November 4th, 2025

Draft SCHC Minimal Architecture

From deployment scenarios to a minimal architecture

Authors: Q. Lampin, M. Dumay, P. Surbayrole

Presenters: Q. Lampin, M. Dumay

IETF 124, Montreal

Agenda

- Foreword, objective & methodology
- From scenarios to architecture requirements
  - The simplest deployment
  - Three endpoints
  - The dynamic traffic scenario
- Conclusion, questions to the WG, action points

Foreword, objectives & methodology

Why this draft?

- LPWAN WG evolved to SCHC WG
  - New use-cases, new requirements, new challenges
    → new Architecture

- Draft started as a home-work exercise to study draft-ietf-schc-architecture.
- Explain the terminology, the functional entities, their interactions.
- Document the reasoning, share with the working group.

How?

- Start from the legacy LPWAN scenario
  - Identify REQUIRED entities and functions
  - Discuss terminology & interactions
  - Compare with draft-ietf-schc-architecture
  - Document the findings → this document
  - Consider a more realistic/complex scenario & repeat

IETF

From scenarios to architecture requirements

6 scenarios are considered (as of -00)

1 - Point-to-Point communication between an **IoT device** and **gateway**

2 - Multipoint-to-Point communication between **two IoT devices** and **gateway**

3 - **Dynamic traffic** between IoT devices and gateway

4 - **Multiple instances** on Endpoints

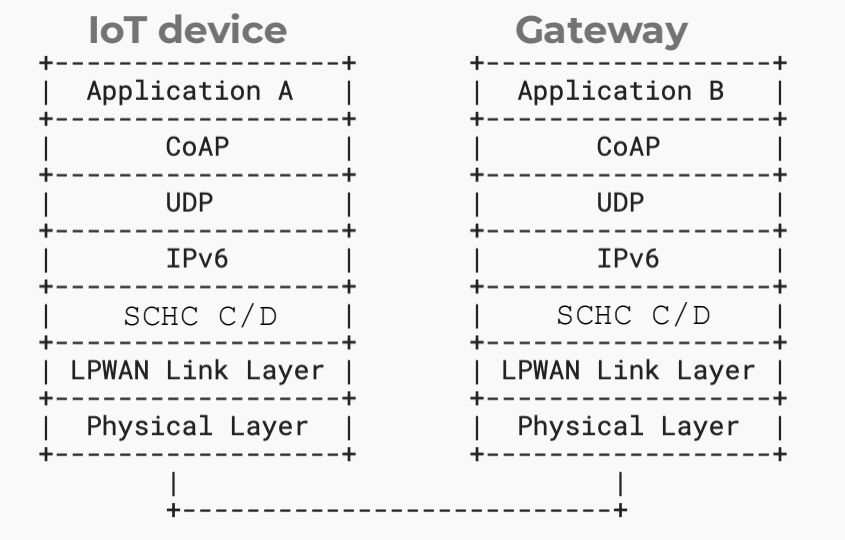
5 - **Heterogeneous** devices, modern OSes

6 - **Dynamic context provisioning**, the cold boot scenario

Point-to-Point communication

Key considerations

- Single network stack/link-layer/network-interface, etc.
- SCHC hard-wired in network stack, evaluates & compresses every packet.



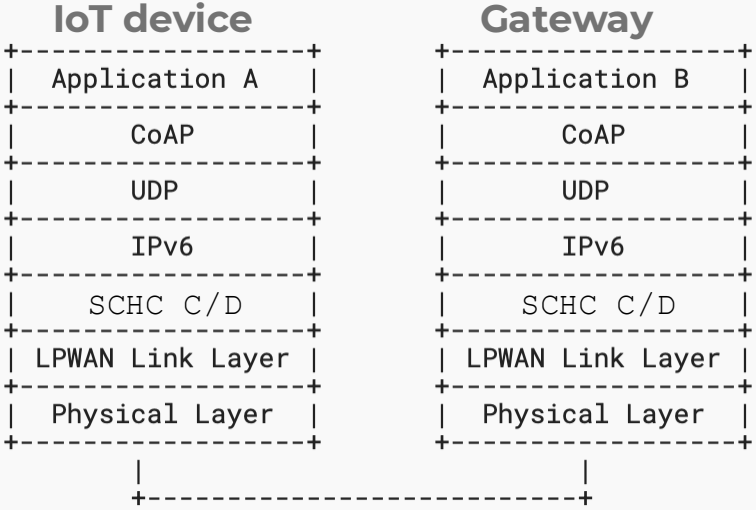
Point-to-Point communication: terminology requirements

Generic name for devices running SCHC

- Not a hierarchical relationship.
  → Rules out core/device or equivalent naming.
- A single device may run SCHC at multiple stages in its network stack.
  → Need to distinguish from the entity running SCHC.

draft-ietf-schc-architecture proposes Endpoint

→ Aligns well with those requirements.



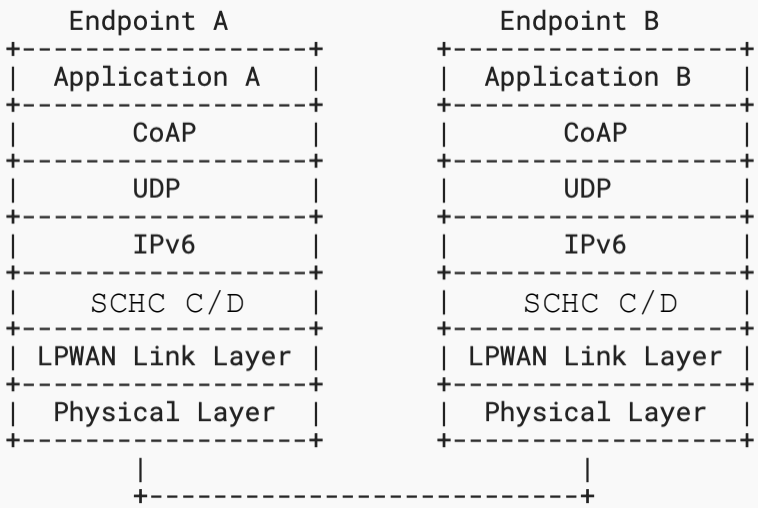
Point-to-Point communication: terminology requirements

Generic name for SCHC processes running on Endpoint

- Different concept from the association of the two processes.
  → Rules out session, association, or equivalent.

draft-ietf-schc-architecture has no definition yet

RFC8824, 9363, 9441 use Instance
→ Aligns well with … dictionaries' definitions



"an example of a particular type of action or situation : an occasion of something happening", src: https://www.britannica.com/dictionary/instance
"Instance (computer science), referring to any running process or to an object as an instance of a class", src: https://en.wikipedia.org/wiki/Instantiation
"an example or single occurrence of something", src: Oxford English dictionary

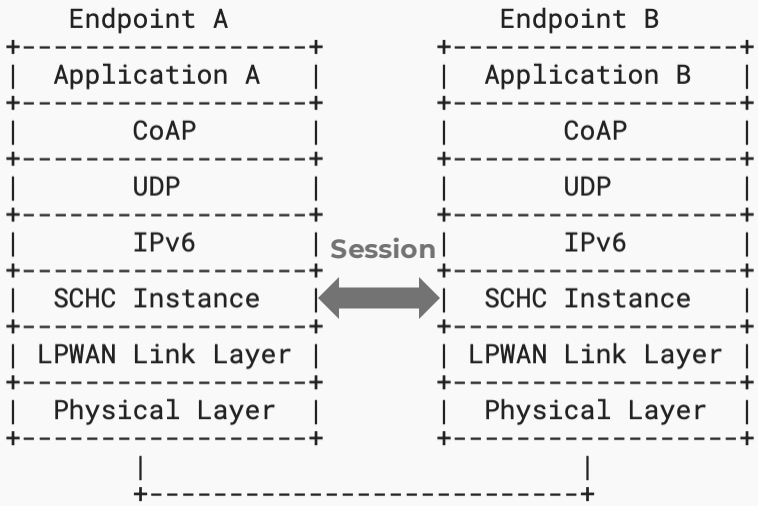
Point-to-Point communication: terminology requirements

Generic name for the communication session between the two instances

- Session may be terminated, restarted, etc.
  → Concept of duration.

draft-ietf-schc-architecture uses Instance

*"SCHC **Instance**. The **session** between SCHC end-points in two or more peer nodes [...]"*
→ Conflicts with RFC8824, 9363, 9441
→ Session != Instance



Let's keep it simple, it's a Session

"a period devoted to a particular activity" src: Oxford English Dictionary
"In computer science, a session is a short trade of information between two or more devices, or between a computer and a person.", src: https://simple.wikipedia.org/wiki/Session_(computer_science)

Point-to-Point communication: terminology requirements

About the legacy use of the term Instance

Focus on LPWAN RFCs

RFC8824
*"CoAP is an application protocol, so CoAP compression requires installing common Rules between the **two SCHC instances**."*

RFC9363
*"This document formalizes the description of the Rules for better **interoperability between SCHC instances** either to exchange a set of Rules or to modify the parameters of some Rules."*

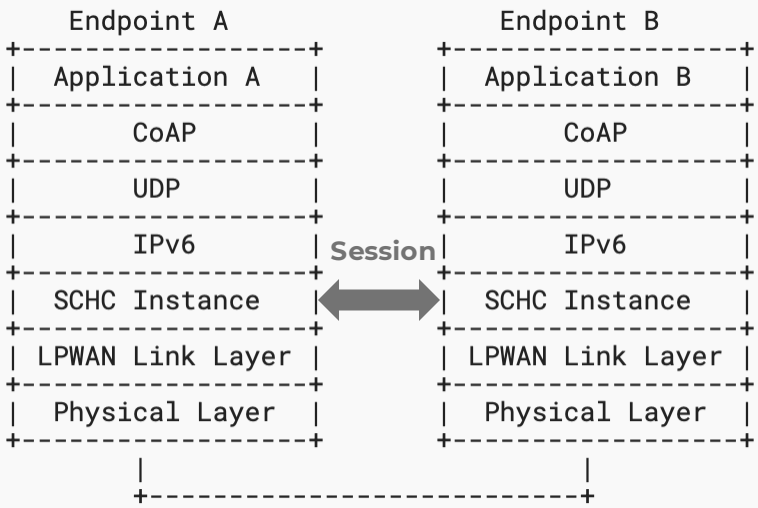
RFC9441
*"Therefore, a device must be allowed to modify only its own Rules on the **remote SCHC instance**."*

If Instance means session, then we need to publish errata for all those RFCs!

Point-to-Point communication: functional requirements

SoR of Endpoint A MUST be compatible with that of B

- Any compression rule **in use** by Instance@A **to communicate with** Instance@B **MUST** be known by Instance@B
- Rules' Field Descriptors in the same order*
- Parsers at Instance@A and Instance@B delineate the same header fields**.
  → This is not an implementation consideration as it requires signaling.



*See appendix A

**fid-ipv6-trafficclass != fid-ipv6-trafficclass-ds + fid-ipv6-trafficclass-ecn
not convinced? Try with Match-Mapping/Mapping-Sent
and see appendix B

Three Endpoints

Key considerations

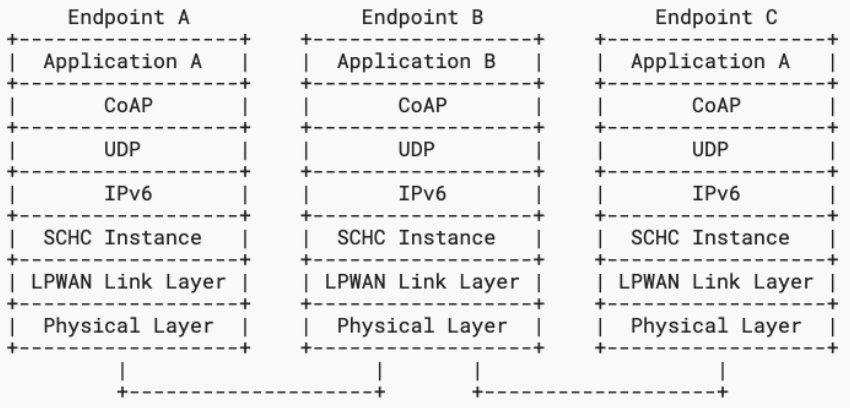
- Additional Endpoint(s), **same owner/tenant.**
- **Similar traffic** (same protocol stack, same header fields, similar field values).

In Laurent own words, “imagine zillions of Endpoints”

“How do we **deploy** ‘SCHC’ to zillions of Endpoints?”

Not one-by-one*

→ **Shared SCHC** configuration for a **Fleet** of Endpoints.



*see Appendix C

Three Endpoints: terminology requirements

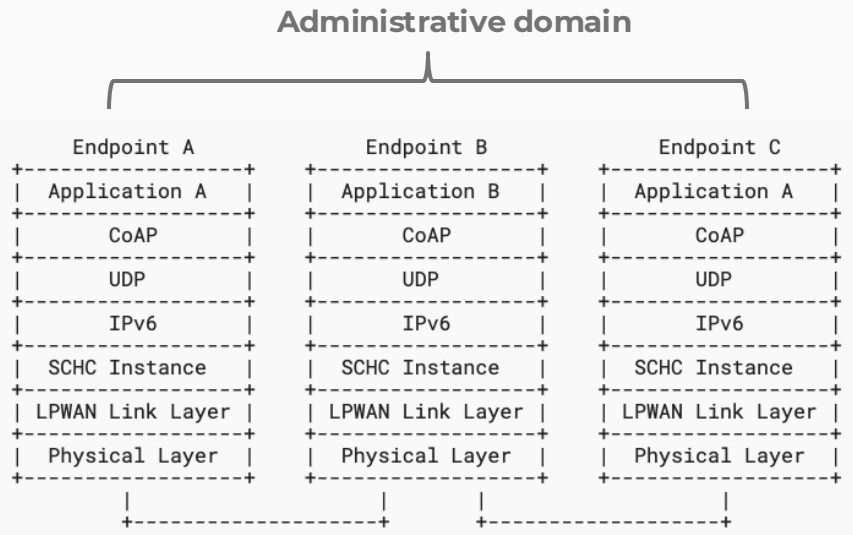
Name for the “administrative domain”

- Refers to groups of instances that shares a common “configuration”, for ex. Context.
- Related to Instance Configuration management.

draft-ietf-schc-architecture has no definition yet

draft-ietf-6lo-schc-15dot4 introduces “provisioning domains” (page 8)

draft-lampin-schc-minimal-architecture proposes **Domain**



Eric Vyncke, during interim 13 of 2025: “domain is overloaded at the IETF. I would prefer “area” or “zone”.
→ needs clarification, question asked on ML.

Three Endpoints: terminology requirements

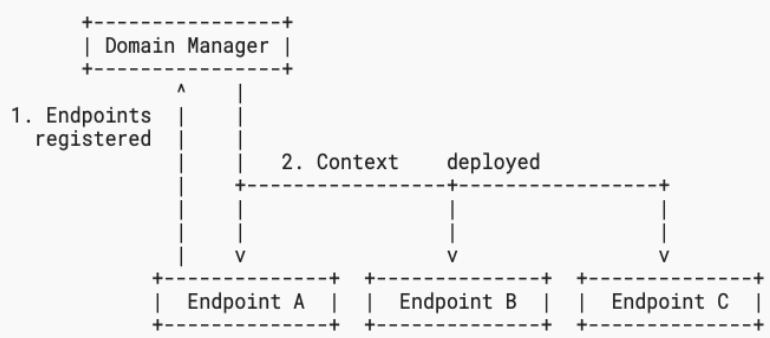
Name for the entity in charge of the Domain

- Admission of Endpoints in the Domain
- Provisioning of configuration, e.g. Context to Instances.
- Life-cycle of instances

draft-ietf-schc-architecture proposes "Instance Manager"

draft-lampin-schc-minimal-architecture agrees with the term **"Manager"** and aligns it with the "Domain" terminology

Hence, "Domain Manager"



Three Endpoints: functional requirements

Desired features

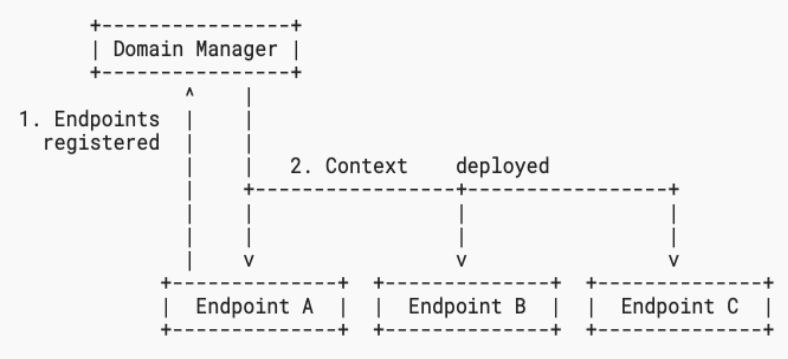
- Contexts updates of all endpoints of the “administrative domain”.
- Shared base/template rules between endpoints of an “administrative domain”.
- Specialized rules for specific instances (such as customizing the TV of a CoAP port)

The **Context** of all three Endpoints **MUST** be “***compatible****”

- SoRs, parsers, rule IDs, DI, consistent across the Domain*.

→ **Domain Manager in charge of synchronization of Contexts**

*More details in appendix D and E



Dynamic Traffic

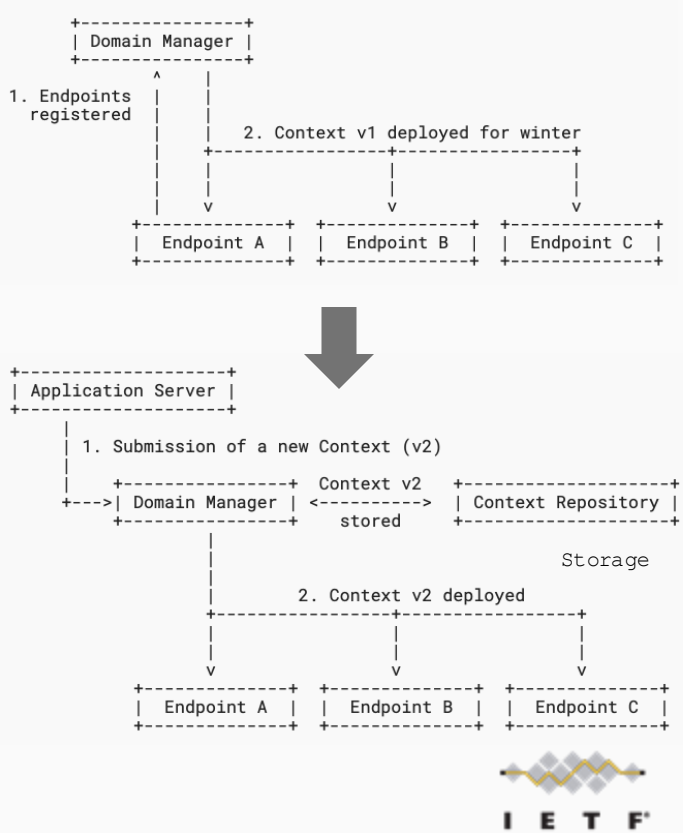
Key considerations

- Traffic characteristics change over time.
- New Context pushed to Endpoints.

“This scenario highlights the need for a dynamic context update mechanism that allows the Domain Manager to update the Context of all Instances belonging to the Domain.”

The protocol to deploy the Context is out of scope of the architecture, however we need an **Entity to store Contexts**.

→ Deploying a Context takes time.
→ Persisting the latest Context for new Endpoints.
→ Resolve synchronization issues.



Dynamic Traffic

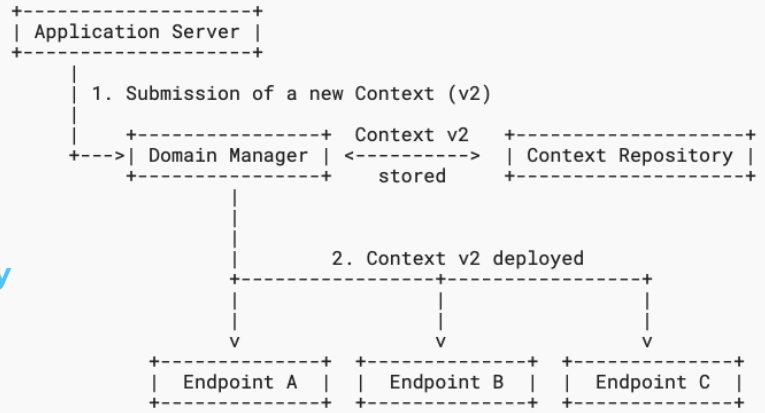
Terminology requirements

Name for Context Storage

- Functions of storage, versioning for contexts.
- Maps instances to context content.

draft-ietf-schc-architecture has no definition yet

draft-lampin-schc-minimal-architecture proposes Context Repository



Functional requirements for the Context Repository

Context storage/persistence in the Domain

- Source authority for the domain
- In charge of Context versioning

About Context versioning:
→ Contexts MUST have a version attribute

Multiple SCHC Instances

Key considerations

- Multiple Instances hosted by the same Endpoint
- May coexist at **different** levels, or the **same**

Challenges

- **Datagram dispatch**

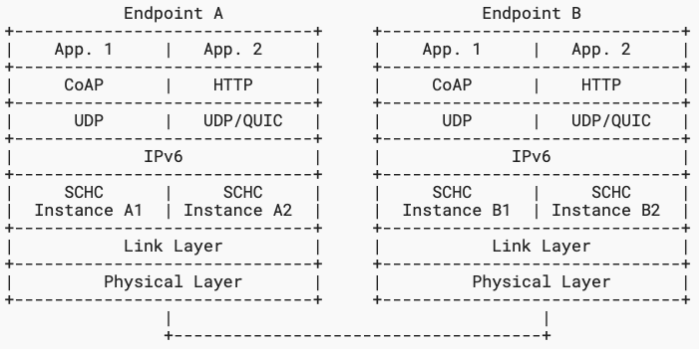
The Endpoints must be able to **dispatch datagrams to** the appropriate **Instances** based on the protocol or application in use.

→ **Need an entity for dispatching datagrams**

- **Instance/Context Identification**

Each **Instance or Context must be uniquely identifiable** to allow the Domain Manager to **update the Context of a specific Instance**.

→ **Need unique identifiers for Contexts/Instances.**



Multiple SCHC Instances: terminology requirements

Name for the entity that dispatches datagrams

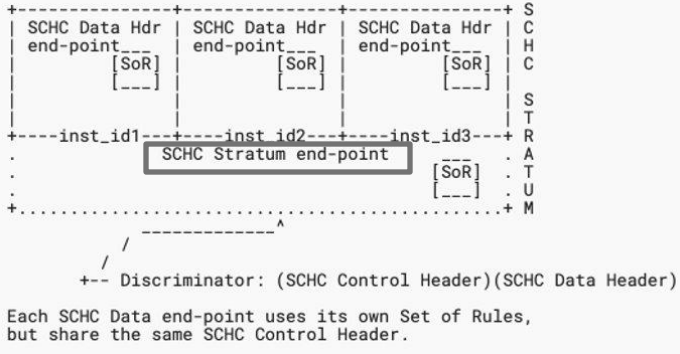
- Dispatches, routes “datagrams” to Instances
- Single point of entry for all Instances
  → Need to distinguish from the entity running SCHC.

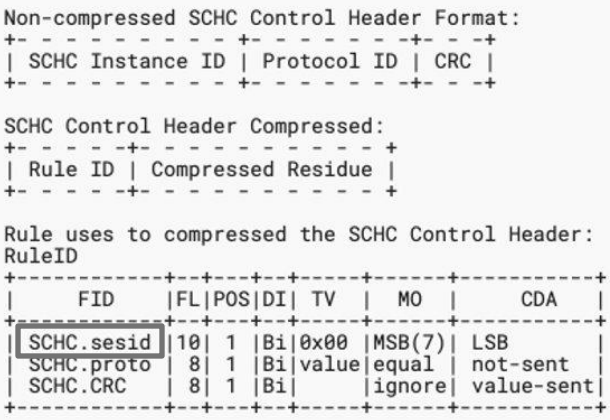
draft-ietf-schc-architecture describes a 2-tiers routing

SCHC Stratum Endpoint/SCHC Data Endpoint
→ Lacks the single point of entry.

draft-lampin-schc-minimal-architecture proposes “Dispatcher*”

*Dispatcher != Discriminator, see Appendix F





Multiple SCHC Instances: terminology requirements

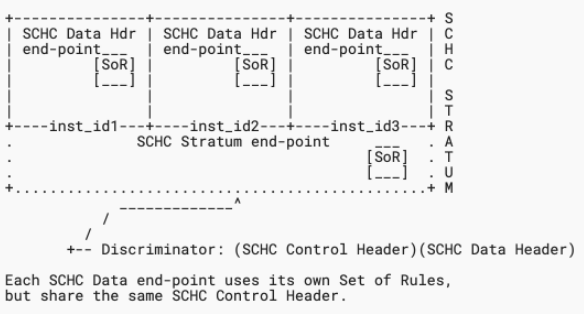
Name for Unique Identifier

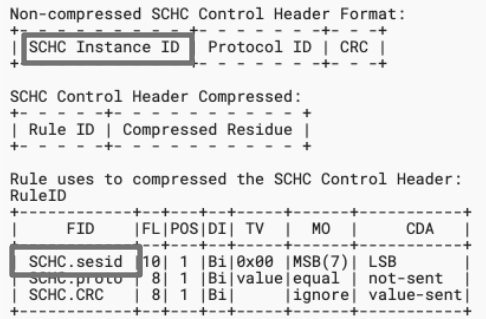
- Related to the Context/Instance deployed on devices.
- Unique among all Endpoints of the Domain.*

Each **Instance or Context must be uniquely identifiable** to allow the Domain Manager to **update the Context of a specific Instance**.

draft-ietf-schc-architecture uses *Instance ID (within a stratum), was SessionID (sesid)*

draft-lampin-schc-minimal-architecture proposes "Context ID"





* Endpoints registered in multiple Domains?

IETF

The other scenarios, and more

Heterogeneous devices, modern OSes

- Considerations on the implementation of SCHC in “Modern” OSes, on top of heterogeneous hardware.
- What do we **need in the architecture** to enable an implementation of SCHC?
  → **Profile**, per-device, deployed with the Context,
  i.e. what’s needed to “plug” the SCHC instance in the network stack.

Dynamic context provisioning, the cold boot scenario

- A device is pre-provisioned in a Domain.
- What do we **need in the architecture** to let the device fetch its SCHC Contexts?
  → **Discovery mechanism/known URI** of the Domain Manager, e.g. DNS-SD.

+ descriptions of the Core Components
+ Tentative diagrams of the interactions of the Components

Wrap-up

Work In Progress document.

GOAL: Discuss, explain, record the arguments for each element of the Architecture

- Uses concrete scenarios
- Identifies requirements
- Proposes terminology
- Exposes interactions

In our opinion, a good practice for identifying issues.

IETF

Pending issues & WG Action Points

Terminology for Endpoint/Instance/Session

- RFCs 8824, 9363, 9441 at stake

Pair-wise vs. Domain-wise configuration

- Need for an entity for fleet management

SoR vs Context

- In our opinion, **Context != SoR,**
- Context := { Context ID, version, SoR, parser ID, Direction Indicator, ...}
  → **YANG Data model**

Domain, Context Equality or Compatibility

- Strict Context Equality is not possible (Direction Indicator)
- **Device-specific Rules** vs **n-rules-fit-all**
- **WG discussion needed on SoR equality vs compatibility**
- Only Pair-wise Contexts vs **Domain-wise Contexts**

IETF

Pending issues & WG Action Points

Discriminator vs Dispatcher

- The Dispatcher should be in the Architecture
- Its configuration should be mentioned in the Architecture
- How it works is out of scope
- Content of its configuration is out of scope

Not strictly related to the architecture but NEEDS TO BE ADDRESSED ASAP

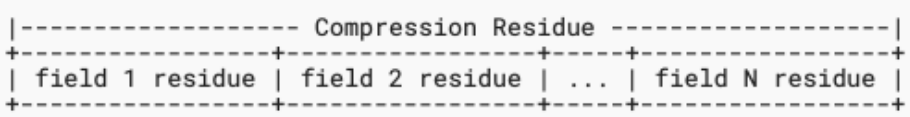
- Field Order for rule entries
- Context Data Model
- Parser Identification

THANK YOU

Appendix A

Order of the Field Descriptors

Exact same field order?



RFC8724 specifies:

"The Compression Residue for the packet header is the concatenation of the non-empty residues for each field of the header, ***in the order the Field Descriptors appear in the Rule****. The order in which the Field Descriptors appear in the Rule is therefore semantically important."*

Pending issue:

In 9363, Rule Entries (Field Descriptors) are unordered unless… we change how field-position is used.

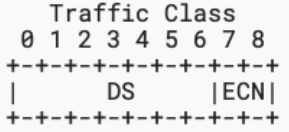
```
container schc {
  list rule {
    key "rule-id-value rule-id-length";
    uses rule-id-type;
    leaf rule-nature {
      type nature-type;
      mandatory true;
      description
        "Specify the Rule's nature.";
    }
    choice nature {
      case fragmentation {
        if-feature "fragmentation";
        uses fragmentation-content;
      }
      case compression {
        if-feature "compression";
        uses compression-content;
      }
      description
        "A Rule is for compression, for no-compression, or for
         fragmentation.";
    }
    description
      "Set of compression, no-compression, or fragmentation
       Rules identified by their rule-id.";
  }
  description
    "A SCHC set of Rules is composed of a list of Rules that are
     used for compression, no-compression, or fragmentation.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}
```

Appendix B

Parsing

fid-ipv6-trafficclass != fid-ipv6-trafficclass-ds + fid-ipv6-trafficclass-ecn

- DS subfield assume the 5 following binary values: 0010100, 001100, 001110, 010010, 010110
- ECN subfield assume the 3 following binary values: 01, 10, 11
- all combinations of DS and ECN values are valid → 15 distinct values.



- CANNOT WRITE equivalent rules using fid-ipv6-trafficclass and fid-ipv6-trafficclass-ds + fid-ipv6-trafficclass-ecn

→ Rule written with fid-ipv6-trafficclass: 4 bits residue
→ Rule written with –ds and –ecn: 5bits residue

Rule 1.a

FID	...	MO/CDA	Target Value	Residue Size
fid-ipv6-version	...	EQ/NS	0110	0
fid-ipv6-trafficclass	...	MM/MS	001010 01 '' 10 '' 11 001100 01 '' 10 '' 11 001110 01 '' 10 '' 11 010010 01 '' 10 '' 11 010110 01 '' 10 '' 11	4
			...	

Rule 1.b

FID	...	MO/CDA	Target Value	Residue Size
fid-ipv6-version	...	EQ/NS	0110	0
fid-ipv6-trafficclass-ds	...	MM/MS	001010 001100 001110 010010 010110	3
fid-ipv6-trafficclass-ecn	...	MM/MS	01 10 11	2
			...	

IETF

Appendix C

Why not Point-to-point management?

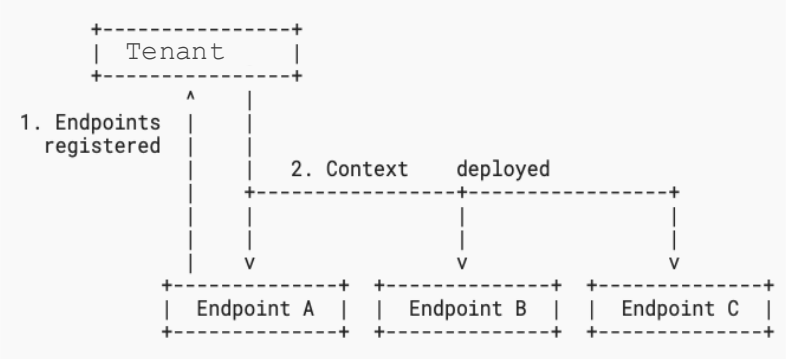
Fleets of devices:

- same firmware
- same configuration
- same traffic pattern (same headers, same fields)
- few device-specifics (device UID, network addresses, random port numbers)

→ **Shared SCHC** configuration for a **Fleet** of Endpoints.

- **Shared base/template rules**

→ **Fallback Rules** in case specific ones become inoperative.
→ **Default compression** for devices joining the Domain (better than no-compression).



We also need:

- **Per-endpoint specialization**
  → Address device specifics.
- **Fleet-wide updates**
  → Coherent configuration updates, e.g. when new configuration/firmware is pushed.

Appendix D

Context != SoR (will be discussed in -01)

Following a discussion with Laurent on the ML
(*https://mailarchive.ietf.org/arch/msg/schc/cFz0xdDAp84PlBb_0XqebNpKTUg/*)

The SoR specifies the C/D and F/R rules

How does an Endpoint know which direction is which?
→ We need something encapsulating the SoR, the Direction Indicator, this is the Context.

How does an Endpoint know which Instance a Context/SoR is for?
→ ContextID to be specified, again, attribute of the Context.

```
// SCHC table for a specific device.

container schc {
  list rule {
    key "rule-id-value rule-id-length";
    uses rule-id-type;
    leaf rule-nature {
      type nature-type;
      mandatory true;
      description
        "Specify the Rule's nature.";
    }
    choice nature {
      case fragmentation {
        if-feature "fragmentation";
        uses fragmentation-content;
      }
      case compression {
        if-feature "compression";
        uses compression-content;
      }
      description
        "A Rule is for compression, for no-compression, or for
         fragmentation.";
    }
    description
      "Set of compression, no-compression, or fragmentation
       Rules identified by their rule-id.";
  }
  description
    "A SCHC set of Rules is composed of a list of Rules that are
     used for compression, no-compression, or fragmentation.";
  reference
    "RFC 8724 SCHC: Generic Framework for Static Context Header
              Compression and Fragmentation";
}
```

SoR model

4.6. SCHC Operation

The SCHC operation requires a shared sense of which SCHC Device is Uplink (Dev to App) and which is Downlink (App to Dev), see [rfc8376]. In a star deployment, the hub is always considered Uplink and the spokes are Downlink. The expectation is that the hub and spoke derive knowledge of their role from the network configuration and SCHC does not need to signal which is hub thus Uplink vs. which is spoke thus Downlink. In other words, the link direction is determined from extrinsic properties, and is not advertised in the protocol.

Nevertheless, SCHC is very generic and its applicability is not limited to star-oriented deployments and/or to use cases where applications are very static and the state provisioned in advance. In particular, a peer-to-peer (P2P) SCHC end-point (see Section 4.4) may be set up between peers of equivalent capabilities, and the link direction cannot be inferred, either from the network topology nor from the device capability.

In that case, by convention, the device that initiates the connection that sustains the SCHC end-point is considered as being Downlink, i.e. it plays the role of the Dev in [rfc8724].

This convention can be reversed, e.g., by configuration, but for proper SCHC operation, it is required that the method used ensures that both ends are aware of their role, and then again this determination is based on extrinsic properties.

4.6.2. SoR identification

ToDo

Appendix E

Context Compatibility or Equality?

Equality*

- **Identical** SoR shared among all Instances of a Domain

→ **Single source of truth**
Easiest to manage

→ **Memory overhead**
Rules stored by every instance

→ **Management traffic overhead**
Rules provisioned on every device

Compatibility

- **Different** SoRs deployed
  - Base/template rules to all instances
  - Specific rules allocated per instance

Shared base rules
- Default rules, better than no-compression.
- Serve as fallback rules e.g., mobility use-case.
- Serve as “template rules” to reduce the cost of creating specific rules (derivation vs creation**)

Specific rules per instance
- Specialized, more efficient compression.
- Only deployed on Instances that use them.

* DI will be different
** as in draft-toutain-schc-coreconf-management

Appendix F

Discriminator vs Dispatcher

Discriminator

- Introduced in **draft-ietf-schc-architecture, information identifying the Instance.**

Dispatcher

- Introduced in **draft-lampin-schc-minimal-architecture, the entity in charge of dispatching** the datagram to the Instance.

4.2. Discriminator

The key to determine how to decompress a SCHC Control Header in a stratum is called a Discriminator.

The Discriminator is typically extrinsic to the stratum data.¶

It may be found in the datagram context, e.g., the ID of the interface, VLAN, SSID, or PPP SCHC Instance on which the datagram is received.

It may also be received in the datagram, natively or uncompressed from a nesting stratum, e.g.:

- A source and destination MAC or IP addresses of the datagram carrying SCHC datagrams
- A source and destination port number of the transport layer carrying SCHC datagrams
- A next header field
- An MPLS label
- A TLS Association
- Any other kind of connection id.

The Discriminator enables to determine the SCHC end-point that is used to decompress the SCHC Control Header, called a SCHC Control end-point.

Once uncompressed, the SCHC Control Header enables to determine the SCHC end-point, called a SCHC Data end-point, that is used to restore the datagram data that is compressed in the stratum.