SCHC Context Management Extensions

draft-corneo-schc-ctx-mgmt-00

Lorenzo Corneo, Magnus Westerlund

IPR disclosure

- This draft has the following IPR disclosures:
  - https://datatracker.ietf.org/ipr/7407/
  - https://datatracker.ietf.org/ipr/7406/
  - http://datatracker.ietf.org/ipr/7405/

Problem Statement

SCHC contexts face scalability issues in modern deployments:

- **Context growth**: more devices, payload compression → duplicated field descriptions, frequent updates
- **Rule matching latency**: larger rule sets → slower matching
- **Combinatorial explosion**: N address variants × M protocols = N×M rules with duplicated IPv6 fields
- **Variable headers**: IPv6 extension headers require a separate rule per combination

Proposed solution

1. **Rule Referencing CDAs** — static composition
   - `ref(N)`: reference another rule unconditionally
   - `ref-edit(N,M)`: reference with field overrides
2. **Rule Fragment Branching** — dynamic composition
   - `branch` CDA: select next rule fragment at de/compression time
   - `match-mapping` MO: branch on explicit header field values
   - `match-rule` MO: branch by probing candidate rules against payload

`ref(N)` CDA — Static Rule Composition

- `ref(N)` suspends current rule, applies Rule N, then resumes
- Eliminates duplication of shared protocol layers
- Example: UDP rule references shared IPv6 rule

FID	FL	FP	DI	TV	MO	CDA
IPv6 Rule		1	Bi		ignore	**ref(2)**
.....	..	..	..	...	...	.....

`ref-edit(N,M)` CDA — Reference with Overrides

`ref-edit(N,M)` extends ref(N) by modifying M field descriptions:

1. Load referenced Rule N
2. Replace M fields (matched by FID) with overrides from current rule
3. Apply the modified rule

Example: TCP rule uses `ref-edit(2,1)` to change IPv6 Next Header from 17 (UDP) to 6 (TCP)

FID	FL	FP	DI	TV	MO	CDA
IPv6 Rule		1	Bi		ignore	**ref-edit(2,1)**
IPv6 Next Header	8	1	Bi	6	equal	not-sent
.....	..	..	..	...	...	.....

`branch` CDA — Dynamic Rule Fragment Selection

- Encodes a selection among alternative rule fragments into the residual
  - FL > 0: branch on actual header field value (match-mapping)
  - FL = 0: branch on payload probing (match-rule)
  - branch(NULL): terminate further processing

branch CDA — match-mapping

- Used when it is possible to match a field value to TV, e.g., IP Next Header
- TV contains an array of values
- The first value matching triggers the use of the branched Rule ID

FID	FL	FP	DI	TV	MO	CDA	Sent
.....	..	..	..	.....	.....	.....	....
IPv6 Next Header	8	1	Bi	17	mapping	branch(3)	0b000
				6	mapping	branch(4)	0b001
				44	mapping	branch(5)	0b010
				60	mapping	branch(6)	0b011
				41	mapping	branch(7)	0b100
				59	mapping	branch(NULL)	0b101
.....	..	..	..	.....	.....	.....	....

`branch` CDA — `match-rule`

Used when no header field identifies what follows (e.g., UDP payload):

1. Save current packet location
2. Try each candidate rule's matching operations
3. First successful match → encode its index as residual
4. No match + no NULL entry → compression fails

Example: After UDP header, probe for RTP (Rule 8), CoAP (Rule 9), or fall back to NULL

FID	FL	FP	DI	TV	MO	CDA	Sent
....	..	..	..	..	.....	.....	....
UDP.PayProto	0	1	Bi		match-rule(8)	branch(8)	0b00
					match-rule(9)	branch(9)	0b01
					match-rule(NULL)	branch(NULL)	0b10
....	..	..	..	..	.....	.....	....

Security Considerations

- **Circular references**: implementations **MUST** validate rule graph is a DAG
- **Resource exhaustion**: enforce limits on reference depth, mapping table size, match-rule candidates, and processing time
- **Context integrity**: provisioning/update mechanisms **MUST** ensure authenticity and integrity of rule definitions

THANK YOU! FEEDBACK?