6lo Fragmentation DT

Thomas Watteyne (Chair)
Carsten Bormann
Rahul Jadhav
Gorry Fairhurst
Pascal Thubert
Gabriel Montenegro

Context & Agenda

- pre-IETF99: fragmentation comes up regularly in 6lo WG meetings
- IETF99: 6lo chairs ask for volunteers to create a DT
- IETF100: DT meets
- IETF101:

• Problem Statement & Goal	(Thomas Watteyne)	10 min
• draft-watteyne-6lo-minimal-fragment	(Carsten Bormann)	10 min
• draft-thubert-6lo-forwarding-fragments	(Pascal Thubert)	10 min
• Q&A		10 min

Problem Statement & Goal

6lo Fragmentation DT

Thomas Watteyne

Outline

- Standardized per-hop reassembly solutions
  - RFC4944
  - RFC6282
- Problem statement
- Candidate fragment forwarding solutions
  - Carsten’s book
  - draft-thubert-6lo-forwarding-fragments-08
- Goal of the DT

RFC4944

- Link-layer fragmentation only in route-over → reassembly at ea op
- Fragment header

Units of 8 octets

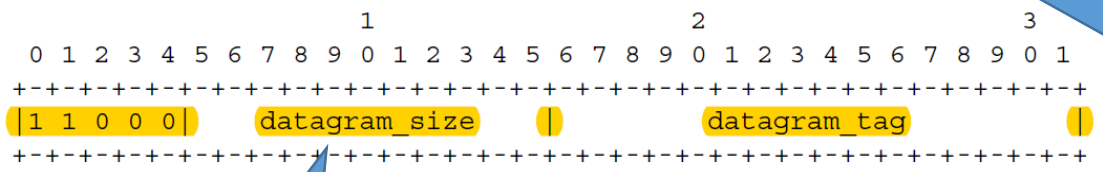


Figure 4: First Fragment

bytes

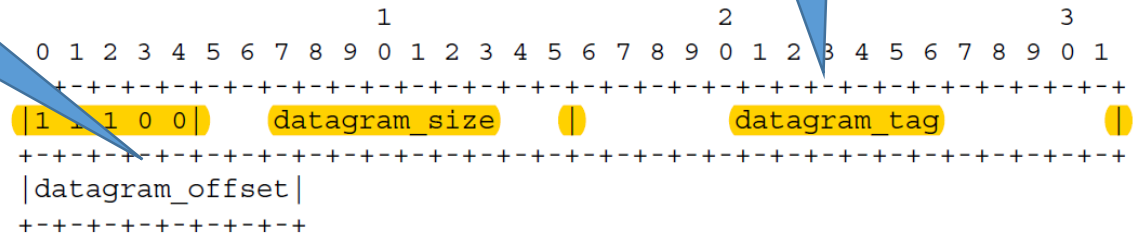


Figure 5: Subsequent Fragments

- +1 on each new frag
- No initial value specified

- Reassembly timer:
  - Starts when node receives first fragment
  - Timeout value MUST be <60s
  - When times out, buffer cleared, packet dropped

Pattern	Header Type	
00 xxxxxx	NALP	- Not a LoWPAN frame
01 000001	IPv6	- Uncompressed IPv6 Addresses
01 000010	LOWPAN_HC1	- LOWPAN_HC1 compressed IPv6
01 000011	reserved	- Reserved for future use
...	reserved	- Reserved for future use
01 001111	reserved	- Reserved for future use
01 010000	LOWPAN_BC0	- LOWPAN_BC0 broadcast
01 010001	reserved	- Reserved for future use
...	reserved	- Reserved for future use
01 111110	reserved	- Reserved for future use
01 111111	ESC	- Additional Dispatch byte follows
10 xxxxxx	MESH	- Mesh Header
11 000xxx	FRAG1	- Fragmentation Header (first)
11 001000	reserved	- Reserved for future use
...	reserved	- Reserved for future use
11 011111	reserved	- Reserved for future use
11 100xxx	FRAGN	- Fragmentation Header (subsequent)
11 101000	reserved	- Reserved for future use
...	reserved	- Reserved for future use
11 111111	reserved	- Reserved for future use

Figure 2: Dispatch Value Bit Pattern

RFC6282

Section 5.3 of [RFC4944] also defines how to fragment compressed IPv6 datagrams that do not fit within a single link frame. Section 5.3 of [RFC4944] defines the fragment header's datagram_size and datagram_offset values as the size and offset of the IPv6 datagram before compression. As a result, all fragment payload outside the first fragment must carry their respective portions of the IPv6 datagram before compression. This document does not change that requirement. When using the fragmentation mechanism described in Section 5.3 of [RFC4944], any header that cannot fit within the first fragment MUST NOT be compressed.

Problem statement

- Per-hop fragmentation and reassembly has 2 issues:
  - Latency:
    - Increases end-to-end latency as you need to wait for each fragment at each hop
  - Reliability:
    - Limited memory → limited number of buffers (1-2?) → packet dropped when new frag received and old not fully reassembled yet
    - No frag recovery: 1 frag loss == packet dropped
- Proposed solution:
  - Fragment forwarding:
    - Source fragments
    - Intermediate nodes relays
    - LBR reassembles

6LoWPAN: The Wireless Embedded Internet

a.k.a. “Carsten’s book”

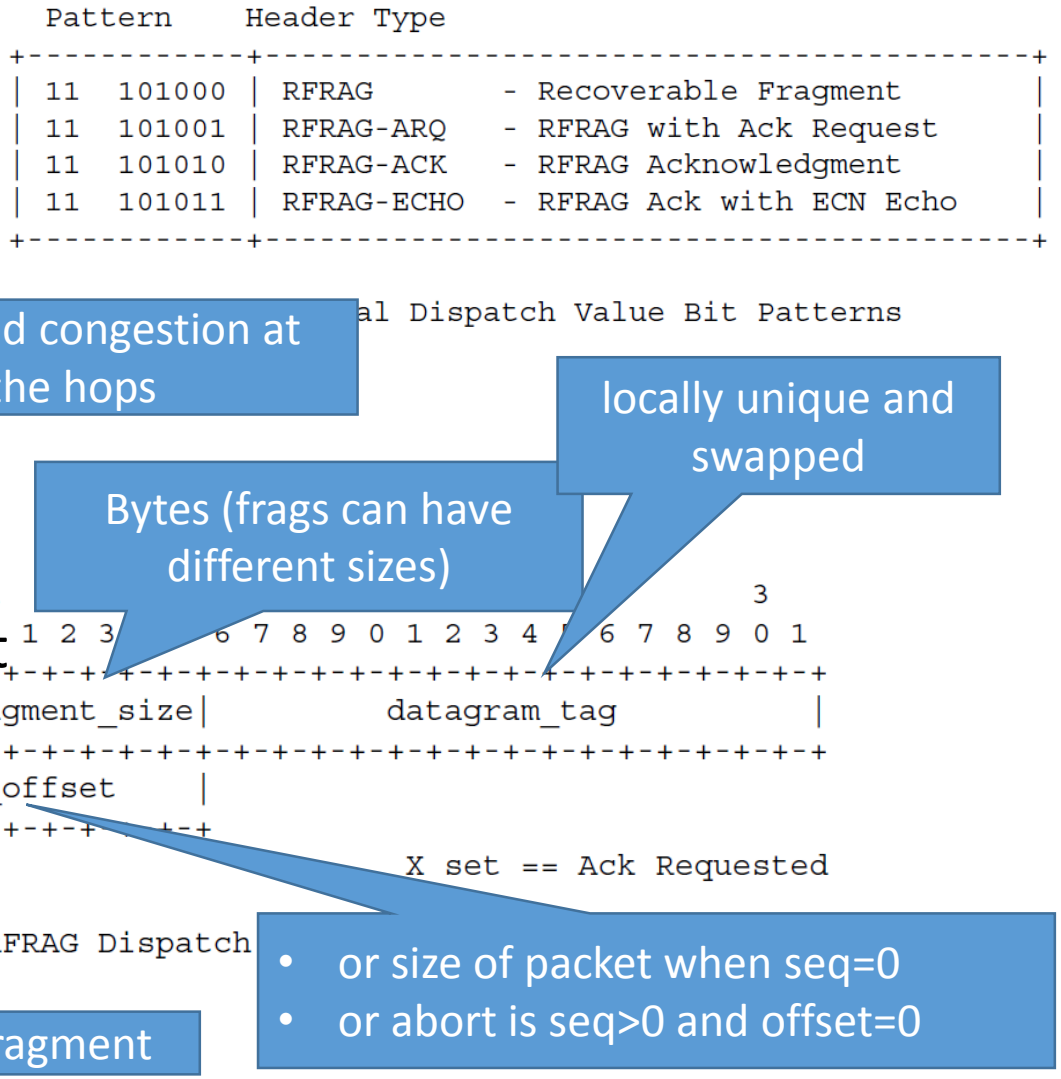
2.5.2 L3 routing (“Route-Over”)

Layer-3 Route-Over forwarding is illustrated in Figure 2.6. In contrast to layer-2 mesh forwarding, layer-3 Route-Over forwarding does not require any special support from the adaptation layer format. Before the layer-3 forwarding engine sees the packet, the adaptation layer has done its work and decapsulated the packet – at least conceptually (implementations may be able to perform some optimizations by keeping the encapsulated form if they know how to rewrite it into the proper encapsulated form for the next layer-3 hop).

Note that this in particular means that fragmentation and reassembly are performed at each hop in Route-Over forwarding – it is hard to imagine otherwise, as the layer-3 addresses are part of the initial bytes of the IPv6 header, which is present only in the first fragment of a larger packet. Again, implementations may be able to optimize this process by keeping virtual reassembly buffers that remember just the IPv6 header including the relevant addresses (and the contents of any fragments that arrived out of order before the addresses).

draft-thubert-6lo-forwarding-fragments-07

- Fragment forwarding
  - Locally unique label, swapped at each hop
- End-to-end ACK
  - ACK requested by source
    - for any fragment
  - ACK travels reverse LSP
- Fragment recovery
  - bitmap in RFRAG-ACK, one bit per fragment
- Flow control capabilities
- Different size per frag



Goal of the DT

- Produce 2 documents (to be submitted to 6lo WG):
  - informational document
    - summarize fragmentation as standardized now
    - describes Carsten's virtual reassembly buffer implementation
    - discusses its limits
  - standards-track document
    - builds upon the first one
    - adds fragment recovery
- Philosophy
  - keep activity as swift as possible
  - small DT, but regular information to WGs
  - ideally close the DT after London *(to be discussed)*

draft-watteyne-6lo-minimal-fragment-00

Thomas Watteyne

Carsten Bormann

Pascal Thubert

ToC

context

VRB

Context: typical fragmentation implementation

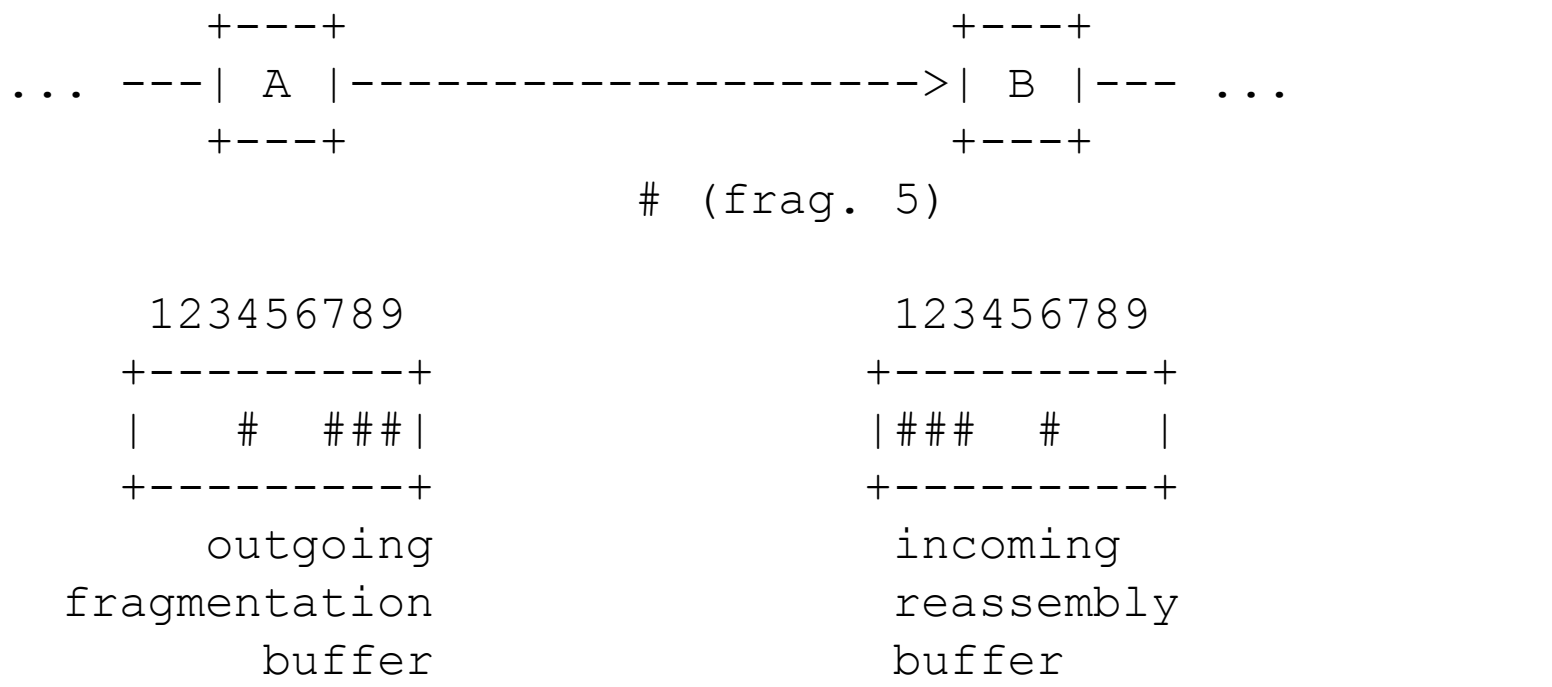
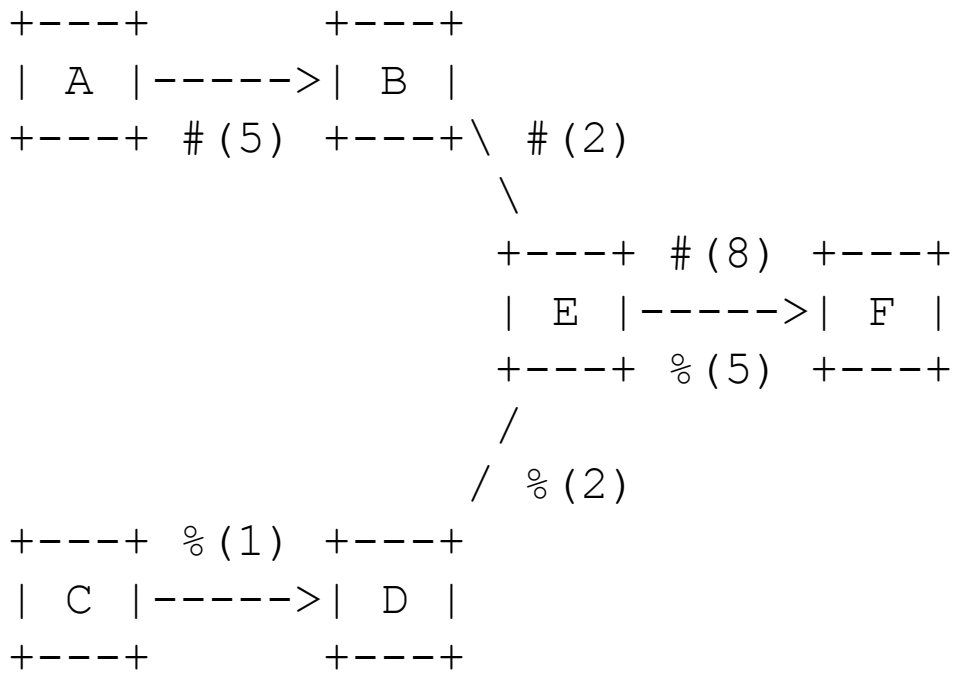


Figure 1: Fragmentation at node A, reassembly at node B.

→ Limits: latency, end-to-end reliability *(see preliminary simulation results)*

Virtual Reassembly Buffer (VRB) Implementation



incoming		outgoing	
L2 src	tag	L2 dest	tag
B	2	F	8
D	2	F	5
empty			
empty			

Node E's VRB table.

Figure 3: Illustrating VRB. #(5) and %(1) are fragments from packets coming from nodes A and C, with datagram_tag set to 5 and 1, respectively.

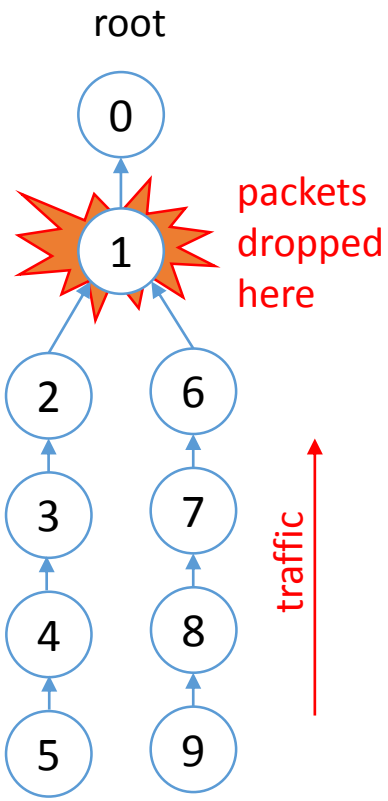
VRB: Gotchas and Limits

- Gotchas
  - Long headers (e.g. source routing) → receive multiple fragments before forwarding
  - Out-of-order fragments → receive multiple fragments before forwarding
  - Changing header length → put all slack in the frame sizes into the _first_ fragment
- Security
  - DoS attack by “fragment 1” flood to overflow VRB
- Limits
  - Non-zero Packet Drop Probability
  - No Fragment Recovery
  - No Per-Fragment Routing

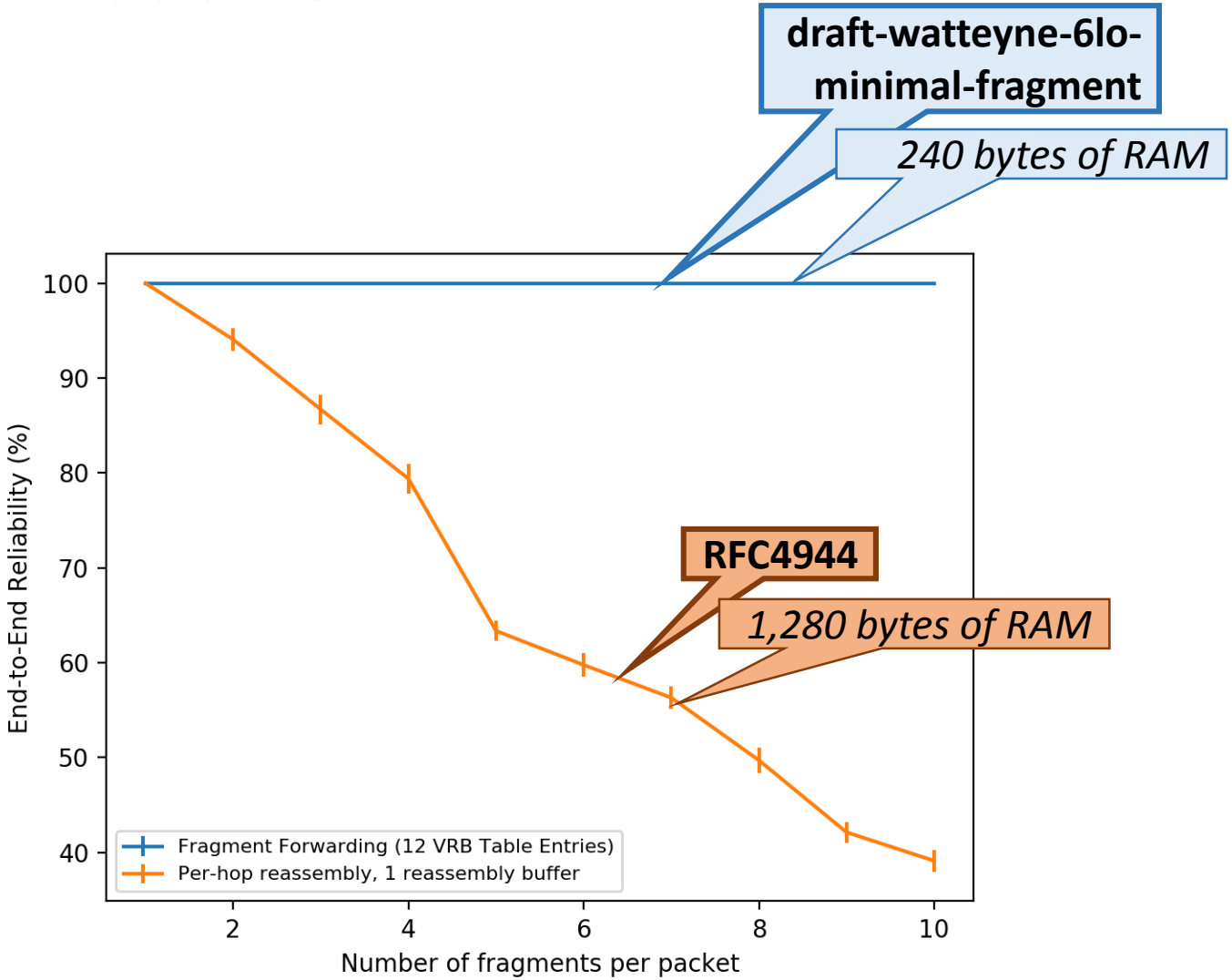
→ If limits are not acceptable you need an actual protocol (such as draft-thubert-6lo-fragment-recovery)

Preliminary Simulation Results

Yasuyuki Tanaka



- Using 6TiSCH simulator (https://bitbucket.org/6tisch/simulator/src)
- topology shown on the left
- RFC8180 with 101 slot slotframe, sufficient bandwidth, no 6P, no RPL
- all nodes generate data pkPeriod = U[54s,66s]
- One data point = 100 runs
- 95% confidence intervals



6lo vs. lwig

- same content was first published as draft-bormann-lwig-6lowpan-virtual-reassembly
- Then, IETF IoT Directorate call and follow-up e-mails with 6lo chairs
- Plan is to:
  - submit both informational and standards-track drafts to 6lo
  - Present at IETF 101
  - Get reviews on both from the 6lo WG
  - Discuss which WG final version belongs to

6LoWPAN Selective Fragment Recovery

draft-thubert-6lo-fragment-recovery-00

P.Thubert

IETF 101

London

Features

- New formats for the fragment header
- Selective Fragments Recovery
  - Expects but does not depend on IOD
- Window-based Flow Control
  - ACK at the end of the window
- Explicit Congestion Notification
  - ECN flag echoed to the source
- Explicit Signaling to both set up and clean up
  - Including Abort and Fin

Status

- Draft -00
  - Based on draft-thubert-6lo-forwarding-fragments
  - Same operation as / limited diffs from it: https://tools.ietf.org/rfcdiff?url2=draft-thubert-6lo-fragment-recovery-00.txt

- Summary of the changes
  - Removed description of the forwarding, same as VRB
  - Complements draft-watteyne-6lo-minimal-fragment
  - Introduces the concept of 2 LSPs for one VRB

Past IETF presentation

P.Thubert

IETF

Prague

History

- Presented 6lo Fragmentation issues in Chicago
  - In appendix of this slideware
  - Mostly issues for route-over
  - Summarized in next slide

- Work on fragmentation at LPWAN
  - As part of the SCHC IP/UDP draft
  - Optional: Windowing/individual retry of fragments
  - Does not need to support multihop

Context

- TCP rarely used,
  - Pro is MSS to avoid fragmentation
- 6LoWPAN applications handle their reliability
  - UDP
  - to get exactly what they need
  - They also expect very long round trips.
- Time gained by streamlining fragments is available for retries without a change in the application behavior.

6lo Route-Over fragmentation issues

- Recomposition at every L3 hop
  - Cause latency and buffer overutilization
- Uncontrolled sending of multiple fragments
  - Interferences in single frequency meshes
- Fragment flows interfere with one another
  - Buffer bloat / congestion loss
- Loss locks buffers on receiver till time out
  - Readily observable, led to RFC 7388

6lo Fragmentation reqs

- Provide Fragment Forwarding
  - There are pitfalls, better specify one method
  - E.g. datagram tag switching ala MPLS
  - Stateful => state maintenance protocol
- Provide pacing/windowing capabilities
  - Mesh awareness? (propagation delay, nb hops)
- Provide fragment reliability
  - individual ack/retry/reset, e.g. ala SCHC
- Provide congestion control for multihop
  - E.g. ECN

Path Forward

- Solutions exist (as shown by draft-thubert..):

1. Produce a problem statement at 6lo
   - Based on this slideware
2. Form a design team
   – Need TSV skills to solve the problem
   – Also MPLS and radio skill, CoAP, CoCoA
3. Find a host WG and produce a std track
   – at TSVWG?
4. Also recommendations for application design

APPENDIX

Backup slides
The problem with fragments in 6lo mesh networks

P.Thubert

IETF 99

Prague

draft-thubert-6lo-forwarding-fragments-04

Recomposition at every hop

- Basic implementation of RFC 4944 would cause reassembly at every L3 hop
- In a RPL / 6TiSCH network that's every radio hop
- In certain cases, this blocks most (all?) of the buffers
  - Buffer bloat
- And augments latency dramatically

Research was conducted to forward fragments at L3.

Early fragment forwarding issues #1

- Debugging issues due to Fragments led to RFC 7388
- Only one full packet buffer
- Blocked while timing out lost fragments
- Dropping all packets in the meantime
- Arguably there could be implementation tradeoffs
  - but there is no good solution with RFC4944,
  - either you have short time outs and clean up too early,
  - or you lose small packets in meantime

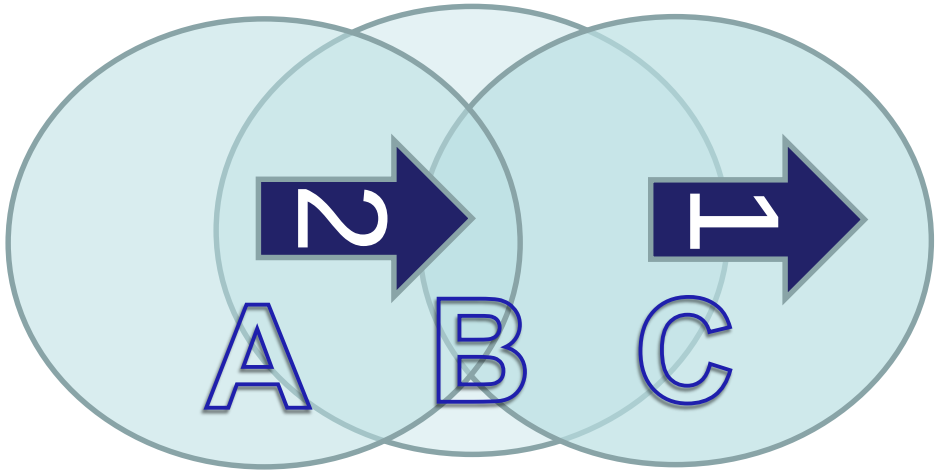
Early fragment forwarding issues #1 c’d

- Need either to abandon fragmented packet
- or discover loss and retry quickly, both need signaling
- Solution is well-know:
  - selective acknowledgement
  - reset
- Requires new signaling

=> Implementation recommendations are not sufficient

Early fragment forwarding issues #2

- On a single channel multihop network (not 6TiSCH):
  Next Fragment interferes with previous fragment
- No end-to-end feedback loop
- Blind throttling can help
- New signaling can be better



Deeper fragment forwarding issues #3

- More Fragments pending than hops causes bloat
- No end-to-end feedback loop for pacing
- Best can do is (again) blind throttling
- Solution is well-known, called dynamic windowing
- Need new signaling

=> Implementation recommendations are not sufficient

Deeper fragment forwarding issues #4

- Multiple flows through intermediate router cause congestions
- No end-to-end feedback for Congestion Notification.
- Blind throttling doesn’t even help there
- Fragments are destroyed, end points time out, packets are retried, throughput plummets
- Solution is well-known, called ECN
- Need new signaling

=> Implementation recommendations are not sufficient

Deeper fragment forwarding issues #5

- Route over => Reassembly at every hop creates a moving blob per packet
- Changes the statistics of congestion in the network
- Augments the latency by preventing streamlining
- More in next slides

=> Need to forward fragments even in route over case

Current behaviour

	Sender	Router 1	Router 2	Receiver
T=0	III			
T=1	II(I)	I		
T=2	I(I)	II		
T=3	(I)	III		
T=4		II(I)	I	
T=5		I(I)	II	
T=6		(I)	III	
T=7			II(I)	I
T=8			I(I)	II
T=9			(I)	III

Window of 1 fragment

	Sender	Router 1	Router 2	Receiver
T=0	III			
T=1	II(I)	I		
T=2	II	(I)	I	
T=3	II		(I)	I
T=4	I(I)	I		I
T=5	I	(I)	I	I
T=6	I		(I)	II
T=7	(I)	I		II
T=8		(I)	I	II
T=9			(I)	III

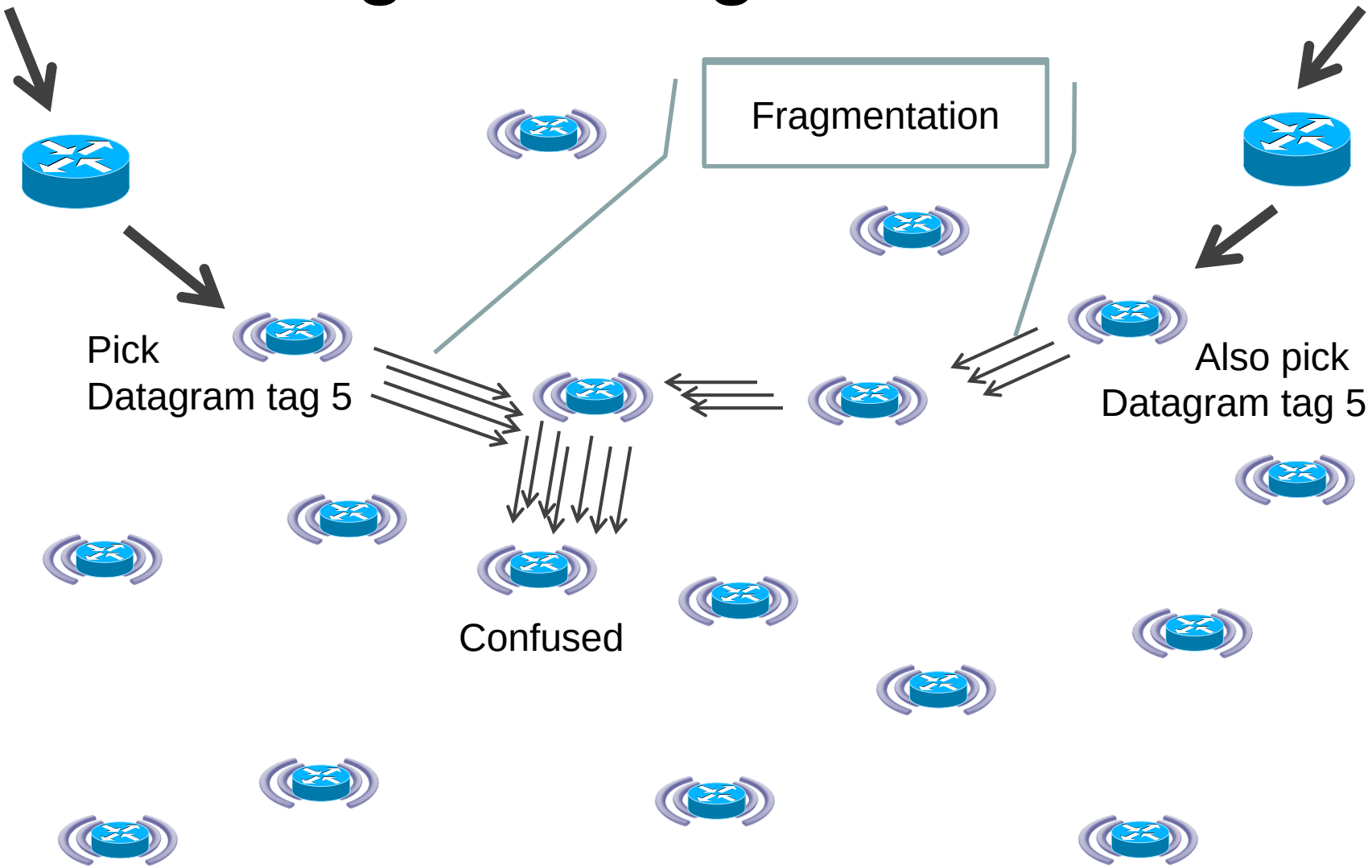
Streamlining with larger window

	Sender	Router 1	Router 2	Receiver
T=0	III			
T=1	II(I)	I		
T=2	II	(I)	I	
T=3	I(I)	I	(I)	I
T=4	I	(I)	I	I
T=5	(I)	I	(I)	II
T=6		(I)	I	II
T=7			(I)	III
T=8				
T=9				

Even Deeper fragment forwarding issues #6

- Original datagram tag is misleading
- Tag is unique to the 6LoWPAN end point
- Not the IP source, not the MAC source
- 2 different flows may have the same datagram tag
- Implementations storing FF state can be confused
- Solution is well known, called label swapping
- An easy trap to fall in, need IETF recommendations

Datagram Tag Confusion



Even Deeper fragment forwarding issues #6

- Forwarding Fragments requires state in intermediate nodes
- This state has the same time out / cleanup issues as in the receiver end node
- Solution is well known: Proper cleanup requires
  - signaling that the flow is completely received
  - or reset

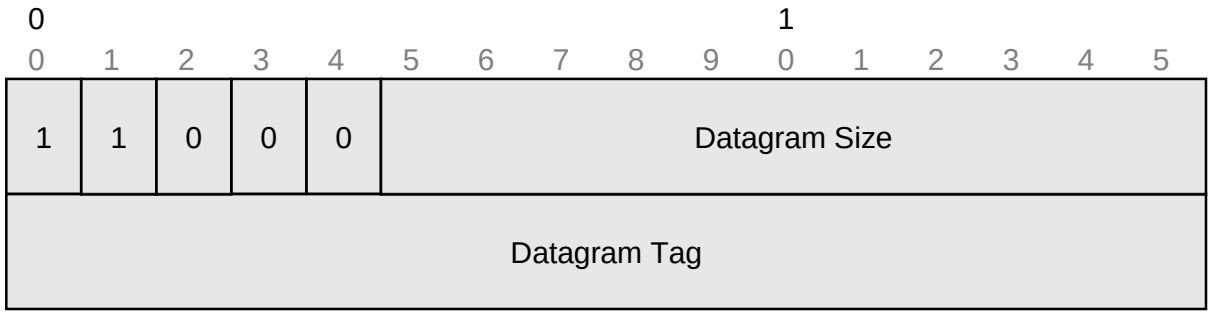
Conclusion

- People are experiencing trouble that was predictable from the art of Internet and Switching technologies
- The worst of it (collapse under load and hard-to-debug misdirected fragments) was not even seen yet but is predictable
- Some issues can be alleviated by Informational recommendations
- Some require a more appropriate signaling
- Recommendation is rethink 6LoWPAN fragmentation

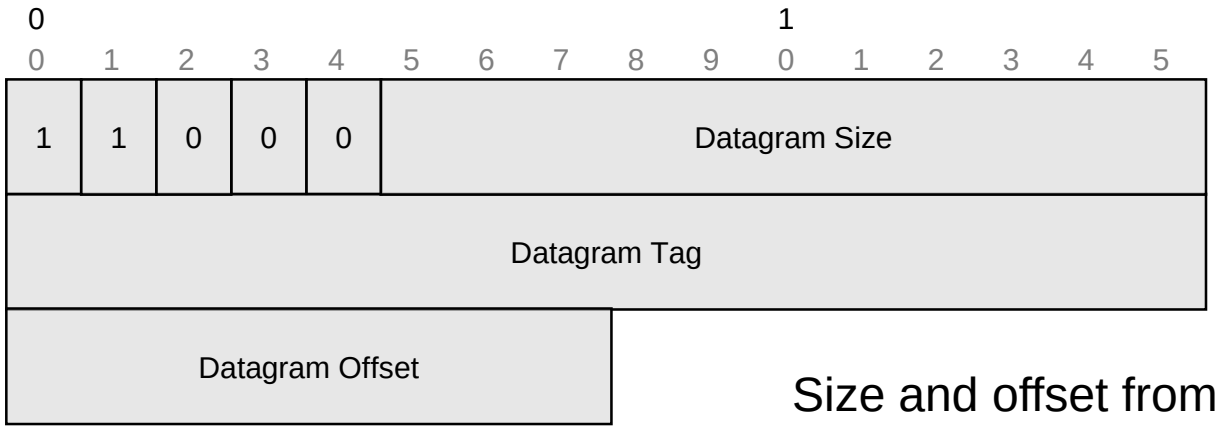
draft-thubert-6lo-forwarding-fragments

- Provides Label Switching
- Selective Ack
- Pacing and windowing + ECN
- Flow termination indication and reset
- Yes it is transport within transport (usually UDP)
- Yes that is architecturally correct because fragment re-composition is an endpoint function
- And No splitting the draft is not appropriate, because the above functionalities depend on one another.

RFC 4944: 6LoWPAN Fragmentation



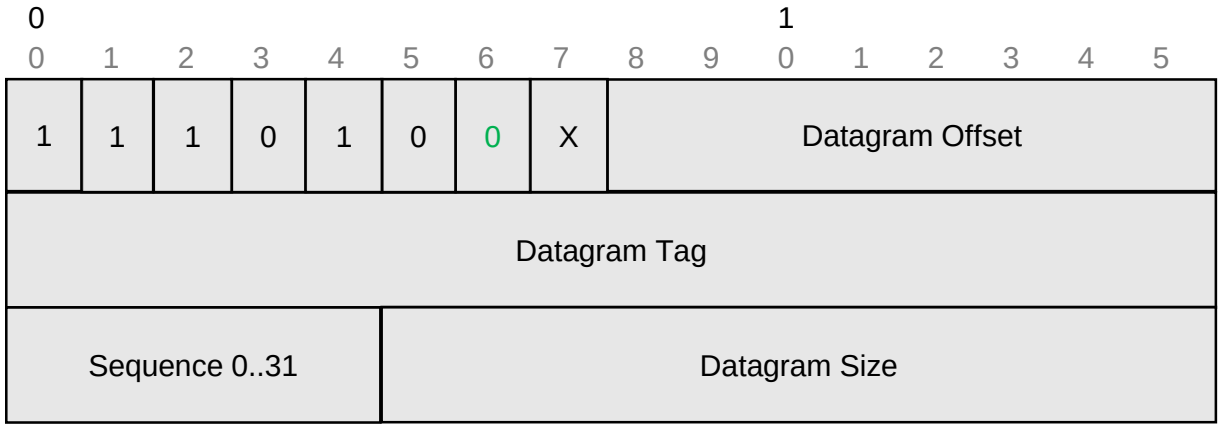
1st fragment



Next fragments

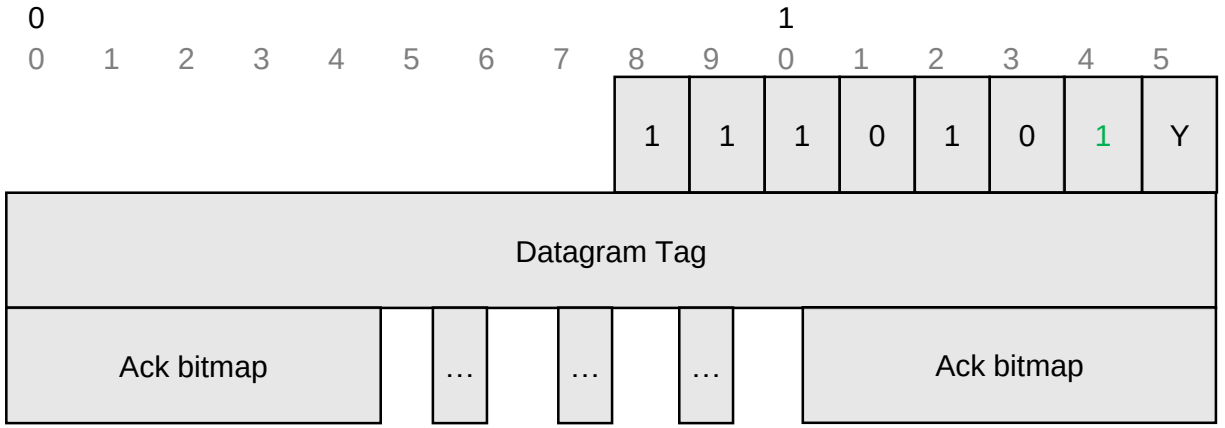
Size and offset from uncompressed form
1-hop technology

draft-thubert-6lo-forwarding-fragments



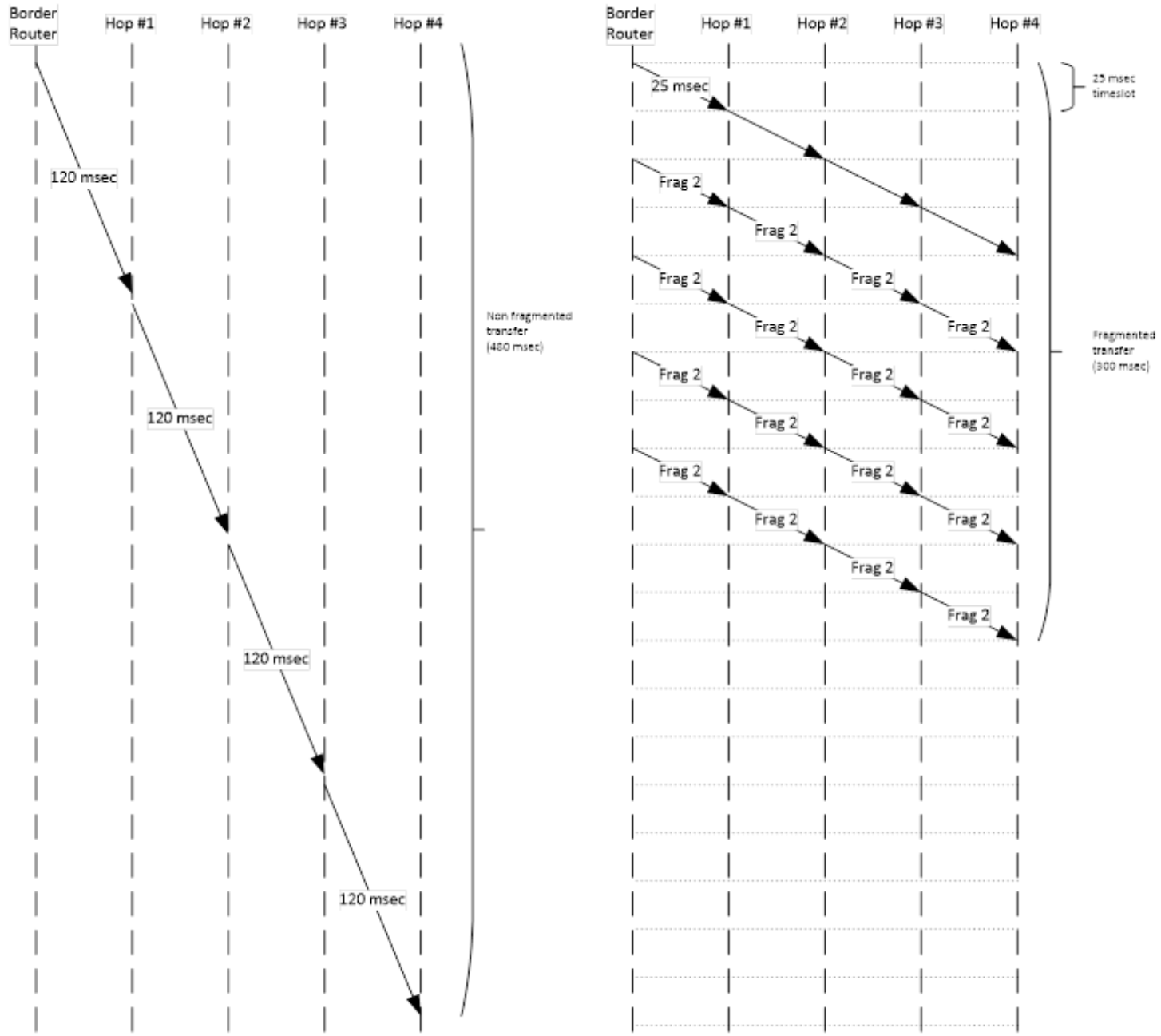
fragment
X <= ack request

Size and offset from compressed form



ACK
Y <= ECN

multi-hop technology


Border Router
Hop #1
Hop #2
Hop #3
Hop #4
120 msec
120 msec
120 msec
120 msec
Non fragmented transfer (480 msec)
Border Router
Hop #1
Hop #2
Hop #3
Hop #4
25 msec
25 msec timeslot
Frag 2
Fragmented transfer (300 msec)

Current behaviour

	Sender	Router 1	Router 2	Receiver
T=0	III			
T=1	II(I)	I		
T=2	I(I)	II		
T=3	(I)	III		
T=4		II(I)	I	
T=5		I(I)	II	
T=6		(I)	III	
T=7			II(I)	I
T=8			I(I)	II
T=9			(I)	III

Single fragment

	Sender	Router 1	Router 2	Receiver
T=0	III			
T=1	II(I)	I		
T=2	II	(I)	I	
T=3	II		(I)	I
T=4	I(I)	I		I
T=5	I	(I)	I	I
T=6	I		(I)	II
T=7	(I)	I		II
T=8		(I)	I	II
T=9			(I)	III

Streamlining

	Sender	Router 1	Router 2	Receiver
T=0	III			
T=1	II(I)	I		
T=2	II	(I)	I	
T=3	I(I)	I	(I)	I
T=4	I	(I)	I	I
T=5	(I)	I	(I)	II
T=6		(I)	I	II
T=7			(I)	III
T=8				
T=9				