The problem with fragments in 6lo mesh networks

P.Thubert

IETF 98

Chicago

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

History

- Fragmentation Considered Harmful

http://uojcourses.awardspace.com/network%20system%20design/Course%20files/Fragmentation%20Considered%20Harmful.pdf

  - Fragmentation causes inefficient resource usage
  - Poor performance when fragments are lost
  - Efficient reassembly is difficult

- RFC 4963: corruption due to tagging space limits
- From IPv4: avoid fragmenting in the Network
- From IPv6: MTU discovery is still trouble
- From early 6LoWPAN experimentation: Damn

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

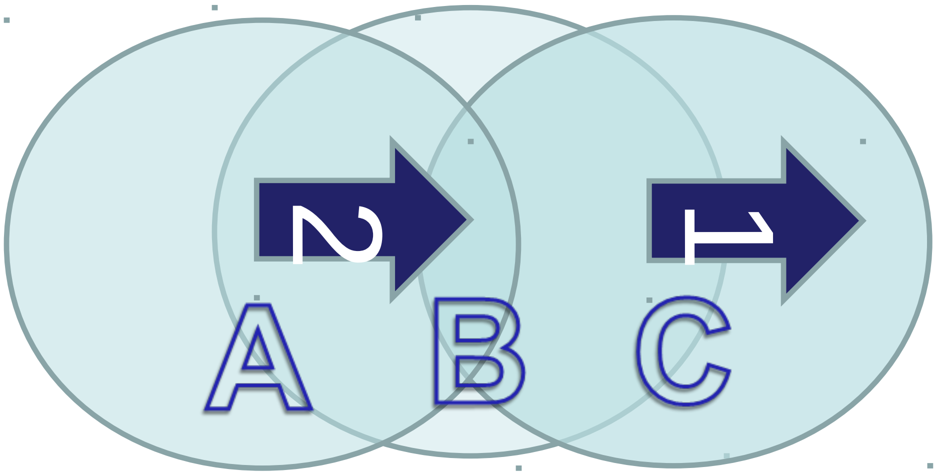
- Need either to abandon fragmented packet
- or discover loss and retry quickly, both need signaling
- Solution is well-know:
  - selective acknowledgement
  - reset
- Requires new signaling

=> 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 then 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

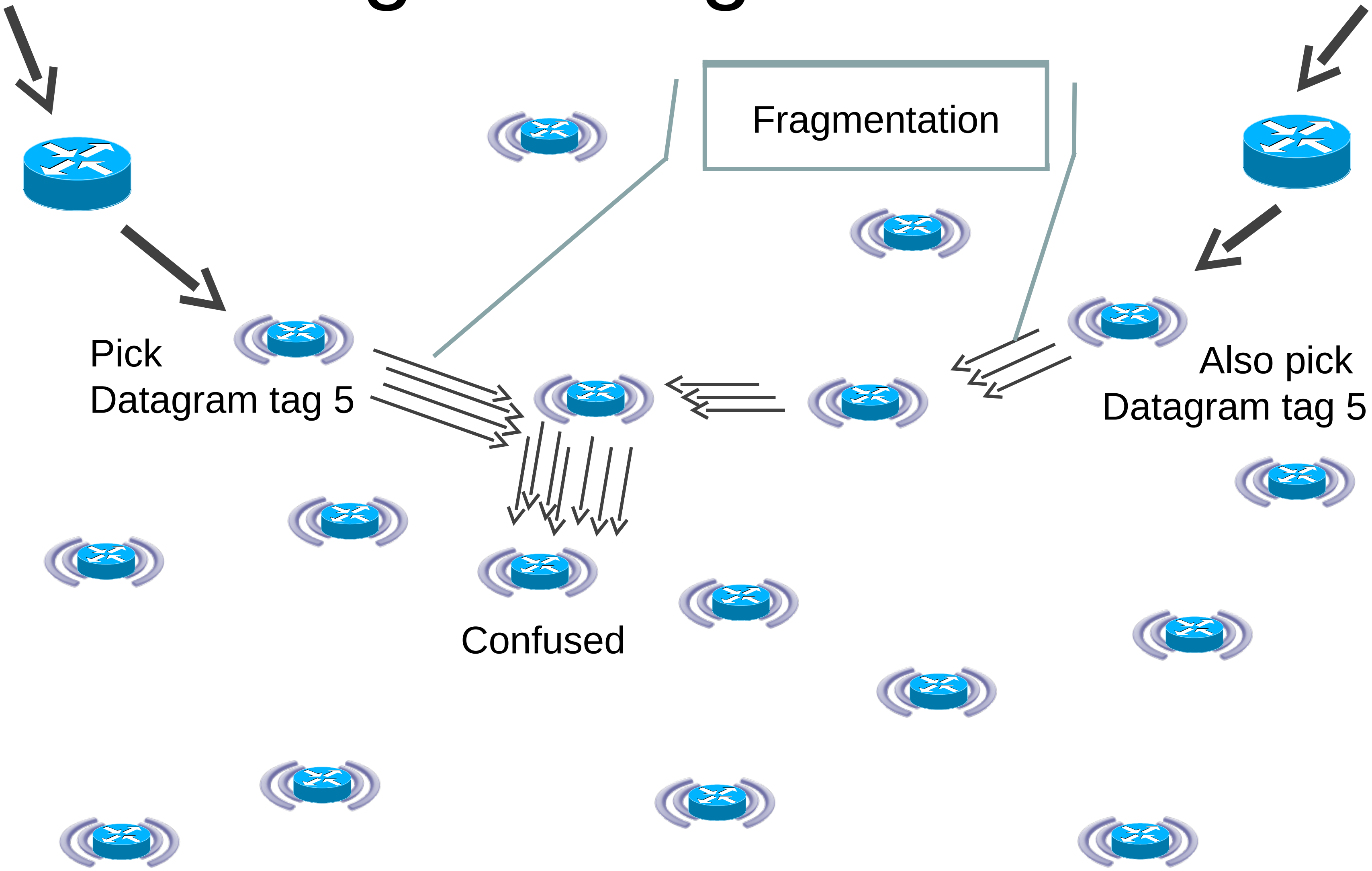
Streamlining with larger window

	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



Fragmentation

Pick
Datagram tag 5

Also pick
Datagram tag 5

Confused

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

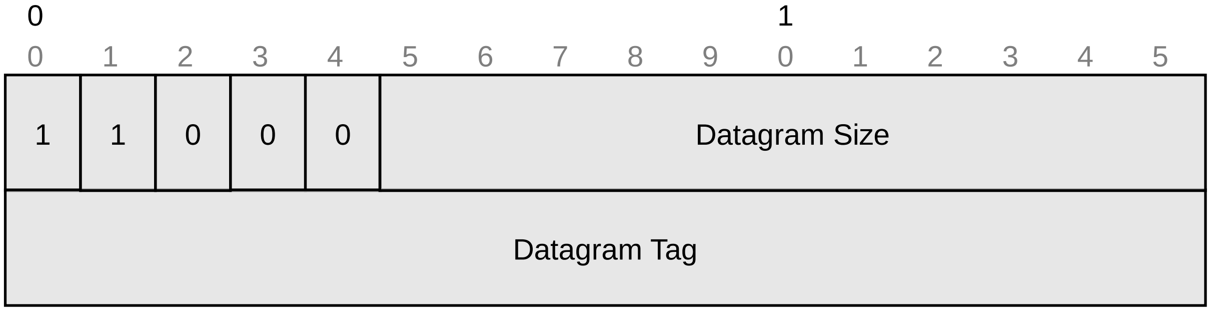
Conclusion

- People are experimenting 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 as predictable as what was already observed
- Some issues can be alleviated by Informational recommendations
- Some require a more appropriate signaling
- Recommendation is rethink 6LoWPAN fragmentation

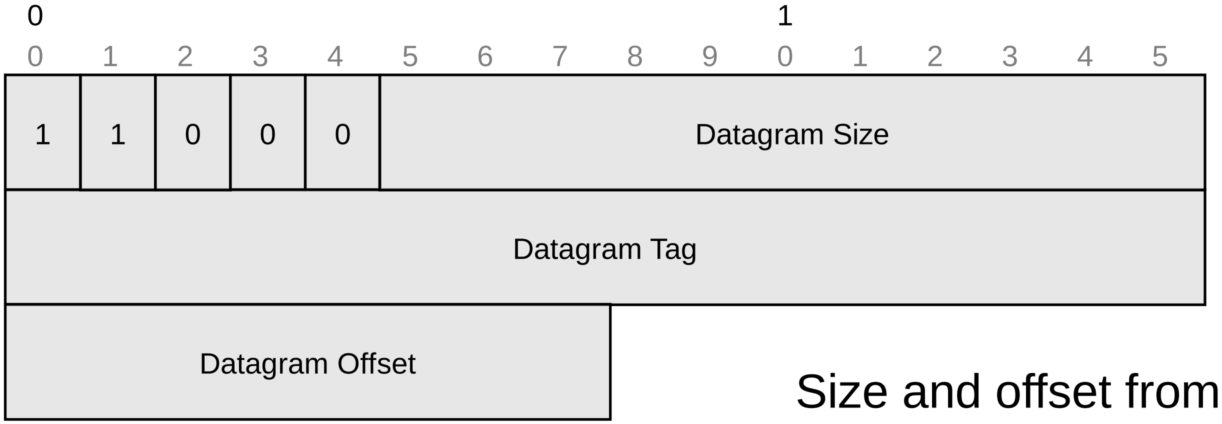
draft-thubert-6lo-forwarding-fragments

- 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



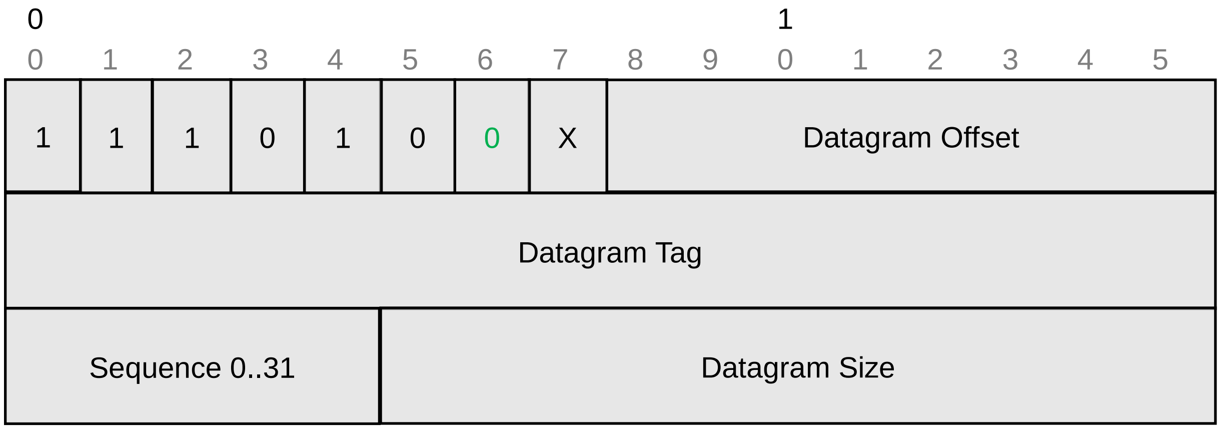
1st fragment



Next fragments

Size and offset from uncompressed form
1-hop technology

draft-thubert-6lo-forwarding-fragments



0 1

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5

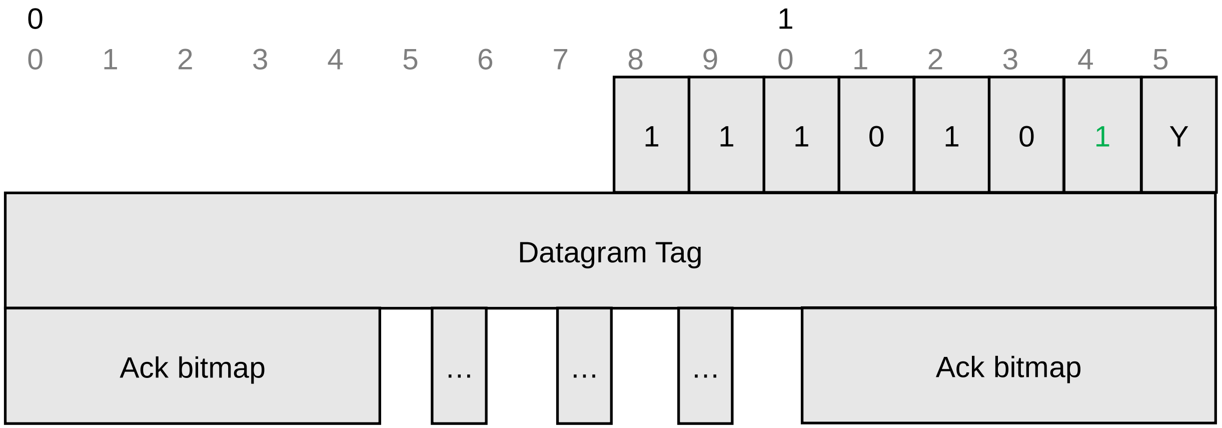
1 1 1 0 1 0 0 X Datagram Offset

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

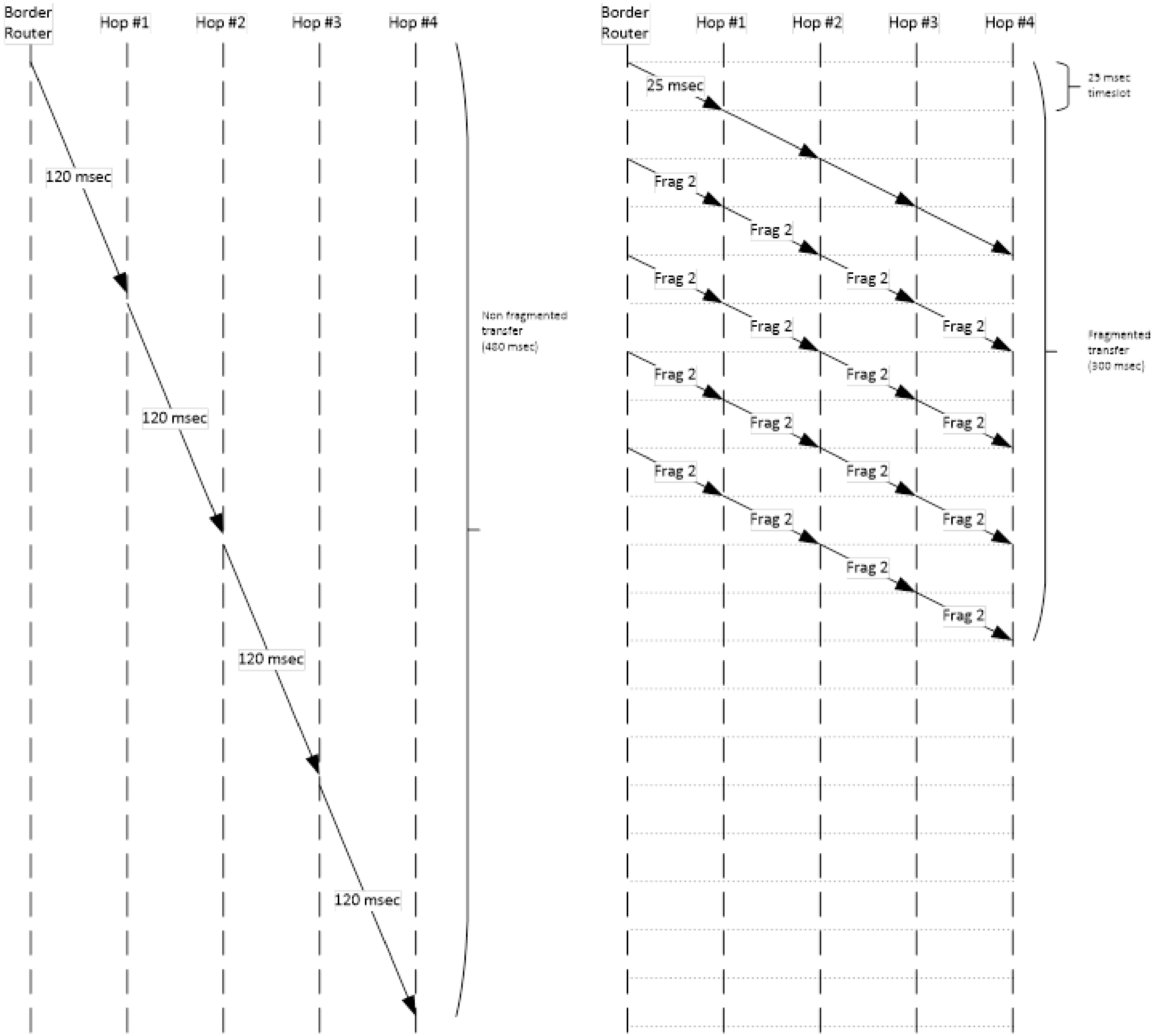
1 1 1 0 1 0 1 Y

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
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				