6LoWPAN Selective Fragment Recovery

- P. Thubert

IETF 104

Prague

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 -02
  - Limited reorg, terminology first
  - Discussion on the needed slack in the first fragment and how to compute it
  - Discussion modifying the first fragment which impacts the datagram size
  - New formats

About Slack in the 1st frag.

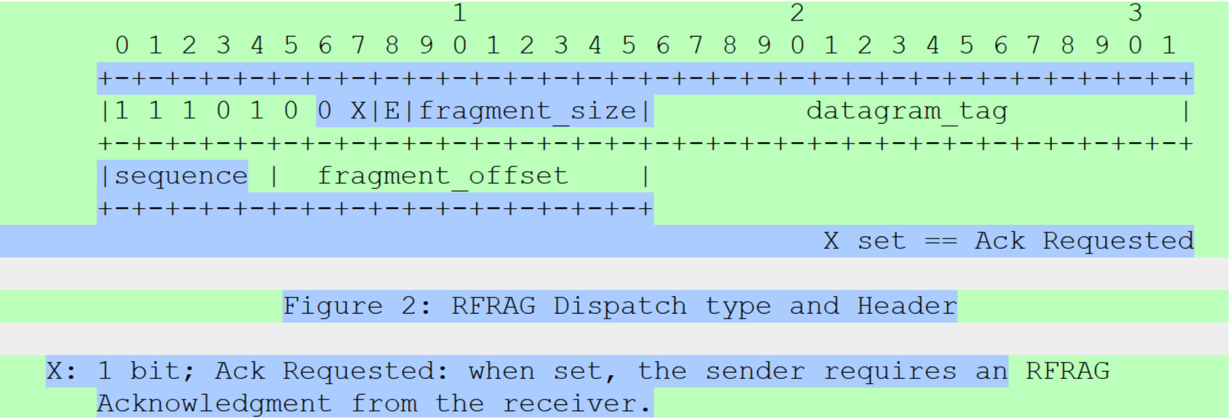
“For instance, if the IID of the source IPv6 address is elided by the originator, then it MUST compute the fragment_size as if the MTU was 8 bytes less. This way, the next hop can restore the source IID to the first fragment without impacting the second fragment.”

About changing the size for 1st frag.

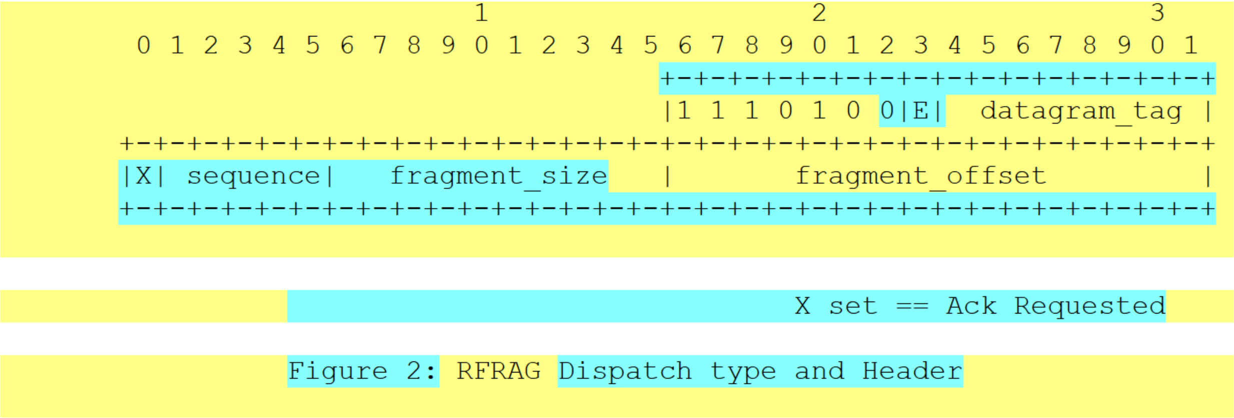
“If the size of the first fragment is modified, then the intermediate node MUST adapt the datagram_size to reflect that difference. The intermediate node MUST also save the difference of datagram_size of the first fragment in the VRB and add it to the datagram_size and to the fragment_offset of all the subsequent fragments for that datagram.

New formats

Moved 8 bits from datagram_tag: 5 to fragment_offset, 3 to sequence, to accomodate large MTUs



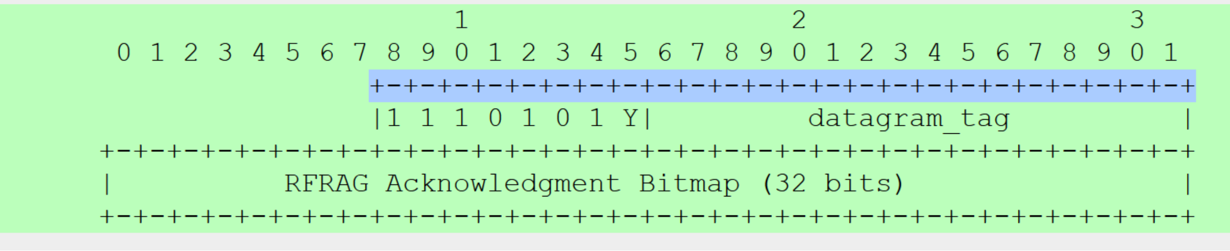
```
                           1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |1 1 1 0 1 0 0 X|E|fragment_size|         datagram_tag          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |sequence |  fragment_offset    |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                                 X set == Ack Requested

                 Figure 2: RFRAG Dispatch type and Header

   X: 1 bit; Ack Requested: when set, the sender requires an RFRAG
      Acknowledgment from the receiver.
```



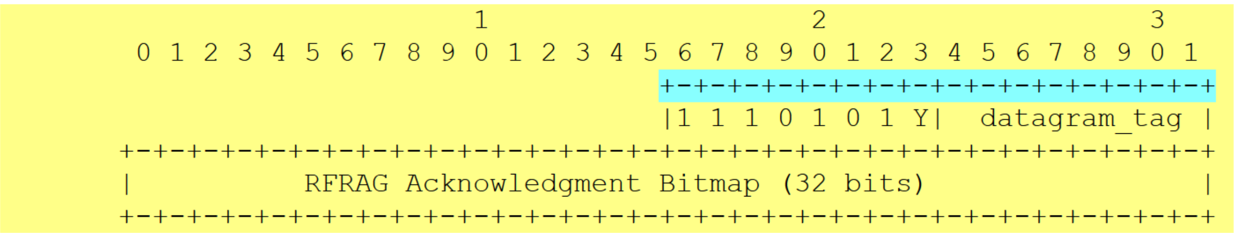
```
                           1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                      |1 1 1 0 1 0 0|E|  datagram_tag |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |X| sequence|   fragment_size   |       fragment_offset         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                              X set == Ack Requested

                Figure 2: RFRAG Dispatch type and Header
```

__Rfrag__

__Rfrag_Ack__



```
                           1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                      |1 1 1 0 1 0 1 Y|         datagram_tag          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          RFRAG Acknowledgment Bitmap (32 bits)                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```



```
                           1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
                                      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                      |1 1 1 0 1 0 1 Y|  datagram_tag |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          RFRAG Acknowledgment Bitmap (32 bits)                |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Next Steps

- Ready for WGLC…

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

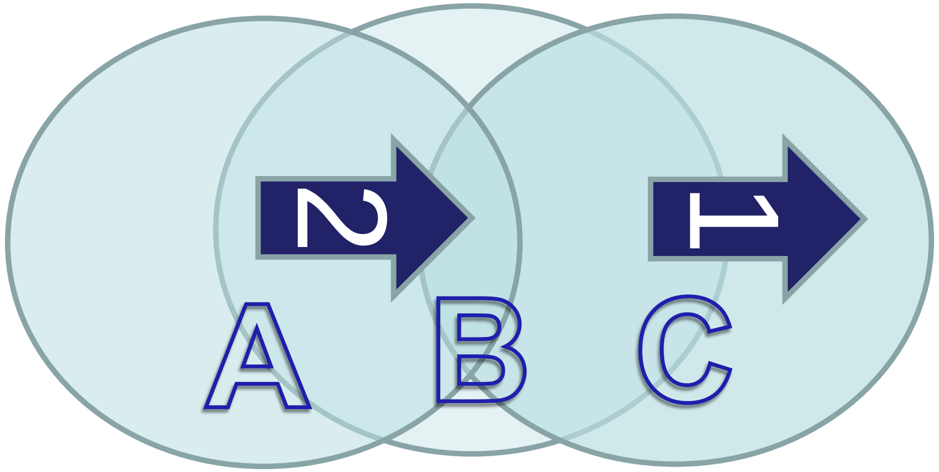
Early fragment forwarding issues #1 c’d

- 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

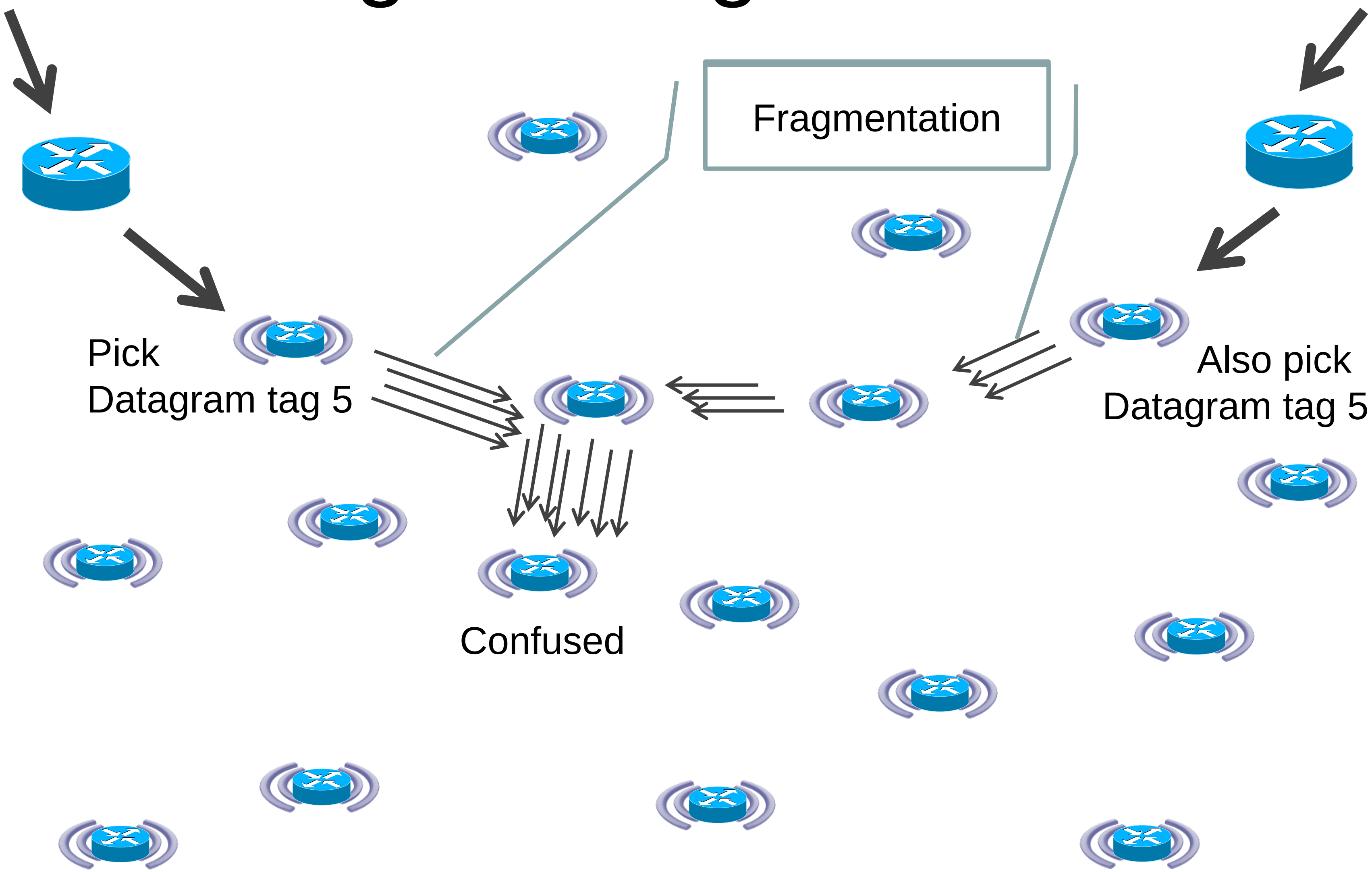
Streamlining with larger window

	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

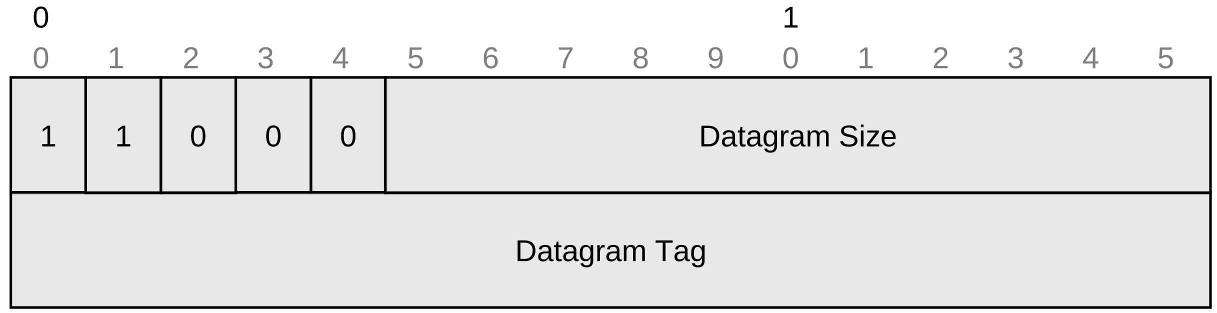
Conclusion

- 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

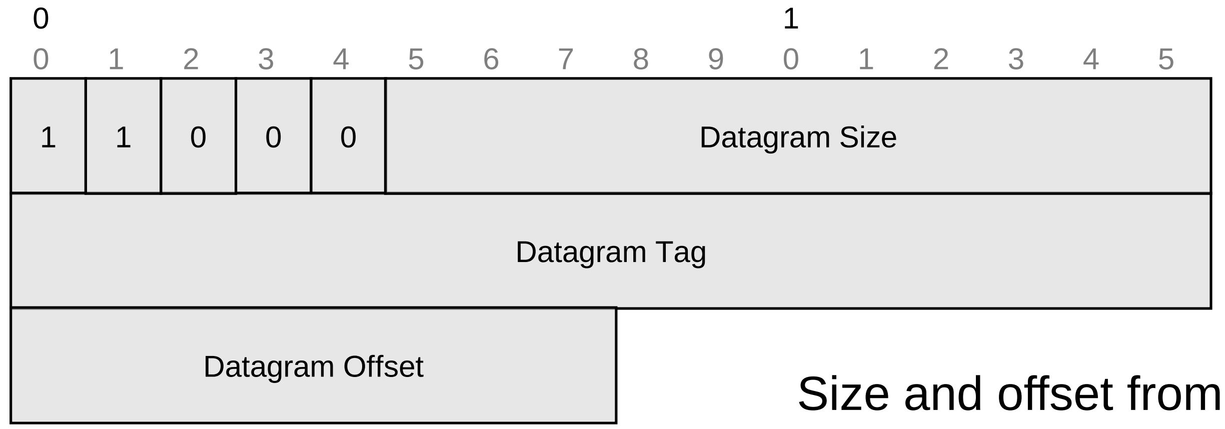
- Provides Label Switching
- Selective Ack
- Pacing and windowing + ECN
- Flow termination indication and reset
- Yes it is transport within transport (usually UDP)
- Yes that is architecturally correct because fragment re-composition is an endpoint function
- And No splitting the draft is not appropriate,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

RFC 4944: 6LoWPAN Fragmentation



0										1					
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	1	0	0	0	Datagram Size										
Datagram Tag															

1st fragment

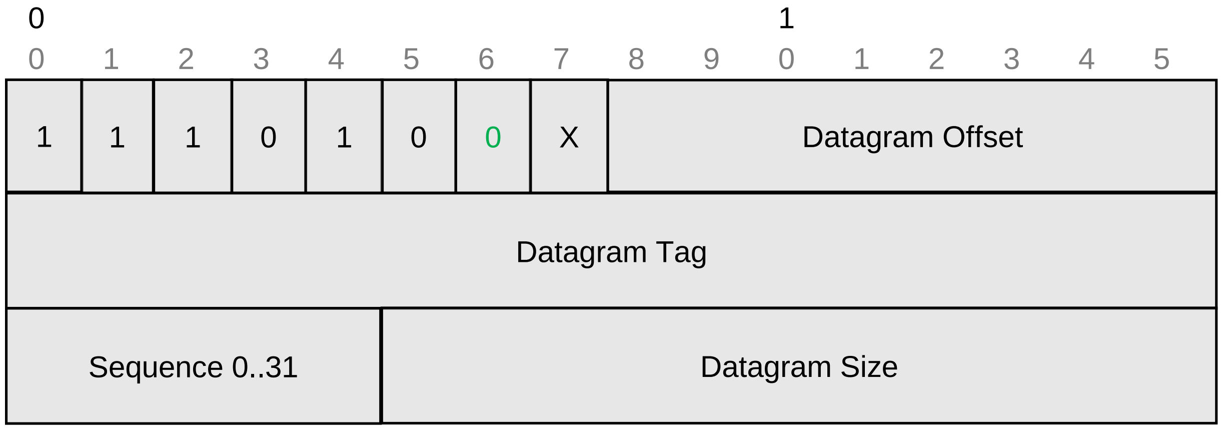


0										1					
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	1	0	0	0	Datagram Size										
Datagram Tag															
Datagram Offset															

Next fragments

Size and offset from uncompressed form
1-hop technology

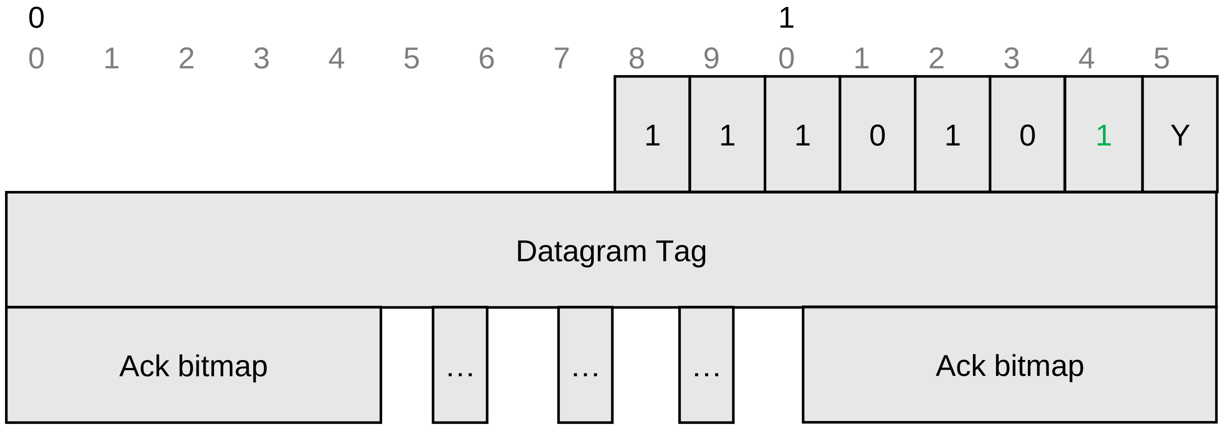
draft-thubert-6lo-forwarding-fragments



0										1					
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
1	1	1	0	1	0	0	X	Datagram Offset							
Datagram Tag															
Sequence 0..31					Datagram Size										

fragment
X <= ack request

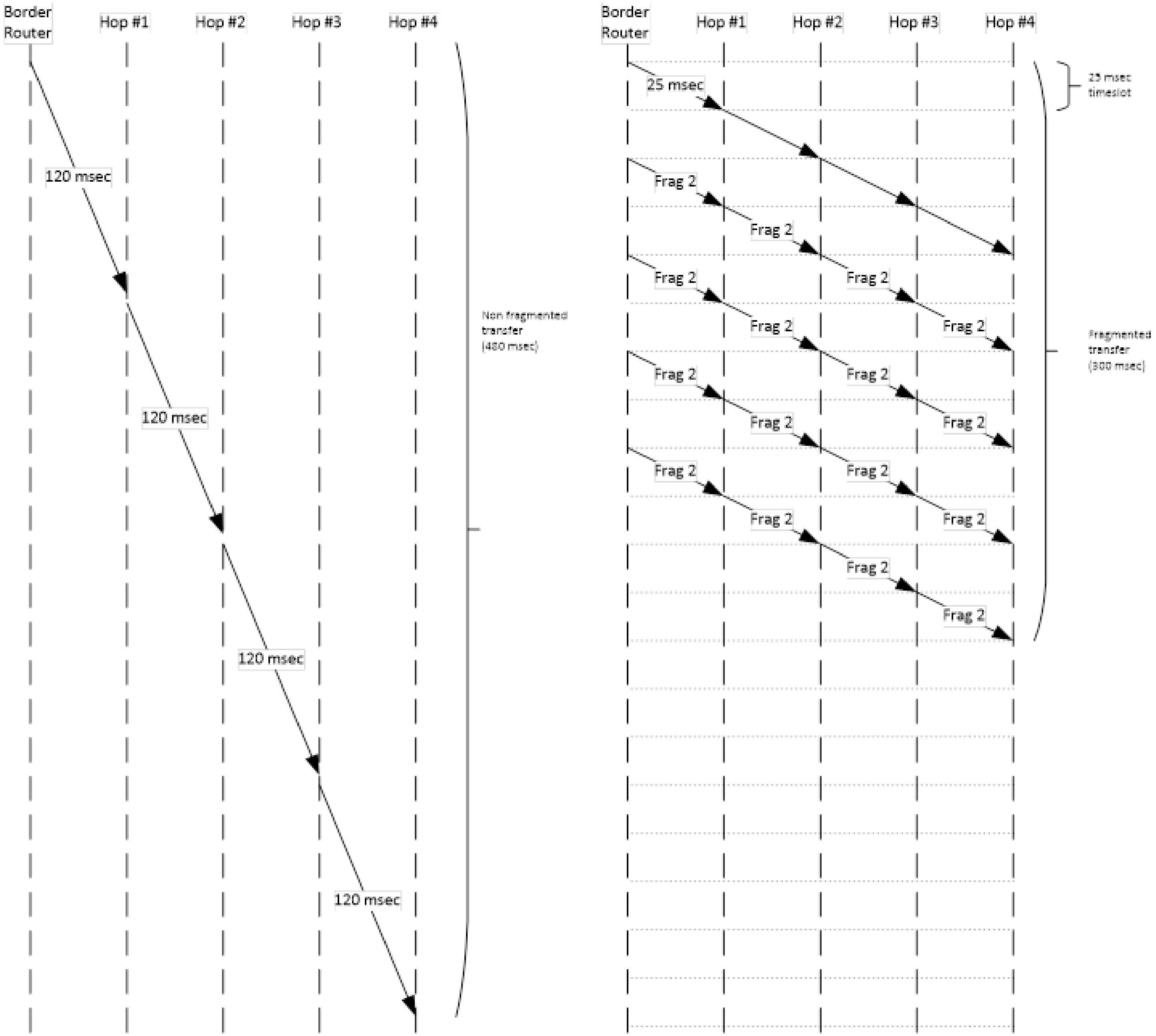
Size and offset from compressed form



0										1					
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5
								1	1	1	0	1	0	1	Y
Datagram Tag															
Ack bitmap						…		…		…		Ack bitmap			

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
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
Frag 2
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				