draft-ietf-lpwan-ipv6-static-context-hc-17

Authors:
Laurent Toutain <Laurent.Toutain@imt-atlantique.fr>
Carles Gomez <carlesgo@entel.upc.edu>
Ana Minaburo <ana@ackl.io>
Dominique Barthel <dominique.barthel@orange.com>
Juan Carlos Zuniga <JuanCarlos.Zuniga@sigfox.com>

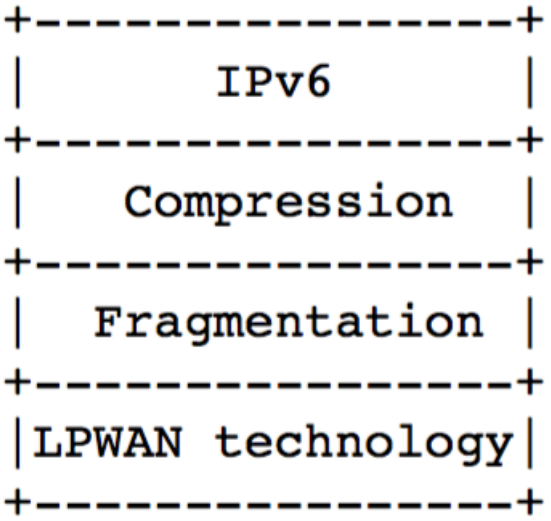
Presentation agenda

- What is this draft about?
- What has happened since IETF102?
- Ticket status
- What is coming up next?
- Fragmentation section re-structuring
- ACK-on-Error fragmentation mode (Laurent)

What is this draft about?

3 deliverables of this draft

- Specification of a Header Compression engine (Section 7)
  - Generic engine, uses Static Context (-> SCHC)
- Specification of UDP/IPv6 compression (Section 10)
  - Using this SCHC engine
- Specification of a fragmentation protocol (Section 8)
  - Has 3 different "modes" described in this draft
  - Different modes address different requirements



What has happened since IETF102?

What has happened since IETF102?

- Focused mostly on fragmentation
- Designed new ACK-on-Error fragmentation mode
  - Analysis side meeting in Montreal
  - Design sessions
  - Interim meetings
- Extensively edited Fragmentation section

Changes in the draft, by sections (1/2)

- Abstract, Intro (Section 1)
  - Some text improvement
  - Introduction of Profile
  - Removal of no Out-of-Order delivery assumption, deferred to Fragmentation section
- Terminology (Section 4)
  - Removed Fragmentation terminology, moved to Frag. section
  - Profile added
- SCHC overview (Section 5)
  - Removed SCHC Fragment and SCHC ACK messages format
- Compression/Decompression (Section 7)
  - Some text improvement

Changes in the draft, by sections (2/2)

- Fragmentation/Reassembly (Section 8)
  - Restructured (tools, messages formats, modes)
  - New ACK-on-Error mode
- UDP/IPv6 compression (Section 10)
  - Mentions ECN bits
  - Fixed a few errors, text improvements
- Fragmentation examples (Appendix B) updated
- Fragmentation State Machine drawings (Appendix C) updated
- Parameters (Appendix D) restructured and updated

Hackathon at IETF103

- 10 contributors
- New GitHub project
  - https://github.com/openschc
  - Python3/uPython
  - Arch, interfaces, code
  - Test scenarios
- Thorough reading of draft
  - Better understanding
  - Feedback
  - More work items to WG

Ticket status

Tickets

- All Tickets by the LPWAN WG https://trac.ietf.org/trac/lpwan/report/6
- Selective link to Tickets pertaining to *this* draft ipv6-schc-all-tickets
- Ticket #23 : optional MIC?
  - resulted from IETF102 discussion
  - CLOSED: MIC is mandatory in this specification
  - Message formats allow other SDOs to reuse SCHC Fragmentation and dispense with MIC
- Ticket #23 : description of MIC computation
  - CLOSED: new text.
- No OPEN Ticket on this draft at this time

What is coming up next?

What is coming up next?

- Feedback from Hackathon implementers
  - Assumption on All-1 SCHC Fragment format to be written down in a MUST statement
- Chairs to launch WGLC on Fragmentation section?
  - Review by Charlie Perkins already in progress
- Implementation of ACK-on-Error fragmentation in progress
  - At least one private and one Open Source project (Hackathon)
- Presentation of ACK-on-Error fragmentation to LoRa Alliance in two weeks

Fragmentation section re-structuring

New section layout

- Tools
- Message formats
- Algorithms (Frag "modes")
  - No-ACK
  - ACK-Always
  - ACK-on-Error (new)

Tiles, windows of tiles, bitmaps

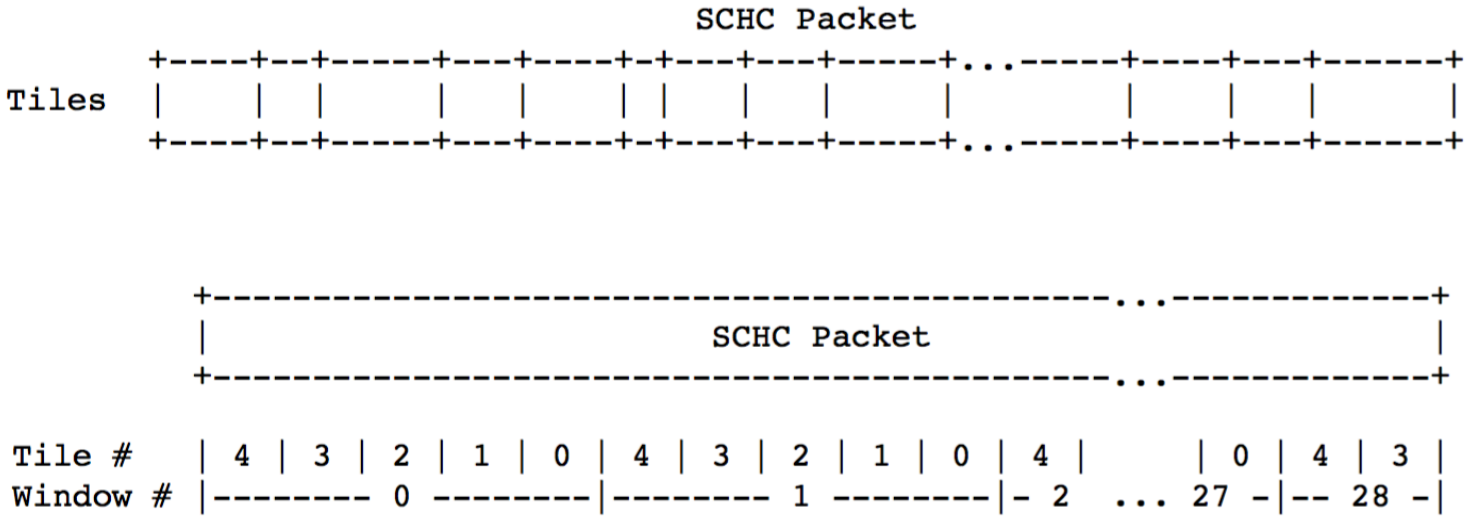
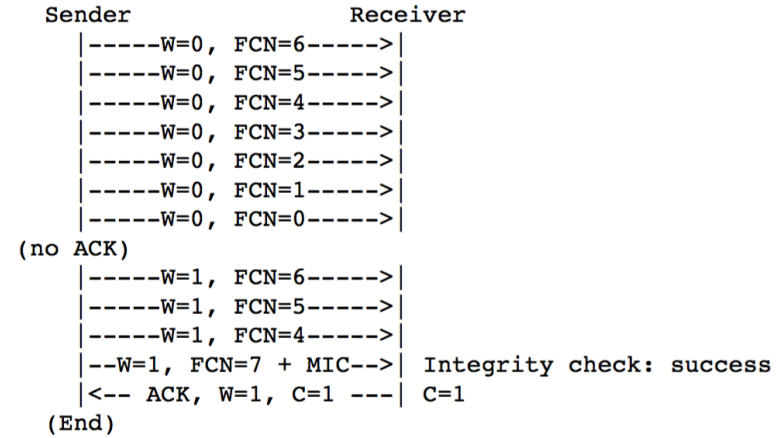
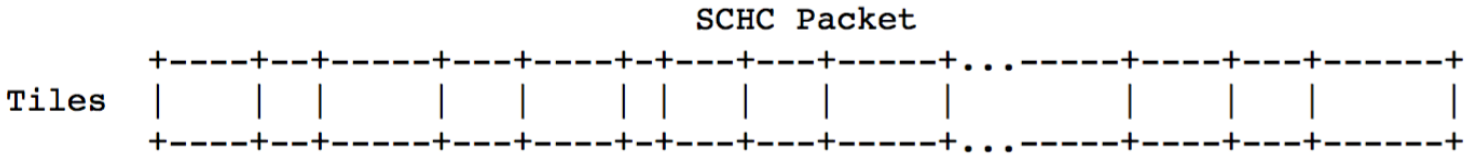


Figure 9: a SCHC Packet fragmented in tiles grouped in 28 windows, with WINDOW_SIZE = 5

Tiles, windows of tiles, bitmaps





ACK-on-Error fragmentation mode

ACK on Error

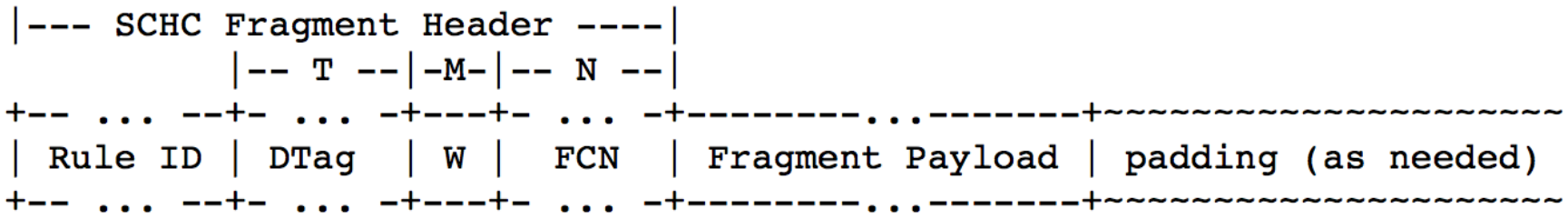
- Goal:
  - Reduce the number of ACK messages
  - Optimize downlink (assuming dominant uplink data traffic)

- Method:
  - Don't acknowledge windows that are fully received
  - In the best case, only one ACK at the end (All-1)
    - Confirms that the receiver has correctly received the full packet

Ack-on-Error at IETF102 meeting

- W field size was 1 bit
  - Used the same message format as Ack-Always mode
- Led to ambiguities when two consecutive ACKs were lost
  - Very complex State Machine.
- Solution:
  - Open the window: increase the W (window) field to several bits.
  - Each tile of the SCHC Packet is uniquely identified through W/FCN values.
  - Good property: relaxed synchronization between the sender and the receiver.

W field size ?



- Worst case:
  - Tile size: 6 bytes
  - Packet size: 1280 bytes
  - FCN size: 3 bits (7 tiles per window)
    - 214 tiles, 31 windows
- Window number needs 5 bits
  - M is 5 bits
- Have different rules: one for small packets and one for largest packets?

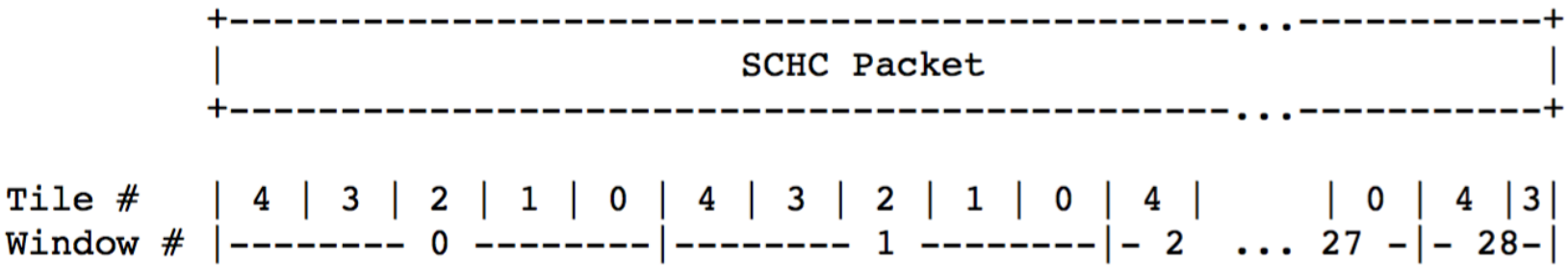
Relaxed synchronization

- Since tiles are uniquely identified, Acknowledgment strategy is more flexible
  - Ack at the end of a window (like in Ack-Always mode)
  - Ack at the end of the SCHC Packet
  - Ack at other times (slotted network)
- Ack strategy must be defined in the Profile

- State Machine is simplified:
  - Sender sends again tiles marked by a bitmap until MIC OK
  - Receiver sends ACKs with bitmaps for incomplete windows, or final ACK (MIC OK)

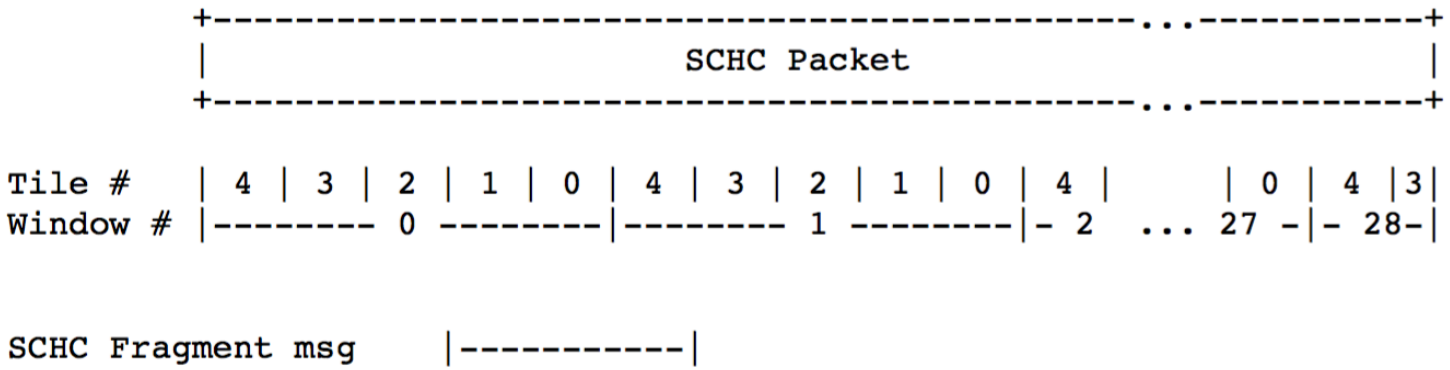
Variable MTU – Tiles (1/2)

- In ACK-on-Error mode, tiles have a fixed size
- SCHC Fragments messages transport tiles
- If a SCHC Fragment message contains only one tile
  - The W/FCN fields give the tile absolute position (window # / tile #)
  - The tile size may be adapted so that the Fragment message avoids padding



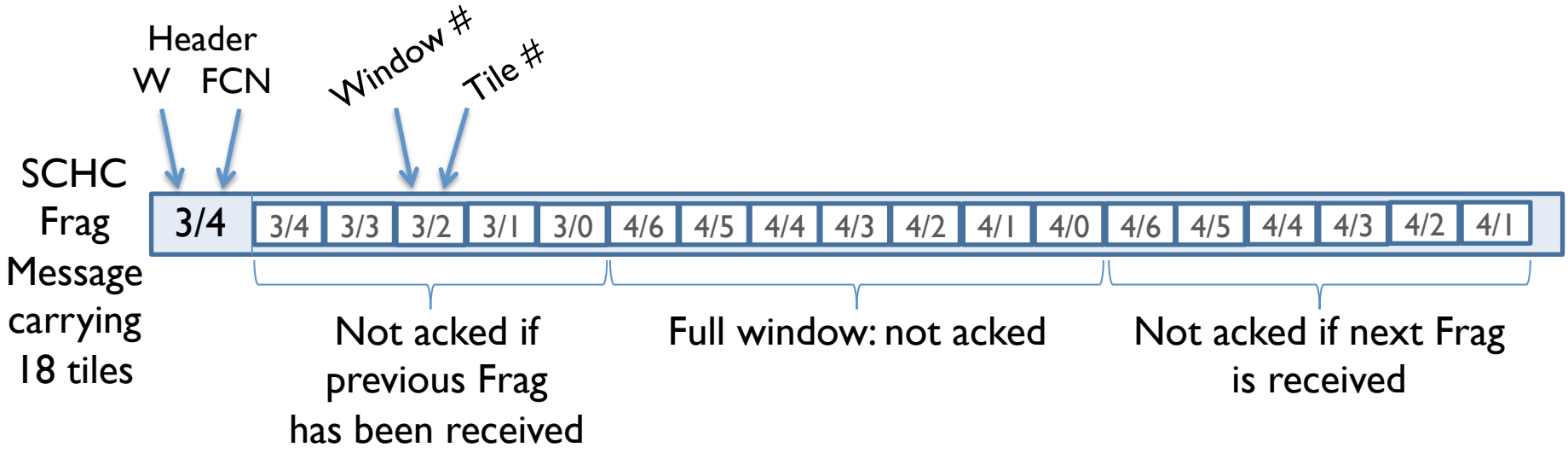
Variable MTU – Tiles (2/2)

- If a SCHC Fragment message contains several tiles
  - W/FCN gives the absolute position of the first tile,
    - The others are numbered given their position in the Fragment message
    - Can span windows
  - Tile size is adapted to fit several MTU sizes
    - For instance in LoRaWAN: 11, 33, 53, 125, 222, 242, 242 bytes -> 8 bytes tiles

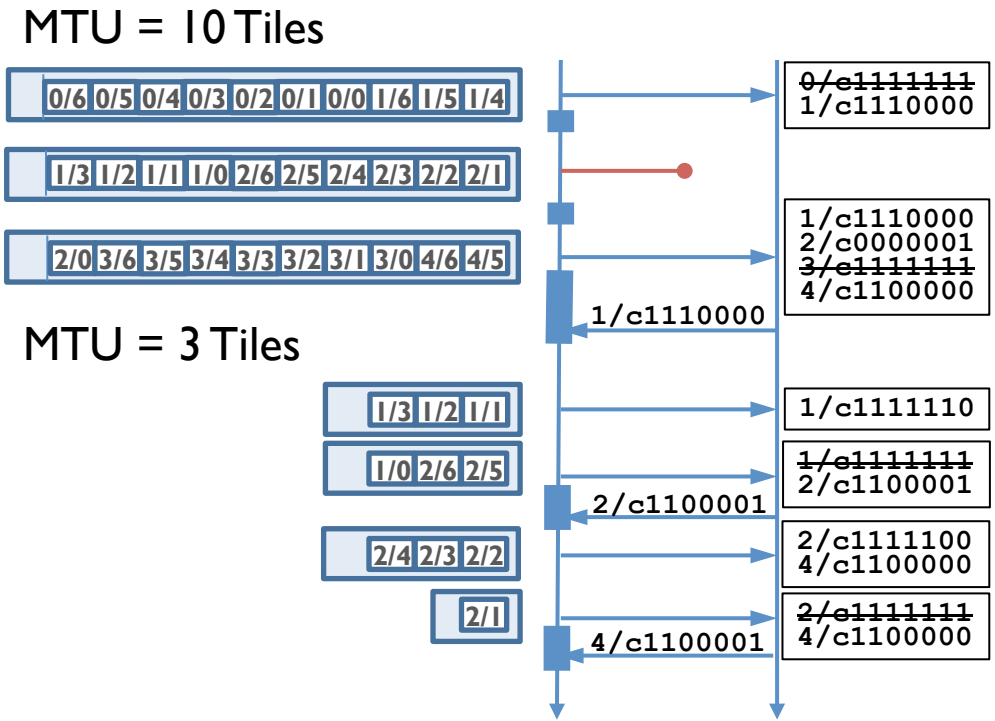


No impact on Ack

- Only incomplete windows lead to Ack
  - Full windows are not acked



When MTU changes



- Profile:
  - Define if Ack is possible after each Tile# 0 is sent
- a SCHC Fragment message carrying a single tile must fit in the smallest MTU
- There must be a separate All-1 Fragment message:
  - MIC only or MIC+Tile

Conclusion

- Pros:
  - Sender/receiver relaxed synchronization, simpler State Machine
  - Reduced number of ACK messages
    - A least 1
    - Exact number is function of the error rate
  - Allows MTU variation
- Cons:
  - Slightly larger message header (W field)
  - Exact ACK policy must be defined in Profile
  - Potentially more padding bits per SCHC Packet (if var. MTU)
  - Downlink fragmentation

Thank you for your attention