AS entry except the last and discard beacons where the ISD-AS of an entry does not equal the ISD-AS of the next entry.

If the PCB verification is successful, the Control Service decides whether to store the PCB as a candidate for propagation based on selection criteria and polices specific for each AS.

2.3.2. Storing Candidate PCBs

An AS stores candidate PCBs in a temporary storage called the _Beacon Store_. The management of this storage is implementation defined.

Current practice is to retain all PCBs until expired or replaced by one describing the same path with a later origination time.

2.3.3. PCB Selection Policies

An AS MUST select which PCBs to propagate further. The selection process can inspect and compare the properties of the candidate PCBs (e.g. length, disjointness across different paths, age, expiration time) and/or take into account which PCBs have been propagated in the past. The PCBs to select or eliminate is determined by the policy of the AS.

In order to avoid excessive overhead on the path discovery system in bigger networks, an AS should only propagate those candidate PCBs with the highest probability of meeting the needs of the endpoints that will perform path construction, in accordance with Section 2.3.4.

As SCION does not provide any in-band signal about the intentions of endpoints nor about the policies of downstream ASes, the policy will typically select a somewhat diverse set optimized for multiple, generic parameters. Selection may be based on criteria such as:

* AS path length: from the originator core AS to the child (non-core) AS.
* Expiration time: the maximum value for the expiration time when extending the segment.
* ISD or AS exclusion lists: certain ASes or ISD that may not appear in a segment.
* ISD loops: if permitted, they allow core AS to reach other core ASes in the same ISD via a third party ISDs.
* Availability of peering links: that is the number of different peering ASes from all non-core ASes on the PCB or path segment. A greater number of peering ASes increases the likelihood of finding a shortcut on the path segment.

* Path disjointness: Paths can be either AS disjointed or link disjointed. AS disjointed paths have no common upstream/core AS for the current AS, whereas link disjointed paths do not share any AS-to-AS link. AS disjointness allows path diversity in the event that an AS becomes unresponsive, and link disjointness provides resilience in case of link failure. Both criteria can be used depending on the objective of the AS.

The relative disjointness of two PCBs A and B may be calculated by assigning a disjointness score, calculated as the number of links in A that don't appear in B. For example, the beacon that has the highest disjointness score and is not the shortest path should be selected, but if this not better than what has already been selected, then the beacon with the shortest path yet to be selected should be chosen instead.

A PCB Selection Policy can be expressed as a stateful filter of segments, i.e., a function which indicates whether to accept or deny a given segment. This filter is stateful in that it can be updated each time its AS registers a new segment. Naturally, an AS's policy selects PCBs corresponding to paths that are commercially or otherwise operationally viable.

2.3.4. Propagation Interval and Best PCBs Set Size

PCBs are propagated in batches to each neighboring AS at a fixed frequency known as the _propagation interval_ which happens for both intra-ISD beaconing and core beaconing. At each propagation event, each AS selects a set of the best PCBs from the candidates in the Beacon Store according to the AS's selection policy. This set SHOULD have a fixed size, the _best PCBs set size_.

The _best PCBs set size_ SHOULD be:

* For intra-AS beaconing (i.e. propagating to children ASes): at most 50.

* For core beaconing (i.e. propagation between core ASes): at most 5 per immediate neighbor core AS. Current practice is that each set of 5 is chosen among the PCBs received from each neighbor.

Note that the PCBs set size should not be too low, in order to make sure that beaconing can discover a wide amount of paths. Values above are RECOMMENDED maxima which represent a tradeoff between scalability and amount of paths discovered. In current practice the intra-ISD set size is typically 20.

Depending on the selection criteria, it may be necessary to keep more candidate PCBs than the _best PCBs set size_ in the Beacon Store in order to determine the best set of PCBs. If this is the case, an AS SHOULD have a suitable pre-selection of candidate PCBs in place in order to keep the Beacon Store capacity limited.

* The _propagation interval_ SHOULD be at least "5" (seconds) for intra-ISD beaconing and at least "60" (seconds) for core beaconing.

Note that to ensure establish quick connectivity, an AS MAY attempt to forward a PCB more frequently ("fast recovery"). Current practice is to increase the frequency of attempts if no PCB propagation is known to have succeeded within the last propagation interval:

* because the corresponding RPC failed;

* or because no beacon was available to propagate.

The scalability implications of such parameters are further discussed in Section 3.3.

2.3.5. Propagation of Selected PCBs

To bootstrap the initial communication with a neighboring beacon service, ASes use one-hop paths. This special kind of path handles beaconing between neighboring ASes for which no forwarding path may be available yet. It is the task of beaconing to discover such forwarding paths and the purpose of one-hop paths is to break this circular dependency. The One-Hop Path Type is described in more detail in [I-D.dekater-scion-dataplane].

2.3.5.1. Propagation of PCBs in Intra-ISD Beaconing

The propagation process in intra-ISD beaconing includes the following steps:

1. From the candidate PCBs stored in the Beacon Store, the Control Service of an AS selects the best PCBs to propagate to its downstream neighboring ASes, based on a selection algorithm specific for this AS.

2. The Control Service MUST add a new AS entry (see Section 2.2.1.3), including any Peer Entry information (see Section 2.2.1.7) the AS is configured to advertise to every selected PCB.

3. The Control Service MUST sign each selected, extended PCB and append the computed signature.

4. As a final step, the Control Service propagates each extended PCB to the neighboring AS specified in the new AS entry by invoking the SegmentCreationService.Beacon remote procedure call (RPC) in the Control Services of the neighboring ASes (see also Section 2.3.5.3).

2.3.5.2. Propagation of PCBs in Core Beaconing

The propagation process in core beaconing includes the following steps:

1. The core Control Service selects the best PCBs to forward to neighboring core ASes observed so far.

2. The service adds a new AS entry to every selected PCB which MUST include:

   * The egress interface to the neighboring core AS in the Hop Field component.

   * The ISD_AS number of the neighboring core AS in the signed body component of the AS entry.

3. The core Control Service MUST sign the extended PCBs and append the computed signature.

4. As a final step, the service propagates the extended PCBs to the neighboring core ASes by invoking the SegmentCreationService.Beacon remote procedure call (RPC) in the Control Services of the neighboring core ASes (see also Section 2.3.5.3).

2.3.5.3. Propagation of PCBs in Protobuf Message Format

The last step of the above described core and intra-ISD propagation procedures is implemented as follows in Protobuf message format:

```
service SegmentCreationService {
    rpc Beacon(BeaconRequest) returns (BeaconResponse) {}
}

message BeaconRequest {
    PathSegment segment = 1;
}

message BeaconResponse {}
```

The propagation procedure includes the following elements:

* SegmentCreationService: Specifies the service via which the extended PCB is propagated to the Control Service of the neighboring AS.
  - Beacon: Specifies the method that calls the Control Service at the neighboring AS in order to propagate the extended PCB.
* BeaconRequest: Specifies the request message sent by the Beacon method to the Control Service of the neighboring AS. It contains the following element:
  - PathSegment: Specifies the path segment to propagate to the neighboring AS. For more information on the Protobuf message type PathSegment, see Section 2.2.1.1.
* BeaconResponse: An empty message returned as an acknowledgement upon success.

3. Deployment Considerations

3.1. Monitoring Considerations

In order to maintain service availability, an AS SHOULD monitor the following aspects when deploying the SCION control plane:

* For routers (to enable correlation with link states): state of configured links (core, child, parent)
* For any control service:
  - Fraction of path lookups served successfully (see Section 5)
  - Time synchronization offset with other ASes (see Section 3.2)
  - Fraction of ASes found in non-expired segments for which a non-expired certificate exists

* For a core AS:
  - Fraction of core ASes (preferably only those to which the link is up) that can be found in non-expired core segments
  - Fraction of ASes, core or children, (preferably only those to which the link is up) whereto a beacon was initiated during the last propagation interval
  - Fraction of freshly propagated beacons for which at least one corresponding down segment has been registered (see Section 4)
* For a non-core AS:
  - Number of up segments available (may be just 0/non-0) younger than the propagation interval (or some multiple thereof).
  - Fraction of up segments that were successfully registered as down segments (see Section 4).
  - Fraction of children ASes (preferably only those to which the link is up) whereto a beacon was propagated during the last propagation interval

3.2. Effects of Clock Inaccuracy

A PCB originated by a given Control Service is validated by all the Control Services that receive it. All have different clocks and their differences affect the validation process:

* A fast clock at origination or a slow clock at reception will yield a lengthened expiration time for hops, and possibly an origination time in the future.
* A slow clock at origination or a fast clock at reception will yield a shortened expiration time for hops, and possibly an expiration time in the past.

This bias comes in addition to a structural delay: PCBs are propagated at a configurable interval - typically around one minute. As a result of this and the way they are iteratively constructed, PCBs with N hops may be validated up to N intervals (so maximally N minutes) after origination which creates a constraint on the expiration of hops. Hops of the minimal expiration time (337.5 seconds - see Section 2.2.1.6) would make any PCB describing a path longer than 5 hops expire. For this reason it is unadvisable to create hops with a short expiration time - they SHOULD be around 6 hours.

The Control Service and its clients authenticate each-other according to their respective AS’s certificate. Path segments are authenticated based on the certificates of the ASes that they refer to. The RECOMMENDED expiration time of a SCION AS certificate is between 3h and 3 days. Some deployments use up to 5 days. In comparison to these time scales, clock offsets in the order of minutes are immaterial.

Each administrator of a SCION Control Service is responsible for maintaining sufficient clock accuracy. No particular method is assumed by this specification.

3.3. Path Discovery Time and Scalability

The path discovery mechanism balances the number of discovered paths and the time it takes to discover them versus resource overhead of the discovery.

The resource costs for path discovery are as follows:

* Communication overhead is transmitting the PCBs and occasionally obtaining the required PKI material.

* Processing overhead is validating the signatures of the AS entries, signing new AS entries, and to a lesser extent, evaluating the beaconing policies.

* Storage overhead is both the temporary storage of PCBs before the next propagation interval, and the storage of complete discovered path segments.

All of these are dependent on the number and length of the discovered path segments, i.e. the total number of AS entries of the discovered path segments.

Interesting metrics for scalability and speed of path discovery are the time until all discoverable path segments have been discovered after a network bootstrap, and the time until a new link is usable. In general, the time until a specific PCB is built depends on its length, the propagation interval, and whether on-path ASes use "fast recovery".

At each AS, the PCB will be processed and propagated at the subsequent propagation event. As propagation events are not synchronized between different ASes, a PCB arrives at a random point in time during the interval and may be buffered before potentially being propagated. With a propagation interval T at each AS, the mean time until the PCB is propagated in one AS therefore is T / 2 and the mean total time for the propagation steps of a PCB of length L is at worst L * T / 2 (with a variance of L * T^2 / 12).

Note that link removal is not part of path discovery in SCION. For scheduled removal of links, operators let path segments expire. On link failures, endpoints route around the failed link by switching to different paths in the data plane (see [I-D.dekater-scion-dataplane] section "Handling Link Failures").

To achieve scalability, SCION partitions ASes into ISDs and in an ideal topology the inter-ISD core network should be kept to a moderate size. For more specific observations, we distinguish between intra-ISD and inter-ISD beaconing.

3.3.1. Intra-ISD Beaconing

In the intra-ISD beaconing, PCBs are propagated top down along parent-child links from core to leaf ASes. Each AS discovers path segments from itself to the core ASes of its ISD.

This typically produces an acyclic graph which is narrow at the top, widens towards the leafs, and is relatively shallow. Intermediate provider ASes will have a large number of children, while they only have a small number of parents and the chain of intermediate providers from a leaf AS to a core AS is typically not long (e.g. local, regional, national provider, then core).

Each AS potentially receives PCBs for all down path segments from the core to itself. While the number of distinct provider chains to the core is typically moderate, the multiplicity of links between provider ASes has multiplicative effect on the number of PCBs. Once this number grows above the maximum recommended best PCBs set size of 50, ASes SHOULD trim the set of PCBs propagated.

Ultimately, the number of PCBs received by an AS per propagation interval remains bounded by 50 for each parent link of an AS, and at most 50 PCBs per child link are propagated. The length of these PCBs and thus the number of AS entries to be processed and stored, is expected to be moderate and not grow considerably with network size. The total resource overhead for beacon propagation is easily manageable even for highly connected ASes.

To illustrate this, an AS with a rather large number of 100 parent links receives at most 5000 PCBs during a propagation interval. Assuming a generous average length of 10 AS entries for these PCBs, this corresponds to 50000 AS entries.

Due to the variable length fields in AS entries, the sizes for storage and transmission cannot be predicted exactly, but assume an average of 250 bytes per AS entry. At the shortest recommended propagation interval of 5 seconds, this corresponds to an average bandwidth of around 2.5 MB/s and the processing of 10000 signature verifications per second.

If the same AS has 1000 child links, the propagation of the beacons will require signing one new AS entry for each of the propagated PCBs for each link (at most 50 per link), i.e. at most 50000 signatures per propagation event. The total bandwidth for the propagation of these PCBs for all 1000 child links would, be roughly around 25 MB/s which is manageable with even modest consumer hardware.

On a network bootstrap, path segments to each AS are discovered within a number of propagation steps proportional to the longest path. With a 5 second propagation interval and a generous longest path of length 10, all path segments are discovered after 25 seconds on average. When all ASes start propagation just after they’ve received the first PCBs from any of their upstreams (see ’fast recovery’), the construction of a first path to connect each AS to the ISD core is accelerated.

When a new parent-child link is added to the network, the parent AS will propagate the available PCBs in the next propagation event. If the AS on the child side of the new link is a leaf AS, path discovery is thus complete after at most one propagation interval. Otherwise, child ASes at distance D below the new link, learn of the new link after at worst D further propagation intervals.

3.3.2. Inter-ISD Beaconing

In the inter-ISD core beaconing, PCBs are propagated omnidirectionally along core links. Each AS discovers path segments from itself to any other core AS.

The number of distinct paths through the core network is typically very large. To keep the overhead manageable, at most 5 path segments to every destination AS are discovered and the propagation frequency is slower than in the intra-ISD beaconing (at least 60 seconds between propagation events).

Without making strong assumptions on the topology of the core network, we can assume that shortest paths through real world networks are relatively short, e.g. the Barabsi-Albert random graph model predicts a diameter of log(N)/log(log(N)) for a network with N nodes [BollRio-2000] and the average distance scales in the same way. Whilst we cannot assume that the selected PCBs are strictly the shortest paths through the network, they are likely to be not very much longer than the shortest paths either.

With N the number of participating core ASes, an AS receives up to 5 * N PCBs per propagation interval per core link interface. For highly connected ASes, the number of PCBs received thus becomes rather large and in a network of 1000 ASes, a AS with 300 core links receives up to 1.5 million PCBs per propagation interval.

Assuming an average PCB length of 6 and the shortest propagation interval of 60 seconds, this corresponds to roughly 150 thousand signature validations per second or roughly 38 MB/s. For much larger, more highly connected ASes, the path discovery tasks of the Control Service can be distributed over many instances in order to increase the PCB throughput.

On a network bootstrap, full connectivity is obtained after a number of propagation steps corresponding to the diameter of the network. Assuming a network diameter of 6, this corresponds to roughly 3 minutes on average. When a new link is added to the network, it will be available to connect two ASes at distances D1 and D2 from the link, respectively, at worst after a mean time (D1+D2)*T/2.

4. Registration of Path Segments

Path registration is the process where an AS transforms selected PCBs into path segments, and adding these segments to the relevant path databases thereby making them available to other ASes.

As mentioned previously, a non-core AS typically receives several PCBs representing several path segments to the core ASes of the ISD the AS belongs to. Out of these PCBs, the non-core AS selects those down path segments through which it wants to be reached, based on AS-specific selection criteria.

The next step is to register the selected down segments with the Control Service of the relevant core ASes in accordance with a process called _intra-ISD path segment registration_. As a result, a core AS’s Control Service contains all intra-ISD path segments registered by the non-core ASes of its ISD. In addition, each core AS Control Service also stores the preferred core path segments to other core ASes during the _core segment registration_ process.

Both processes are described below.

4.1. Intra-ISD Path Segment Registration

Every _registration period_ (determined by each AS), the AS's Control Service selects two sets of PCBs to transform into two types of path segments:

* Up segments, which allow the infrastructure entities and endpoints in this AS to communicate with core ASes; and

* Down segments, which allow remote entities to reach this AS.

The up segments and down segments do not have to be equal as AS may want to communicate with core ASes via one or more up segments that differ from the down segment(s) through which it wants to be reached. Therefore, an AS can define different selection policies for the up segment and down segment sets. In addition, the processes of transforming a PCB in an up segment or a down segment differ slightly.

4.1.1. Terminating a PCB

Both the up segments and down segments end at the AS, so by transforming a PCB into a path segment, an AS "terminates" the PCB for this AS ingress interface and at that moment in time.

The Control Service of a non-core performs the following steps to "terminate" a PCB:

1. The Control Service adds a new AS entry to the PCB which MUST be defined as follows:

    * The next AS MUST NOT be specified.

        - In Protobuf message format, this means that the value of the next_isd_as field in the ASEntrySignedBody component MUST be "0".

    * The egress interface in the Hop Field component MUST NOT be specified.

        - In Protobuf message format, this means that the value of the egress field in the HopField component MUST be "0".

2. If the AS has peering links, the Control Service MAY add corresponding peer entry components to the signed body of the AS entry - one peer entry component for each peering link that the AS wants to advertise. The egress Interface ID in the Hop Field component of each added peer entry MUST NOT be specified.

   * In Protobuf message format, this means that the value of the egress field in the HopField component MUST be "0".

3. The Control Service MUST sign the modified PCB and append the computed signature.

Note:

* For more information on the signed body component of an AS entry, see Section 2.2.1.4.2.

* For more information on a peer entry, see Section 2.2.1.7.

* For more information on the Hop Field component, see Section 2.2.1.6.

* For more information on signing an AS entry, see Section 2.2.1.4.3.

4.1.2. Transforming a PCB into an Up Segment

Every registration period, the Control Service of a non-core AS performs the following steps to transform PCBs into up segments:

1. The Control Service selects the PCBs that it wants to transform into up segments from the candidate PCBs in the Beacon Store.

2. The Control Service "terminates" the selected PCBs by performing the steps described in Section 4.1.1. From this moment on, the modified PCBs are called *up segments*.

3. The Control Service adds the newly created up segments to its own path database.

Note: For more information on possible selection strategies of PCBs, see Section 2.3.3.

4.1.3. Transforming a PCB into a Down Segment

Every registration period, the Control Service of a non-core AS performs the following steps to transform PCBs into down segments:

1. The Control Service selects the PCBs that it wants to transform into down segments from the candidate PCBs in the Beacon Store.

2. The Control Service "terminates" the selected PCBs by performing the steps described in Section 4.1.1. From this moment on, the modified PCBs are called *down segments*.

3. The Control Service registers the newly created down segments with the Control Services of the core ASes that originated the corresponding PCBs. This is done by invoking the SegmentRegistrationService.SegmentsRegistration remote procedure call (RPC) in the Control Services of the relevant core ASes (see also Section 4.3). The first ISD-AS entry of the path segment MUST be equal to the core ISD-AS where the segment is being registered. If not, the core AS MUST reject the segment.

Note: For more information on possible selection strategies of PCBs, see Section 2.3.3.

4.2. Core Path Segment Registration

The core beaconing process creates path segments from core AS to core AS. These core segments are then added to the Control Service path database of the core AS that created the segment, so that local and remote endpoints can obtain and use these core segments. In contrast to the intra-ISD registration procedure, there is no need to register core segments with other core ASes as each core AS will receive PCBs originated from every other core AS.

In every registration period, the Control Service of a core AS performs the following operations:

1. The core Control Service selects the best PCBs towards each core AS observed so far.

2. The core Control Service "terminates" the selected PCBs by performing the steps described in Section 4.1.1. From this moment on, the modified PCBs are called *core segments*.

3. The Control Service adds the newly created core segments to its own path database.

Note: For more information on possible selection strategies of PCBs, see Section 2.3.3.

4.3. Path Segment Registration gRPC API

The Control Service of a non-core AS has to register the newly created down segments with the Control Services of the core ASes that originated the corresponding PCBs. This registration step is implemented as follows in Protobuf message format:

```
enum SegmentType {
    SEGMENT_TYPE_UNSPECIFIED = 0;
    SEGMENT_TYPE_UP = 1;
    SEGMENT_TYPE_DOWN = 2;
    SEGMENT_TYPE_CORE = 3;
}

service SegmentRegistrationService {
    rpc SegmentsRegistration(SegmentsRegistrationRequest) returns (
      SegmentsRegistrationResponse) {}
}

message SegmentsRegistrationRequest {
    message Segments {
        repeated PathSegment segments = 1;
    }

    map<int32, Segments> segments = 1;
}

message SegmentsRegistrationResponse {}
```

* SegmentType: Specifies the type of the path segment to be registered. Currently, only the following type is used:
  - SEGMENT_TYPE_DOWN: Specifies a down segment.
* map<int32, Segments> segments: Represents a separate list of segments for each path segment type. The key is the integer representation of the corresponding SegmentType.
* SegmentRegistrationResponse: an empty message returned as an acknowledgement upon success.

4.4. Path MTU

SCION paths represent a sequence of ASes and inter-AS links; each with possibly different MTUs. As a result, the path MTU is the minimum of the MTUs of each inter-AS link and intra-AS networks it traverses. Such MTU information is disseminated during path construction:

* The MTU of each intra-AS network traversed (represented by the MTU field of the corresponding AS Entries (Section 2.2.1.4.2))

* The MTU of each inter-AS link or peering link (indicated by the ingress_mtu field of each Section 2.2.1.5 or the peer_mtu field of each Section 2.2.1.7 used)

Such information is then made available to endpoints during the path lookup process (See Section 5). SCION endpoints are oblivious to the topology of intermediate ASes and when looking up a path they assume that all hops are constrained by the intra-AS MTU of each AS traversed.

5. Path Lookup

The _path lookup_ is a fundamental building block of SCION's path management as it enables endpoints to obtain path segments found during path exploration and registered during path registration. This allows the endpoints to construct end-to-end paths from the set of possible path segments returned by the path lookup process. The lookup of paths still happens in the control plane, whereas the construction of the actual end-to-end paths happens in the data plane.

5.1. Lookup Process

An endpoint (source) that wants to start communication with another endpoint (destination) requires up to three path segments:

* An up segment to reach the core of the source ISD (only if the source endpoint is a non-core AS);

* a core segment to reach

  - another core AS in the source ISD, in case the destination AS is in the same source ISD, or;

  - a core AS in a remote ISD, if the destination AS is in another ISD, and;

* a down segment to reach the destination AS.

The actual number of required path segments depends on the location of the destination AS as well as on the availability of shortcuts and peering links. More information on combining and constructing paths is provided by [I-D.dekater-scion-dataplane].

The process to look up and fetch path segments consists of the following steps:

1. The source endpoint queries the Control Service in its own AS (i.e. the source AS) for the required segments by sending a SegmentsRequest. The Control Service has up segments stored in its path database and additionally checks if it has appropriate core segments and down segments stored as well - in this case it returns them immediately in a SegmentsResponse.

2. If there are no appropriate core segments and down segments, the Control Service in the source AS queries the Control Services of the reachable core ASes in the source ISD for core segments to core ASes in the destination ISD. To reach the core Control Services, the Control Service of the source AS uses the locally stored up segments.

3. The Control Service of the source AS combines up segments with the newly retrieved core segments. The Control Service then queries the Control Services of the remote core ASes in the destination ISD to fetch down segments to the destination AS. To reach the remote core ASes, the Control Service of the source AS uses the previously obtained and combined up segments and core segments.

4. The Control Service of the source AS returns all retrieved path segments to the source endpoint.

5. Once it has obtained all path segments, the source endpoint combines them into an end-to-end path in the data plane.

6. The destination endpoint, once it receives the first packet, MAY revert the path in the received packet in order to construct a response. This ensures that traffic flows on the same path bidirectionally.

Table 3 below shows which Control Service provides the source endpoint with which type of path segment.

Segment Type	Responsible Control Service(s)
Up-segment	Control service of the source AS
Core-segment	Control service of core ASes in source ISD
Down-segment	Control service of core ASes in destination ISD (either the local ISD or a remote ISD)

Table 3: Control services responsible for different types of path segments

5.1.1. Sequence of Lookup Requests

The overall sequence of requests to resolve a path SHOULD be as follows:

1. Request up segments for the source endpoint at the Control Service of the source AS.

2. Request core segments, which start at the core ASes that are reachable with up segments, and end at the core ASes in the destination ISD. If the destination ISD coincides with the source ISD, this step requests core segments to core ASes that the source endpoint cannot directly reach with an up segment.

3. Request down segments starting at core ASes in the destination ISD.

The segment lookup API gRPC definition can be found in Figure 12.

5.1.2. Caching

For the sake of efficiency, the Control Service of the source AS SHOULD cache each returned path segment request. Caching ensures that path lookups are fast for frequently used destinations and is also essential to ensure that the path lookup process is scalable and can be performed with low latency.

In general, to improve overall efficiency, the Control Services of all ASes SHOULD do the following:

* Cache the returned path segments.

* Send requests in parallel when requesting path segments from other Control Services.

5.2. Behavior of Actors in the Lookup Process

As described above, the source endpoint resolves paths with a sequence of segment requests to the Control Service of the source AS. The Control Service in the source AS either answers directly or forwards these requests to the responsible Control Services of core ASes. In SCION, the instances that handle these segment requests at the Control Services are called _source AS segment-request handler_ and _core AS segment-request handler_, respectively.

This section specifies the behavior of the segment request handlers in the lookup process.

5.2.1. Use of Wildcard Addresses in the Lookup Process

Endpoints can use wildcard addresses to designate any core AS in path segment requests. The segment request handlers MUST expand these wildcard addresses and translate them into one or more actual addresses. Table 4 below shows who is responsible for what.

Note: For general information on the use of wildcard addresses in SCION, see Section 1.5.2.1.

Segment Request	Wildcard Represents	Expanded/ Translated By	Translated Into
Up segment	"Destination" core AS (where up segment ends)	Control service of the _source AS_	Actual address destination core AS in source ISD
Core segment	Source core AS (where core segment starts)^1	Control service of the _source AS_	Actual address source core AS in source ISD
Core segment	Destination core AS (where core segment ends)	Control service of the _source core AS_	Actual address destination core AS in destination ISD
Down segment	"Source" core AS (where down segment starts)^2	Control service of the _source AS_	Actual address source core AS in destination ISD

Table 4: Use of wildcards in path segments requests

1) Includes all core ASes for which an up segment from the source AS exists.
2) Includes all core ASes in destination ISD with a down segment to destination AS.

5.2.2. Segment-Request Handler of a Non-Core Source AS

When the segment request handler of the Control Service of a _non-core_ source AS receives a path segment request, it MUST proceed as follows:

1. Determine the requested segment type.

2. In the case of an up segment request, look up matching up segments in the path database and return them.

3. In the case of a core segment request from a source core AS to a destination core AS:

    * Expand the source wildcard into separate requests for each reachable core AS in the source ISD.

    * For each core segment request;

        - If possible, return matching core segments from cache;

        - Otherwise, request the core segments from the Control Services of each reachable core AS at the source of the core segment, and then add the retrieved core segments to the cache.

4. In the case of a down segment request:

    * Expand the source wildcard into separate requests for every core AS in the destination ISD (destination ISD refers to the ISD to which the destination endpoint belongs).

    * For each segment request;

        - If possible, return matching down segments from cache;

        - Otherwise, request the down segment from the Control Services of the core ASes at the source of the down segment. Sending the request may require looking up core segments to the source core AS of the down segment, and then adding the retrieved down segments to the cache.

5.2.3. Segment-Request Handler of a Core AS

When the segment request handler of a _core AS_ Control Service receives a path segment request, it MUST proceed as follows:

1. Validate the request:
    * The source of the path segment MUST be this core AS.
    * The request MUST either be;
        - for a core segment to a core AS in this ISD or another ISD, or;
        - for a down segment to an AS in this ISD.
2. If the destination is a core or wildcard address, then load matching core segments from the path database and return.
3. Otherwise, load the matching down segments from the path database and return.

Appendix "Path-Lookup Examples" shows by means of an illustration how the lookup of path segments in SCION works.

6. SCMP

The SCION Control Message Protocol (SCMP) provides functionality for network diagnostics, such as traceroute, and error messages that signal packet processing or network-layer problems. SCMP is a helpful tool for network diagnostics and, in the case of External Interface Down and Internal Connectivity Down messages, a signal for endpoints to detect network failures more rapidly and fail-over to different paths. However, SCION nodes should not strictly rely on the availability of SCMP, as this protocol may not be supported by all devices and/or may be subject to rate limiting.

This document only specifies the messages used for the purposes of path diagnosis and recovery. An extended specification, still a work in progress, can be found in [SCMP].

6.1. General Format

Every SCMP message is preceded by a SCION header and zero or more SCION extension headers (see [I-D.dekater-scion-dataplane] section "SCION Header Specification"). The SCMP header is identified by a NextHdr value of 202 in the immediately preceding header.

The messages have the following general format:

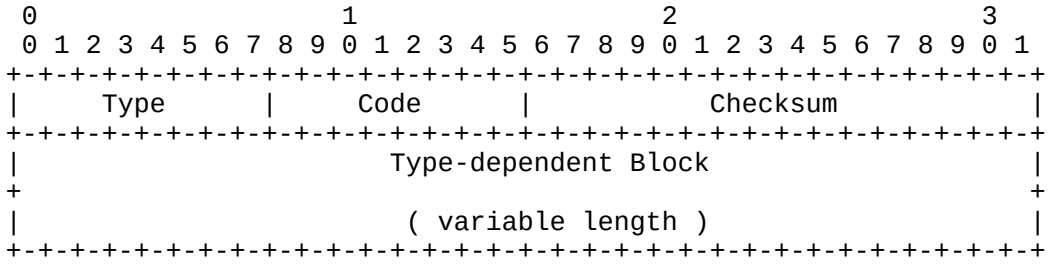


Figure 17: SCMP message format

* Type: it indicates the type of SCMP message. Its value determines the format of the type-dependent block.

* Code: it provides additional granularity to the SCMP type.

* Checksum: it is used to detect data corruption.

* Type-dependent Block: optional field of variable length which format is dependent on the message type.

6.2. Message Types

SCMP messages are grouped into two classes: error messages and informational messages. Error messages are identified by a zero in the high-order bit of the type value. I.e., error messages have a type value in the range of 0-127. Informational messages have type values in the range of 128-255.

This specification defines the message formats for the following SCMP messages:

Type	Meaning
1	Reserved for future use
2	Packet Too Big (Section 6.5.1)
3	Reserved for future use
4	Reserved for future use
5	External Interface Down (Section 6.5.2)
6	Internal Connectivity Down (Section 6.5.3)
100	Private Experimentation
101	Private Experimentation
127	Reserved for expansion of SCMP error messages

Table 5: Error Message Types

Type	Meaning
128	Echo Request (Section 6.6.1)
129	Echo Reply (Section 6.6.2)
130	Traceroute Request (Section 6.6.3)
131	Traceroute Reply (Section 6.6.4)
200	Private Experimentation
201	Private Experimentation
255	Reserved for expansion of SCMP informational messages

Table 6: Informational Message Types

Type values 100, 101, 200, and 201 are reserved for private experimentation.

All other values are reserved for future use.

6.3. Checksum Calculation

The checksum is the 16-bit one’s complement of the one’s complement sum of the entire SCMP message, starting with the SCMP message type field, and prepended with a "pseudo-header" consisting of the SCION address header and the Layer 4 protocol type as defined in [I-D.dekater-scion-dataplane] section "SCION Header Specification/ Pseudo Header for Upper-Layer Checksum".

6.4. Processing Rules

The following rules apply when processing SCMP messages:

* If an SCMP error message of unknown type is received at its destination, it MUST be passed to the upper-layer process that originated the packet that caused the error, if it can be identified.

* If an SCMP informational message of unknown type is received, it MUST be silently dropped.

* Every SCMP error message MUST include as much of the offending SCION packet as possible. The error message packet - including the SCION header and all extension headers - MUST NOT exceed *1232 bytes* in order to fit into the minimum MTU (see [I-D.dekater-scion-dataplane] section "Deployment Considerations/ MTU").

* In case the implementation is required to pass an SCMP error message to the upper-layer process, the upper-layer protocol type is extracted from the original packet in the body of the SCMP error message and used to select the appropriate process to handle the error. In case the upper-layer protocol type cannot be extracted from the SCMP error message body, the SCMP message MUST be silently dropped.

* An SCMP error message MUST NOT be originated in response to any of the following:

  - An SCMP error message.

  - A packet which source address does not uniquely identify a single node. E.g., an IPv4 or IPv6 multicast address.

The maximum size 1232 bytes is chosen so that the entire datagram, if encapsulated in UDP and IPv6, does not exceed 1280 bytes (L2 Header excluded). 1280 bytes is the minimum MTU required by IPv6 and it is assumed that, nowadays, this MTU can also be safely expected when using IPv4.

6.5. Error Messages

6.5.1. Packet Too Big

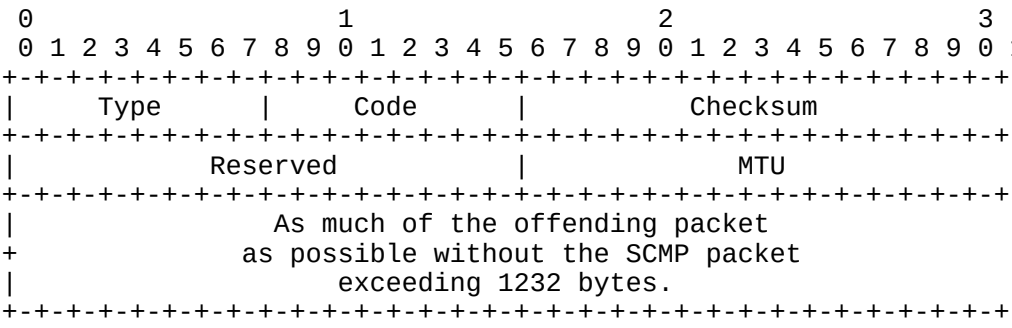


Figure 18: Packet Too Big format

Name	Value
Type	2
Code	0
MTU	The Maximum Transmission Unit of the next-hop link.

Table 7: Error Message field values

A *Packet Too Big* message SHOULD be originated by a router in response to a packet that cannot be forwarded because the packet is larger than the MTU of the outgoing link. The MTU value is set to the maximum size a SCION packet can have to still fit on the next-hop link, as the sender has no knowledge of the underlay.

6.5.2. External Interface Down

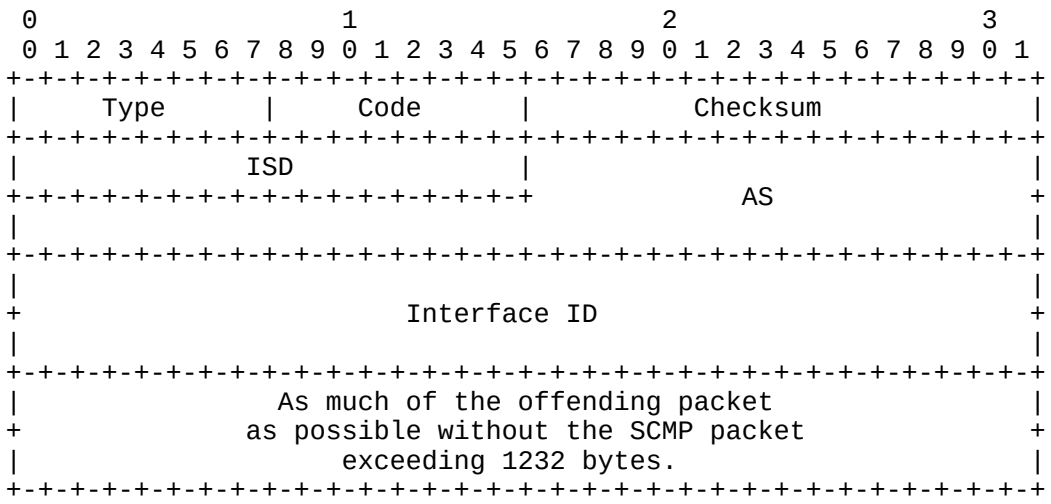


Figure 19: External Interface Down format

Name	Value
Type	5
Code	0
ISD	The 16-bit ISD identifier of the SCMP originator
AS	The 48-bit AS identifier of the SCMP originator
Interface ID	The interface ID of the external link with connectivity issue.

Table 8: Error Message field values

A *External Interface Down* message SHOULD be originated by a router in response to a packet that cannot be forwarded because the link to an external AS is broken. The ISD and AS identifier are set to the ISD-AS of the originating router. The Interface ID identifies the link of the originating AS that is down.

Recipients can use this information to route around broken data-plane links.

6.5.3. Internal Connectivity Down

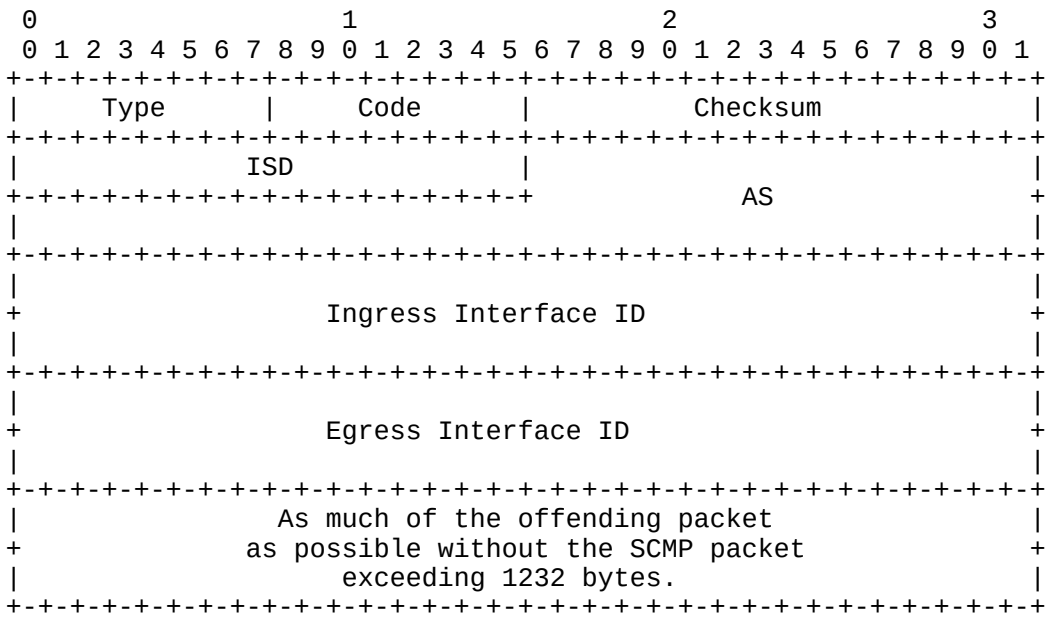


Figure 20: Internal Connectivity Down format

Name	Value
Type	6
Code	0
ISD	The 16-bit ISD identifier of the SCMP originator
AS	The 48-bit AS identifier of the SCMP originator
Ingress ID	The interface ID of the ingress link.
Egress ID	The interface ID of the egress link.

Table 9: Error Message field values

A *Internal Connectivity Down* message SHOULD be originated by a router in response to a packet that cannot be forwarded inside the AS because the connectivity between the ingress and egress routers is broken. The ISD and AS identifier are set to the ISD-AS of the originating router. The ingress Interface ID identifies the interface on which the packet enters the AS. The egress Interface ID identifies the interface on which the packet is destined to leave the AS, but the connection is broken to.

Recipients can use this information to route around a broken data plane inside an AS.

6.6. Informational Messages

6.6.1. Echo Request

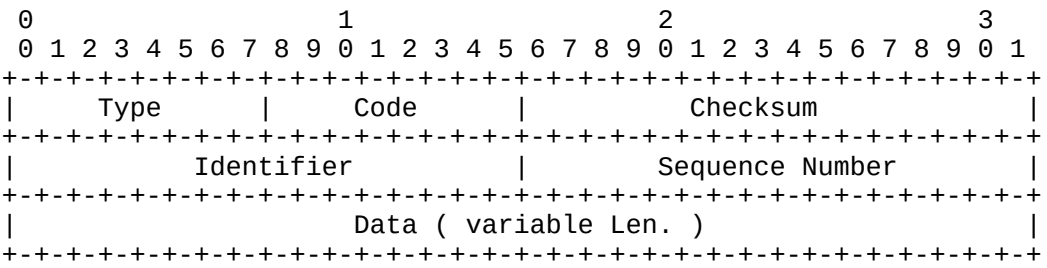


Figure 21: Echo Request format

Name	Value
Type	128
Code	0
Identifier	A 16-bit identifier to aid matching replies with requests
Sequence Nr.	A 16-bit sequence number to aid matching replies with requests
Data	Variable length of arbitrary data

Table 10: Informational Message field values

Every node SHOULD implement a SCMP Echo responder function that receives Echo Requests and originates corresponding Echo replies.

6.6.2. Echo Reply

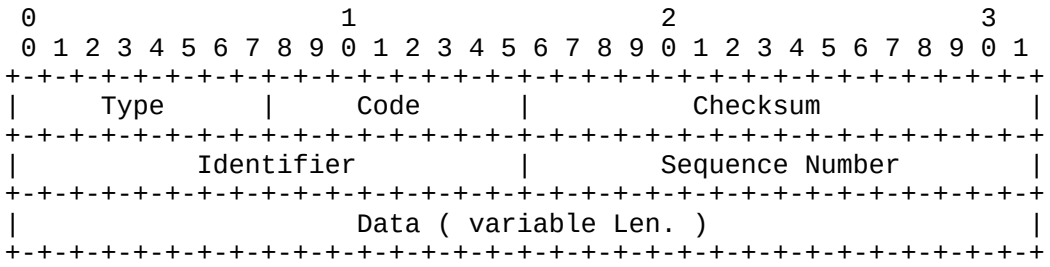


Figure 22: Echo Reply format

Name	Value
Type	129
Code	0
Identifier	The identifier of the Echo Request
Sequence Nr.	The sequence number of the Echo Request
Data	The data of the Echo Request

Table 11: Informational Message field values

Every node SHOULD implement a SCMP Echo responder function that receives Echo Requests and originates corresponding Echo replies.

The data received in the SCMP Echo Request message MUST be returned entirely and unmodified in the SCMP Echo Reply message.

6.6.3. Traceroute Request

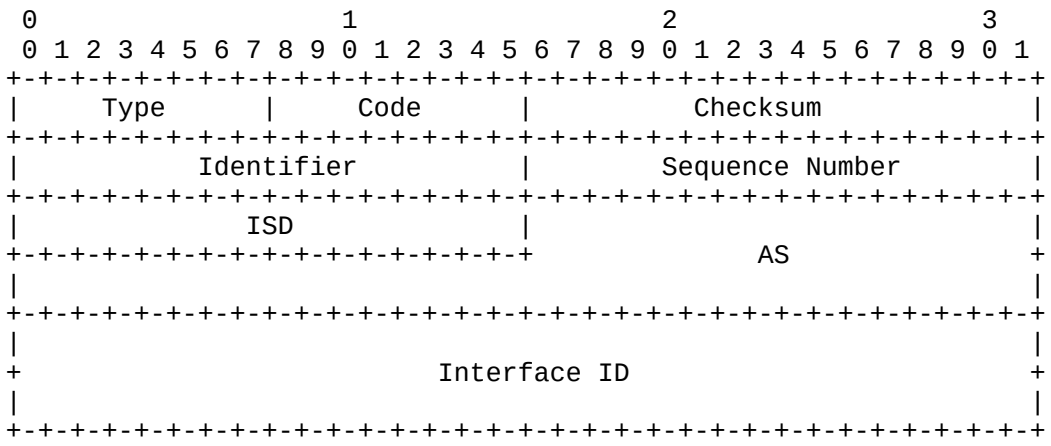


Figure 23: Traceroute Request format

Given a SCION path constituted of hop fields, traceroute allows to identify the corresponding on-path ISD-ASes.

Name	Value
Type	130
Code	0
Identifier	A 16-bit identifier to aid matching replies with requests
Sequence Nr.	A 16-bit sequence number to aid matching replies with request
ISD	Place holder set to zero by SCMP sender
AS	Place holder set to zero by SCMP sender
Interface ID	Place holder set to zero by SCMP sender

Table 12: Informational Message field values

A border router is alerted of a Traceroute Request message through the Ingress or Egress Router Alert flag set to 1 in the hop field that describes the traversal of that router in a packet's path (see [I-D.dekater-scion-dataplane] section "SCION Header Specification"). When such a packet is received, the border router SHOULD reply with a Traceroute Reply message (Section 6.6.4).

6.6.4. Traceroute Reply

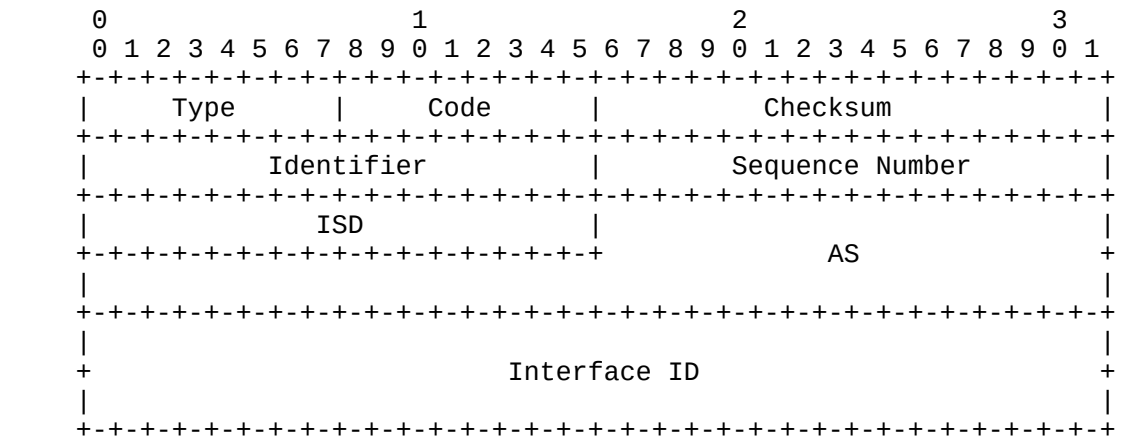


Figure 24: Traceroute Reply format

Name	Value
Type	131
Code	0
Identifier	The identifier set in the Traceroute Request
Sequence Nr.	The sequence number of the Tracroute Request
ISD	The 16-bit ISD identifier of the SCMP originator
AS	The 48-bit AS identifier of the SCMP originator
Interface ID	The interface ID of the SCMP originating router

Table 13: Informational Message field values

The identifier is set to the identifier value from the Traceroute Request message (Section 6.6.3). The ISD and AS identifiers are set to the ISD-AS of the originating border router.

6.7. SCMP Authentication

Authentication of SCMP packets is not specified here. It is currently still experimental so endpoints should validate link down messages (External Interface Down (Section 6.5.2) and Internal Connectivity Down (Section 6.5.3)) with additional signals for reliable operations. These additional signals are outside the scope of this specification.

7. Security Considerations

As described previously, the goal of SCION's beaconing process in the control plane is to securely discover and disseminate paths between any two ASes. This section describes security considerations for SCION's Control Plane that focuses on _inter_-domain routing. SCION does not provide intra-domain routing, nor does it provide end-to-end payload encryption so these topics lie outside the scope of this section.

This section discusses three kinds of threats to the control plane:

1. When an adversary controls one or all core ASes of an ISD and tries to manipulate the beaconing process from the top down (see Section 7.2).

2. When "ordinary" (non-core) adversaries try to manipulate the beaconing process (see Section 7.3).

3. Denial of Services (DoS) attacks where attackers overload different parts of the infrastructure (see Section 7.4).

7.1. Security Properties

The SCION control plane provides various security properties, as discussed below. Here, an AS is described as 'honest' if its private keys are unknown to the attacker and if it uses a unique interface identifier for each link. An honest path is one that only traverses honest ASes. A honest segment is the one created by an honest AS.

Security properties are:

* Connectivity - For every pair of honest ASes X and Y, X will eventually register enough segments to build at least one path (of any length) leading to Y.

* Forwarding Path Consistency - For every honest path segment registered in any AS

  - its sequence of AS entries corresponds to a continuous SCION forwarding path in the network of inter-domain links

  - the inter-domain network topology remains unchanged since the segment was first generated.

* Loop Freedom - For every honest path segment registered in any AS, its sequence of AS entries contains no duplicates, including current and next ISD-AS and Interface IDs.

* Path Authorization - For every honest path segment registered in any AS and any AS X appearing on that segment (except for the previous one), AS X propagated a PCB corresponding to the segment portion ending in its own entry to its successor AS on the segment.

To ensure that the properties hold across the overall SCION network, all core ASes should be able to reach each other with some sequence of core links, and all non-core ASes should have at least one path up to a core AS. Furthermore, to ensure that the properties hold within a single ISD, all cores ASes of the ISD should be able to reach each other without leaving the ISD, i.e., for every pair of cores in an ISD there is a sequence of SCION links that only traverses ISD members. A core AS may reach other core ASes in the same ISD via other ISDs. This may be permitted, depending on the ISD’s policies.

7.2. Manipulation of the Beaconing Process by a Core Adversary

The first risk to the beaconing process comes from an adversary controlling one or more core ASes in an ISD. If the adversary stops all core AS(es) within an ISD from propagating PCBs, the discovery of new paths will halt. In this case, downstream ASes will notice that PCBs are no longer being propagated, but all previously discovered and still valid paths remain usable for data plane forwarding until they expire. This is an unlikely scenario, as it would require compromise of all core ASes within an ISD.

7.3. Manipulation of the Beaconing Process by a Non-Core Adversary

This section examines several possible approaches that could be taken by an "ordinary" non-core adversary to manipulate the beaconing process in the Control Plane. For each case it shows to what extent SCION’s design can prevent the corresponding attack or help mitigate it.

7.3.1. Path Hijacking through Interposition

A malicious AS M might try to manipulate the beaconing process between two neighbor ASes A and B, with the goal to hijack traffic to flow via M. If M can interpose itself on the path between A and B, then it could attempt several potential attacks:

* The adversary M could intercept and disseminate a PCB on its way from A to the neighboring AS B, and inject its own AS entry into the PCB toward downstream ASes.

* The adversary could modify the Hop Fields of an already existing path in order to insert its own AS in the path.

* The adversary could fully block traffic between AS A and AS B in order to force traffic redirection through an alternate path that includes its own AS.

The first type of attack is detectable and blocked by downstream ASes (e.g. B) because a PCB disseminated by AS A towards AS B contains the "Next ISD AS" field in the entry of AS A, pointing to AS B, and protected by A’s signature. If M manipulates the PCB while in flight from A to B, then verification of the manipulated inbound PCBs will fail at AS B, as the adversary’s PCBs cannot contain A’s correct signature (see Section 2.3.1).

The second type of attack is made impossible by the Hop Field’s MAC which protects the Hop Field’s integrity and chains it with the previous Hop Fields on the path.

The third type of attack generally cannot be prevented. However the alternate path would be immediately visible to endpoints as traffic MUST include Hop Fields from AS M.

7.3.2. Creation of Spurious ASes and ISDs

An alternative scenario is when an adversary tries to introduce and spoof a non-existent AS. This would enable the adversary to send traffic with the spoofed AS as a source, allowing the adversary to complicate the detection of its attack and to plausibly deny the misbehavior.

However, spoofing a new AS requires a registration of that AS with the ISD core to obtain a valid AS certificate, otherwise the adversary cannot construct valid PCBs. As this registration should include a thorough check and authentication by a CA, this cannot be done stealthily which defeats the original purpose.

Similarly to creating a fake AS, an adversary could try to introduce a new malicious ISD. This involves the generation of its own TRC, finding core ASes to peer with, and convincing other ISDs of its legitimacy to accept the new TRC. Although this setup is not entirely impossible, it requires substantial time and effort and may need the involvement of more than one malicious entity. Here the "costs" of setting up the fake ISD may outweigh the benefits.

7.3.3. Peering Link Misuse

The misuse of a peering link by an adversary represents another type of attack. Consider the case where AS A wants to share its peering link only with one of its downstream neighbors AS B, and therefore selectively includes the peering link only in PCBs sent to B. An adversary may now try to gain access to this peering link by prepending the relevant PCBs to its own path. For this, the adversary needs to be able to (1) eavesdrop on the link from A to B, and (2) obtain the necessary Hop Fields by querying a Control Service and extracting the Hop Fields from registered paths.

Even if an adversary succeeds in misusing a peering link as described above, SCION is able to mitigate this kind of attack. Each AS includes an egress interface as well as specific “next hop” information to the PCB before disseminating it further downstream. If a malicious entity tries to misuse a stolen PCB by adding it to its own segments, verification will fail upstream as the egress interface mismatches. Therefore, the peering link can only be used by the intended AS.

7.3.4. Manipulation of the Path-Selection Process

SCION endpoints select inter-domain forwarding paths. This section discusses some mechanisms with which an adversary can attempt to trick endpoints downstream (in the direction of beaconing) into choosing non-optimal paths. The goal of such attacks is to make paths that are controlled by the adversary more attractive than other available paths.

In SCION, overall path selection is the result of three steps. Firstly, each AS selects which PCBs are further forwarded to its neighbors. Secondly, each AS chooses the paths it wants to register at the local Control Service (as up segments) and at the core Control Service (as down segments). Thirdly, the endpoint performs path selection among all available paths resulting from a path lookup process.

These attacks are only successful if the adversary is located within the same ISD and upstream relative to the victim AS. It is not possible to attract traffic away from the core as traffic travels upstream towards the core. Furthermore, the attack may either be discovered downstream (e.g., by seeing large numbers of paths becoming available), or during path registrations. After detection, non-core ASes will be able to identify paths traversing the adversary AS and avoid these paths.

Announcing Large Numbers of Path Segments
This attack is possible if the adversary controls at least two ASes. The adversary can create a large number of links between the ASes under its control which do not necessarily correspond to physical links. This allows the adversary to multiply the number of PCBs forwarded to its downstream neighbor ASes and in turn increases the chance that one or several of these forwarded PCBs are selected by the downstream ASes.

In general, the number of PCBs that an adversary can announce this way scales exponentially with the number of consecutive ASes the adversary controls. However, this also decreases their chance of being chosen by a downstream AS for PCB dissemination or by an endpoint for path construction as these relatively long paths have to compete with other shorter paths. Furthermore, both endpoints and downstream ASes can detect poorer quality paths in the data plane and switch to better paths.

Wormhole Attack
A malicious AS M1 can send a PCB not only to their downstream neighbor ASes, but also out-of-band to another non-neighbor colluding malicious AS M2. This creates new segments to M2 and M2’s downstream neighbor ASes, simulating a link between M1 and M2 which may not correspond to an actual link in the network topology.

Similarly, a fake path can be announced through a fake peering link between two colluding ASes even if in different ISDs. An adversary can advertise fake peering links between the two colluding ASes, thus offering short paths to many destination ASes. Downstream ASes might have a policy of preferring paths with many peering links and thus are more likely to disseminate PCBs from the adversary. Endpoints are also more likely to choose short paths that make use of peering links. In the data plane, whenever the adversary receives a packet containing a fake peering link, it can transparently exchange the fake peering Hop Fields with valid Hop Fields to the colluding AS. To avoid detection of the path alteration by the receiver, the colluding AS can replace the added Hop Fields with the fake peering link Hop Fields the sender inserted.

To defend against this attack, methods to detect the wormhole attack are needed. Per link or path latency measurements can help reveal the wormhole and render the fake peering link suspicious or unattractive. Without specific detection mechanisms these so-called wormhole attacks are unavoidable in routing.

7.4. Denial of Service Attacks

The beaconing process in the SCION Control Plane relies on control plane communication. ASes exchange control plane messages within each other when propagating PCBs to downstream neighbors, when registering PCBs as path segments, or during core path lookup. Volumetric DoS attacks, where attackers overload a link may make it difficult to exchange these messages.

SCION limits the impact of such attacks which aim to exhaust network bandwidth on links as ASes can switch to alternative paths that do not contain the congested links. Reflection-based attacks are also prevented as response packets are returned on the same path to the actual sender.

Other mechanisms are required to avoid transport protocol attacks where the attacker tries to exhaust the resources on a target server, such as for the Control Services, by opening many connections to this. The means to mitigate these kind of DoS attacks are basically the same as for the current Internet, e.g. filtering, geo-blocking or using cookies.

Thanks to its path awareness, SCION enables more fine-grained filtering mechanisms based on certain path properties. For example, control plane RPC methods that are available to endpoints within an AS are strictly separate from methods available to endpoints from other ASes. Specifically, expensive recursive path segment and trust material lookups are thus shielded from abuse by unauthorized entities.

For RPC methods exposed to other ASes, the Control Service implementation minimizes its attack surface by rejecting illegitimate callers based on ISD/AS, path type and length and any other available data points as soon as possible, i.e. immediately after determining the request type. For example:

* SegmentCreationService.Beacon can only be called by direct neighbors and thus calls from peers with a path length greater than one can immediately be discarded.

* SegmentRegistrationService.SegmentsRegistration can only be called from within the same ISD, thus the source address MUST match the local ISD and the number of path segments MUST be 1.

A combination of the mechanism above is used to prevent flooding attacks on the Control Service. In addition, the Control Service SHOULD be deployed in a distributed and replicated manner so that requests can be balanced and a single instance failure does not result in a complete failure of the control plane of a SCION AS.

8. IANA Considerations

This document has no IANA actions.

The ISD and SCION AS number are SCION-specific numbers. They are currently allocated by Anapaya Systems, a provider of SCION-based networking software and solutions (see [ISD-AS-assignments-Anapaya]). This task is being transitioned from Anapaya to the SCION Association (see [ISD-AS-assignments]).

9. References

9.1. Normative References

[Connect] "Connect Protocol Reference", 2024, <https://connectrpc.com/docs/protocol/>.

[gRPC] "gRPC, an open-source universal RPC framework", 2023, <https://grpc.io/>.

[I-D.dekater-scion-dataplane] de Kater, C., Rustignoli, N., Hugly, J., and S. Hitz, "SCION Data Plane", Work in Progress, Internet-Draft, draft-dekater-scion-dataplane-05, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-dekater-scion-dataplane-05>.

[I-D.dekater-scion-pki] de Kater, C., Rustignoli, N., and S. Hitz, "SCION Control Plane PKI", Work in Progress, Internet-Draft, draft-dekater-scion-pki-09, 3 March 2025, <https://datatracker.ietf.org/doc/html/draft-dekater-scion-pki-09>.

[proto3] "Protocol Buffers Language Guide version 3", 2023, <https://protobuf.dev/programming-guides/proto3/>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC4632] Fuller, V. and T. Li, "Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan", BCP 122, RFC 4632, DOI 10.17487/RFC4632, August 2006, <https://www.rfc-editor.org/rfc/rfc4632>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/rfc/rfc5280>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC9000] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

9.2. Informative References

[BollRio-2000] Bollobs, B. and O. Riordan, "The diameter of a scale-free random graph", n.d., <https://kam.mff.cuni.cz/~ksemweb/clanky/BollobasR-scale_free_random.pdf>.

[CHUAT22] Chuat, L., Legner, M., Basin, D., Hausheer, D., Hitz, S., Mueller, P., and A. Perrig, "The Complete Guide to SCION", ISBN 978-3-031-05287-3, 2022, <https://doi.org/10.1007/978-3-031-05288-0>.

[I-D.dekater-panrg-scion-overview] de Kater, C., Rustignoli, N., and A. Perrig, "SCION Overview", Work in Progress, Internet-Draft, draft-dekater-panrg-scion-overview-06, 7 May 2024, <https://datatracker.ietf.org/doc/html/draft-dekater-panrg-scion-overview-06>.

[ISD-AS-assignments]
"SCION Registry", 2025, <http://scion.org/registry/>.

[ISD-AS-assignments-Anapaya]
"SCION ISD and AS Assignments", 2025,
<https://docs.anapaya.net/en/latest/resources/isd-as-assignments/>.

[RFC1122] Braden, R., Ed., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, DOI 10.17487/RFC1122, October 1989, <https://www.rfc-editor.org/rfc/rfc1122>.

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/rfc/rfc4271>.

[RFC5398] Huston, G., "Autonomous System (AS) Number Reservation for Documentation Use", RFC 5398, DOI 10.17487/RFC5398, December 2008, <https://www.rfc-editor.org/rfc/rfc5398>.

[RFC6996] Mitchell, J., "Autonomous System (AS) Reservation for Private Use", BCP 6, RFC 6996, DOI 10.17487/RFC6996, July 2013, <https://www.rfc-editor.org/rfc/rfc6996>.

[RFC9217] Trammell, B., "Current Open Questions in Path-Aware Networking", RFC 9217, DOI 10.17487/RFC9217, March 2022, <https://www.rfc-editor.org/rfc/rfc9217>.

[RFC9473] Enghardt, R. and C. Krhenbhl, "A Vocabulary of Path Properties", RFC 9473, DOI 10.17487/RFC9473, September 2023, <https://www.rfc-editor.org/rfc/rfc9473>.

[SCIONLAB] Kown, J., Garca-Pardo, J., Legner, M., Wirz, F., Frei, M., Hausheer, D., and A. Perrig, "SCIONLAB - A Next-Generation Internet Testbed", 2020, <https://ieeexplore.ieee.org/abstract/document/9259355>.

[SCMP] Anapaya, ETH, and SCION, "SCMP Documentation", 2024, <https://docs.scion.org/en/latest/protocols/scmp.html>.

Acknowledgments

Many thanks go to Alvaro Retana (Futurewei), Joel M. Halpern (Ericsson), William Boye (Swiss National Bank), Matthias Frei (SCION Association), Kevin Meynell (SCION Association), Juan A. Garcia Prado (ETH Zurich), and Roger Lapuh (Extreme Networks), for reviewing this document. We also thank Daniel Galn Pascual and Christoph Sprenger from the Information Security Group at ETH Zurich for their inputs based on their formal verification work on SCION. We are also very grateful to Adrian Perrig (ETH Zurich), for providing guidance and feedback about every aspect of SCION. Finally, we are indebted to the SCION development teams of Anapaya, ETH Zurich, and the SCION Association for their practical knowledge and for the documentation about the SCION Control Plane, as well as to the authors of [CHUAT22] - the book is an important source of input and inspiration for this draft.

Deployment Testing: SCIONLab

SCIONLab is a global research network that is available to test the SCION architecture. You can create and use your ASes using your own computation resources which allows you to gain real-world experience of deploying and managing a SCION network.

More information can be found on the SCIONLab website and in the [SCIONLAB] paper.

Full Control Service gRPC API

The following code blocks provide, in protobuf format, the entire API by which control services interact.

```
service SegmentLookupService {
    // Segments returns all segments that match the request.
    rpc Segments(SegmentsRequest) returns (SegmentsResponse) {}
}

message SegmentsRequest {
    // The source ISD-AS of the segment.
    uint64 src_isd_as = 1;
    // The destination ISD-AS of the segment.
    uint64 dst_isd_as = 2;
}

enum SegmentType {
    // Unknown segment type.
    SEGMENT_TYPE_UNSPECIFIED = 0;
    // Up segment.
    SEGMENT_TYPE_UP = 1;
    // Down segment.
    SEGMENT_TYPE_DOWN = 2;
    // Core segment.
    SEGMENT_TYPE_CORE = 3;
}

message SegmentsResponse {
    message Segments {
        // List of path segments.
        repeated PathSegment segments = 1;
    }

    // Mapping from path segment type to path segments.
    // The key is the integer representation of the SegmentType enum.
    map<int32, Segments> segments = 1;
}
```

Figure 25: Control Service gRPC API - Segment lookup. This API is exposed on the SCION dataplane by the control services of core ASes and exposed on the intra-domain protocol network.

```
service SegmentRegistrationService {
    // SegmentsRegistration registers segments at the remote.
    rpc SegmentsRegistration(SegmentsRegistrationRequest) returns (
        SegmentsRegistrationResponse) {}
}

message SegmentsRegistrationRequest {
    message Segments {
        // List of path segments.
        repeated PathSegment segments = 1;
    }

    // Mapping from path segment type to path segments.
    // The key is the integer representation of the SegmentType enum.
    map<int32, Segments> segments = 1;
}

message SegmentsRegistrationResponse {}
```

Figure 26: Control Service gRPC API - Segment registration. This API is only exposed by core ASes and only on the SCION dataplane.

```
service SegmentCreationService {
    // Beacon sends a beacon to the remote control service.
    rpc Beacon(BeaconRequest) returns (BeaconResponse) {}
}

message BeaconRequest {
    // Beacon in form of a partial path segment.
    PathSegment segment = 1;
}

message BeaconResponse {}
```

Figure 27: Control Service gRPC API - Segment creation

```
message PathSegment {
    // The encoded SegmentInformation. It is used for signature input.
    bytes segment_info = 1;
    // Entries of ASes on the path.
    repeated ASEntry as_entries = 2;
}

message SegmentInformation {
```

```
    // Segment creation time set by the originating AS. Segment
    // expiration time is computed relative to this timestamp.
    // The timestamp is encoded as the number of seconds elapsed
    // since January 1, 1970 UTC.
    int64 timestamp = 1;
    // The 16-bit segment ID integer used for MAC computation.
    uint32 segment_id = 2;
}

message ASEntry {
    // The signed part of the AS entry. The body of the SignedMessage
    // is the serialized ASEntrySignedBody. The signature input is
    // defined as follows:
    //
    //  input(ps, i) = signed.header_and_body || associated_data(ps, i)
    //
    //  associated_data(ps, i) =
    //          ps.segment_info ||
    //          ps.as_entries[1].signed.header_and_body ||
    //          ps.as_entries[1].signed.signature ||
    //          ...
    //          ps.as_entries[i-1].signed.header_and_body ||
    //          ps.as_entries[i-1].signed.signature
    //
    proto.crypto.v1.SignedMessage signed = 1;
    // The unsigned part of the AS entry.
    proto.control_plane.v1.PathSegmentUnsignedExtensions unsigned = 2;
}

message SignedMessage {
    // Encoded header and body.
    bytes header_and_body = 1;
    // Raw signature. The signature is computed over the concatenation
    // of the header and body, and the optional associated data.
    bytes signature = 2;
}

message HopEntry {
    // Material to create the data-plane Hop Field.
    HopField hop_field = 1;
    // MTU on the ingress link.
    uint32 ingress_mtu = 2;
}

message PeerEntry {
    // ISD-AS of peer AS. This is used to match peering segments
    // during path construction.
    uint64 peer_isd_as = 1;
```

```
    // Remote peer interface identifier. This is used to match
    // peering segments
    // during path construction.
    uint64 peer_interface = 2;
    // MTU on the peering link.
    uint32 peer_mtu = 3;
    // Material to create the data-plane Hop Field
    HopField hop_field = 4;
}

message HopField {
    // Ingress interface identifier.
    uint64 ingress = 1;
    // Egress interface identifier.
    uint64 egress = 2;
    // 8-bit encoded expiration offset relative to the segment
    // creation timestamp.
    uint32 exp_time = 3;
    // MAC used in the dataplane to verify the Hop Field.
    bytes mac = 4;
}
```

Figure 28: Control Service gRPC API - Segment representation

```
enum SignatureAlgorithm {
    // Unspecified signature algorithm. This value is never valid.
    SIGNATURE_ALGORITHM_UNSPECIFIED = 0;
    // ECDS with SHA256.
    SIGNATURE_ALGORITHM_ECDSA_WITH_SHA256 = 1;
    // ECDS with SHA384.
    SIGNATURE_ALGORITHM_ECDSA_WITH_SHA384 = 2;
    // ECDS with SHA512.
    SIGNATURE_ALGORITHM_ECDSA_WITH_SHA512 = 3;
}

// Low-level representation of HeaderAndBody used for signature
// computation input. This should not be used by external code.
message HeaderAndBodyInternal {
    // Encoded header suitable for signature computation.
    bytes header = 1;
    // Raw payload suitable for signature computation.
    bytes body = 2;
}

message Header {
    // Algorithm used to compute the signature.
```

```
      SignatureAlgorithm signature_algorithm = 1;
      // Optional arbitrary per-protocol key identifier.
      bytes verification_key_id = 2;
      // Optional signature creation timestamp.
      google.protobuf.Timestamp timestamp = 3;
      // Optional arbitrary per-protocol metadata.
      bytes metadata = 4;
      // Length of associated data that is covered by the signature, but
      // is not included in the header and body. This is zero, if no
      // associated data is covered by the signature.
      int32 associated_data_length = 5;
  }

  message ASEntrySignedBody {
      // ISD-AS of the AS that created this AS entry.
      uint64 isd_as = 1;
      // ISD-AS of the downstream AS.
      uint64 next_isd_as = 2;
      // The required regular hop entry.
      HopEntry hop_entry = 3;
      // Optional peer entries.
      repeated PeerEntry peer_entries = 4;
      // Intra AS MTU.
      uint32 mtu = 5;
      // Optional extensions.
      proto.control_plane.v1.PathSegmentExtensions extensions = 6;
  }

  message VerificationKeyID {
      uint64 isd_as = 1;
      bytes subject_key_id = 2;
      uint64 trc_base = 3;
      uint64 trc_serial = 4;
  }
```

Figure 29: Control Service gRPC API - Signed ASEntry representation

```
service TrustMaterialService {
    // Return the certificate chains that match the request.
    rpc Chains(ChainsRequest) returns (ChainsResponse) {}
    // Return a specific TRC that matches the request.
    rpc TRC(TRCRequest) returns (TRCResponse) {}
}

message ChainsRequest {
    // ISD-AS of Subject in the AS certificate.
```

```
    uint64 isd_as = 1;
    // SubjectKeyID in the AS certificate.
    bytes subject_key_id = 2;
    // Point in time at which the AS certificate must still be valid. In seconds
    // since UNIX epoch.
    google.protobuf.Timestamp at_least_valid_until = 3;
    // Point in time at which the AS certificate must be or must have been
    // valid. In seconds since UNIX epoch.
    google.protobuf.Timestamp at_least_valid_since = 4;
}

message ChainsResponse {
    // List of chains that match the request.
    repeated Chain chains = 1;
}

message Chain {
    // AS certificate in the chain.
    bytes as_cert = 1;
    // CA certificate in the chain.
    bytes ca_cert = 2;
}

message TRCRequest {
    // ISD of the TRC.
    uint32 isd = 1;
    // BaseNumber of the TRC.
    uint64 base = 2;
    // SerialNumber of the TRC.
    uint64 serial = 3;
}

message TRCResponse {
    // Raw TRC.
    bytes trc = 1;
}

// VerificationKeyID is used to identify certificates that authenticate the
// verification key used to verify signatures.
message VerificationKeyID {
    // ISD-AS of the subject.
    uint64 isd_as = 1;
    // SubjectKeyID referenced in the certificate.
    bytes subject_key_id = 2;
    // Base number of the latest TRC available to the signer at the time of
    // signature creation.
    uint64 trc_base = 3;
    // Serial number of the latest TRC available to the signer at the time of
```

```
    // signature creation.
    uint64 trc_serial = 4;
}
```

Figure 30: Control Service gRPC API - Trust Material representation

In case of failure, gRPC calls return an error as specified by the gRPC framework. That is, a non-zero status code and an explanatory string.

Use of the SCION Data Plane

The SCION Control Plane RPC APIs rely on QUIC connections carried by the SCION dataplane. The main difference between QUIC over native UDP and QUIC over UDP/SCION is the need for a UDP/SCION connection initiator to identify the relevant peer (service resolution) and to select a path to it. Since the Control Service is itself the source of path segment information, the following bootstrapping strategies apply:

* Neighboring ASes craft one-hop-paths directly. This allows multihop paths to be constructed and propagated incrementally.

* Constructed multihop paths are registered with the Control Service at the origin core AS. The path to that AS is the very path being registered.

* Paths to far ASes are available from neighboring ASes. Clients obtain paths to remote ASes from their local Control Service.

* Control services respond to requests from remote ASes by reversing the path via which the request came.

* Clients find the relevant Control Service endpoint by resolving a "service address" (that is an address where the DT/DL field of the common header is set to 1/0 (see [I-D.dekater-scion-dataplane]).

The mechanics of service address resolution are the following:

* To resolve the address of the control service at a given AS, a client sends a ServiceResolutionRequest RPC (which has no parameters) to an endpoint address constructed as follows:

  - Common Header:

    o Path type: SCION (0x01)

    - o DT/DL: "Service" (0b0100)

  - Address Header:

    - o DstHostAddr: "SVC_CS" (0x0002)

  - UDP Header:

    - o DstPort: 0

* The ingress border router at the destination AS resolves the service destination to an actual endpoint address. This document does not mandate any specific method for this resolution.

* The ingress border router forwards the message to the resolved address.

* The destination service responds to the client with a ServiceResolutionResponse. That response contain one or more transport options.

* The client uses the address and port from the "QUIC" option to establish a QUIC connection, which can then be used for regular RPCs.

The following code block provides the full service resolution API in the Protobuf message format.

```
package proto.control_plane.v1;

// A ServiceResolutionRequest must always fit within a UDP datagram. If
// the request does not fit, there is no mechanism for clients and
// servers to establish control-plane reachability.
message ServiceResolutionRequest {}

// A ServiceResolutionResponse must always fit within a UDP datagram. If
// the response does not fit, there is no mechanism for clients and
// servers to establish control-plane reachability.
message ServiceResolutionResponse {
    // Supported transports to reach the service,
    //
    // List of known transports:
    // - QUIC
    //
    // Unknown values should be ignored by clients.
    map<string, Transport> transports = 1;
}

message Transport {
    // Protocol specific server address descriptor.
    //
    // Supported address format for QUIC:
    //  192.168.0.1:80
    //  [2001:db8::1]:80
    //
    //  Missing ports / zero port / invalid port values should be
    // treated by clients as errors.
    string address = 1;
}
```

Figure 31: Service Resolution gRPC API definition

Path-Lookup Examples

To illustrate how the path lookup works, we show two path-lookup examples in sequence diagrams. The network topology of the examples is represented in Figure 32 below. In both examples, the source endpoint is in AS A. Figure 33 shows the sequence diagram for the path lookup process in case the destination is in AS D, whereas Figure 34 shows the path lookup sequence diagram if the destination is in AS G. ASes B and C are core ASes in the source ISD, while E and F are core ASes in a remote ISD. Core AS B is a provider of the local AS, but AS C is not, i.e. there is no up-segment from A to C. "CS" stands for Control Service.

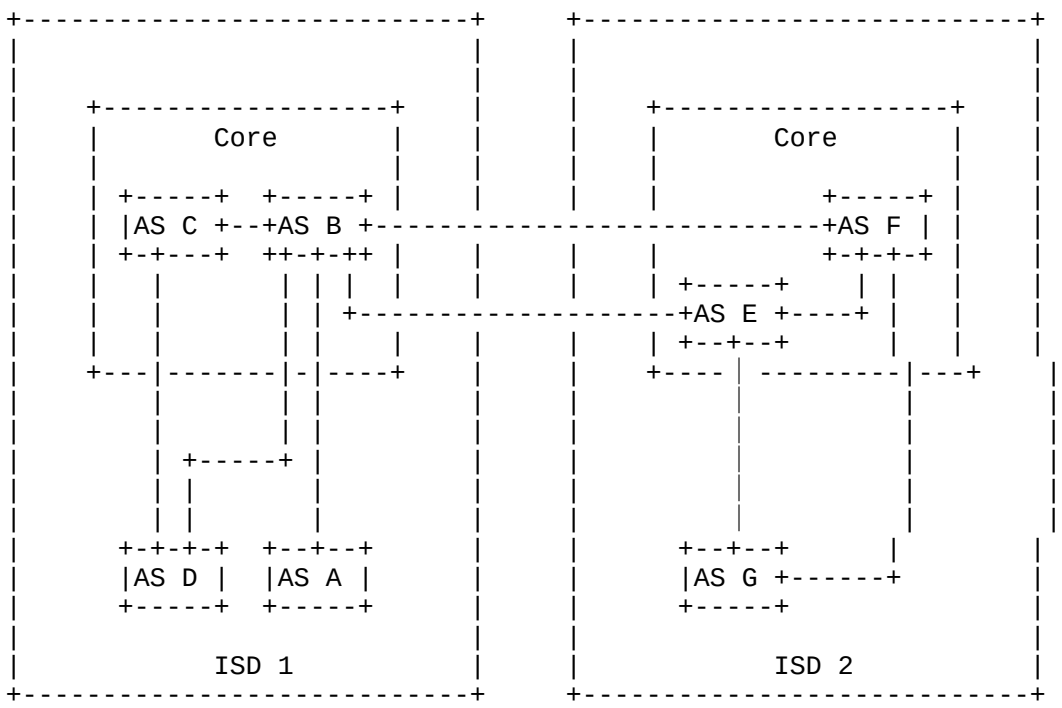
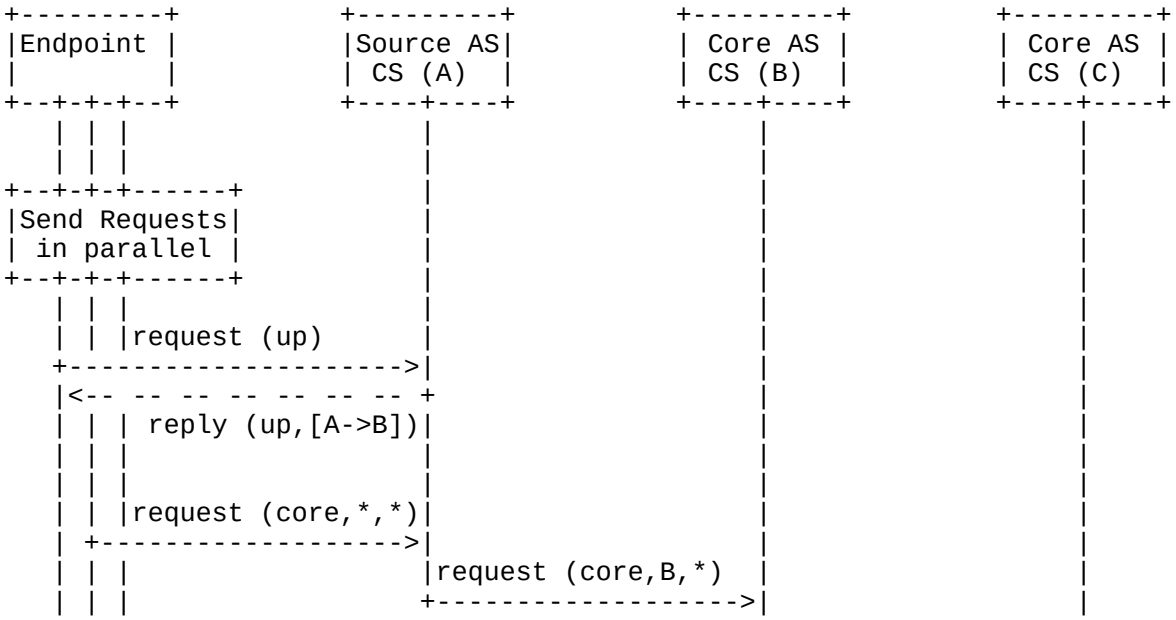


Figure 32: Topology used in the path lookup examples



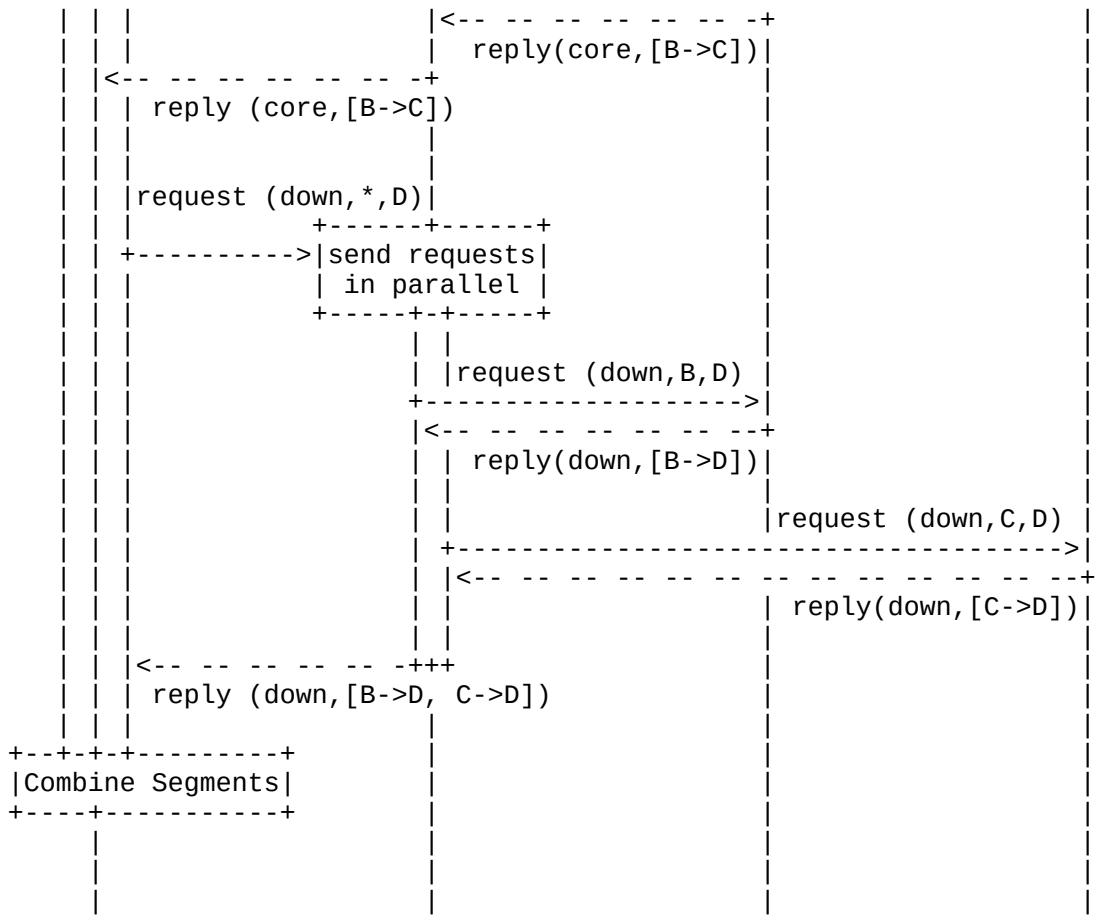
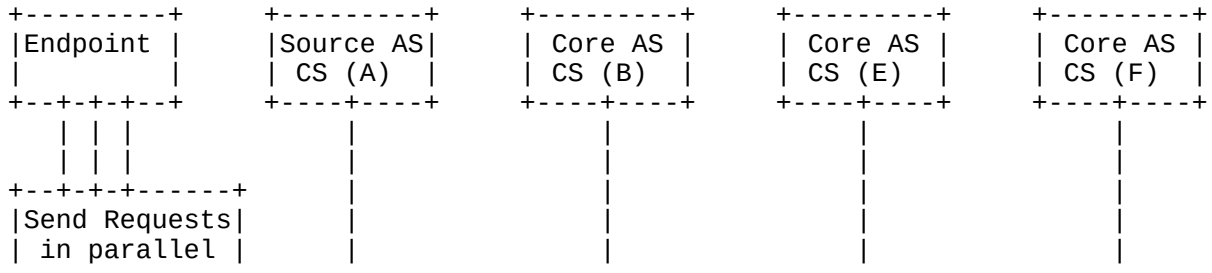


Figure 33: Sequence diagram illustrating a path lookup for a destination D in the source ISD. The request (core, x, x) is for all pairs of core ASes in the source ISD. Similarly, (down, x, D) is for down segments between any core AS in the source ISD and destination D.



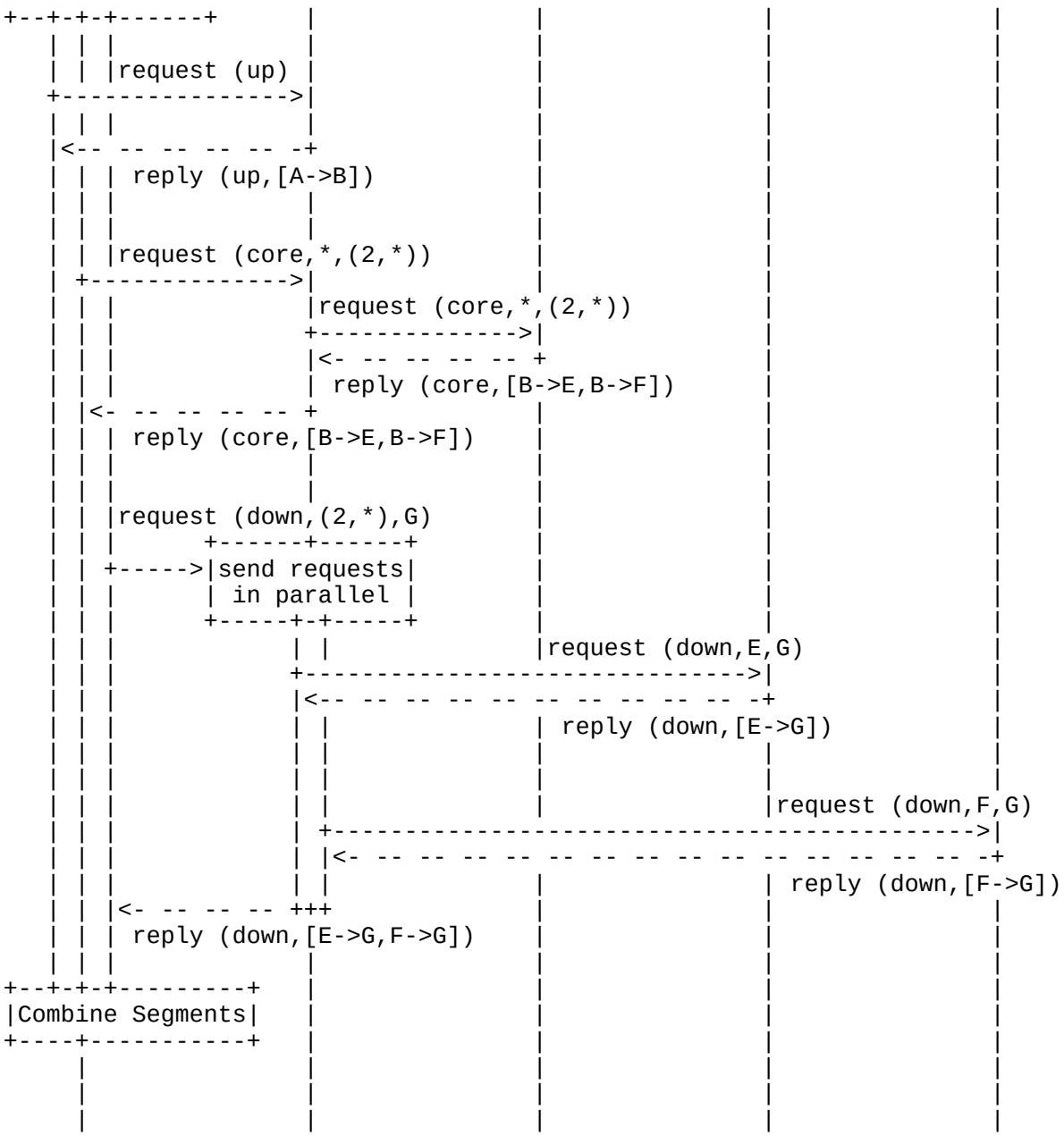

request (up)
reply (up,[A->B])
request (core,*,(2,*))
request (core,*,(2,*))
reply (core,[B->E,B->F])
reply (core,[B->E,B->F])
request (down,(2,*),G)
send requests
in parallel
request (down,E,G)
reply (down,[E->G])
request (down,F,G)
reply (down,[F->G])
reply (down,[E->G,F->G])
Combine Segments

Figure 34: Sequence diagram illustrating a path lookup for a destination G in a remote ISD. The request (core, x, (2, x)) is for all path segments between a core AS in the source ISD and a core AS in ISD 2. Similarly, (down, (2, x), G) is for down segments between any core AS in ISD 2 and destination G.

Change Log

Changes made to drafts since ISE submission. This section is to be removed before publication.

draft-dekater-scion-controlplane-09

Major changes:

* "SCION AS numbers": make text representation for lower 32-bit ASes consistent with PKI draft, add reference to allocation

Minor changes:

* Nits and wording improvements
* Reviewed use of normative language
* Figures: redraw and use aasvg when possible
* "Paths and Links": clarify relationship between path segments and links
* Ensure consistent use of example ranges

draft-dekater-scion-controlplane-08

Major changes:

* Abstract: reword, mention goal and that document is not an Internet Standard
* "Propagation of PCBs" section:
  - clarify checks at reception
  - introduce criteria for PCB selection policies
  - remove superfluous policy example figure
  - Propagation Interval and Best PCBs Set Size: mention tradeoff between scalability and amount of paths discovered.

- reorganize order of paragraphs

* New section "Security Properties" in Security considerations, based on formal model of SCION

* New figure: Control Service gRPC API - Trust Material definitions

Minor changes:

* Moved "Special-Purpose SCION AS Numbers" table later in text

* Split "Circular dependencies and partitioning" into two sections: "Bootstrapping ability" and "Resistance to partitioning".

* Explain why PCBs have a next_isd_as field

* Qualified better the choice of time allowance in the definition of segment from the future in section "PCB Validity".

draft-dekater-scion-controlplane-07

* Moved SCMP specification from draft-dekater-scion-dataplane-03 to this document

draft-dekater-scion-controlplane-06

Major changes:

* New section: Path MTU

* New section: Monitoring Considerations

* Completed description of Control Services gRPC API in appendix

Minor changes:

* Introduction: clarify goal of the document

* Clarify typical vs recommended-limits values for best PCB set size and for certificate validity duration.

* Clarify text representation of ISD-AS

* General rewording

* Added reference to SCIONLab as a testbed for implementors

* Introduced this change log

draft-dekater-scion-controlplane-05

Minor changes:

* Clarify beaconing fast retry at bootstrapping

draft-dekater-scion-controlplane-04

Major changes:

* Clarified selection of MAC including a default algorithm
* New section: PCB validity
* New section: configuration
* New section: Path Discovery Time and Scalability
* New section: Effects of Clock Inaccuracy
* New appendix: Control Service gRPC API

Minor changes:

* Introduction: Added overview of SCION components
* Clarified path reversibility, link types, Interface IDs
* Fixed private AS range typo
* Clarified PCB selection policies and endpoint requirements
* Clarified PCB propagation
* General edits to make terminology consistent, remove duplication and rationalize text
* Removed forward references
* Added RFC2119 compliant terminology

Authors' Addresses

Corine de Kater
SCION Association
Email: c_de_kater@gmx.ch

Nicola Rustignoli
SCION Association
Email: nic@scion.org

Samuel Hitz
Anapaya Systems
Email: hitz@anapaya.net