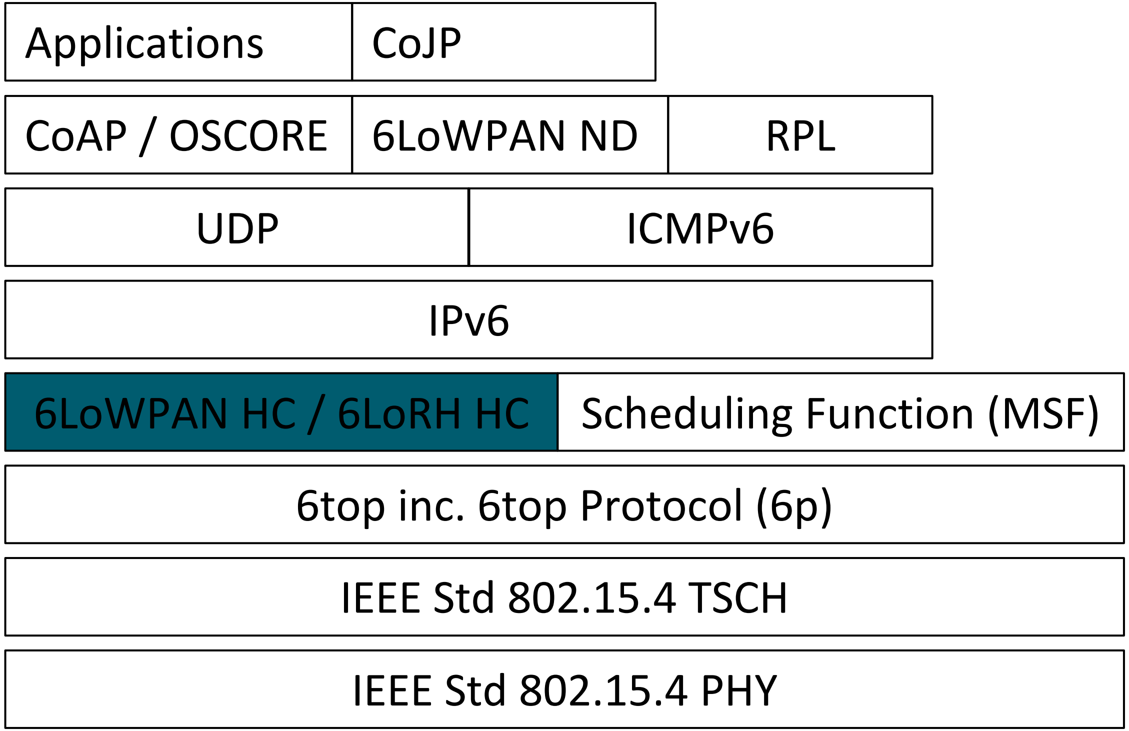
Institut Mines-Télécom

6LoWPAN: Fragmentation & Reassembly, Frame Delivery Modes, & Fragment Forwarding

Georgios Z. PAPADOPOULOS, Professor, IMT Atlantique
georgios.papadopoulos@imt-atlantique.fr
www.georgiospapadopoulos.com
www.youtube.com/c/gzpapadopoulos

The 6TiSCH Protocol Stack



IETF Standards Overview

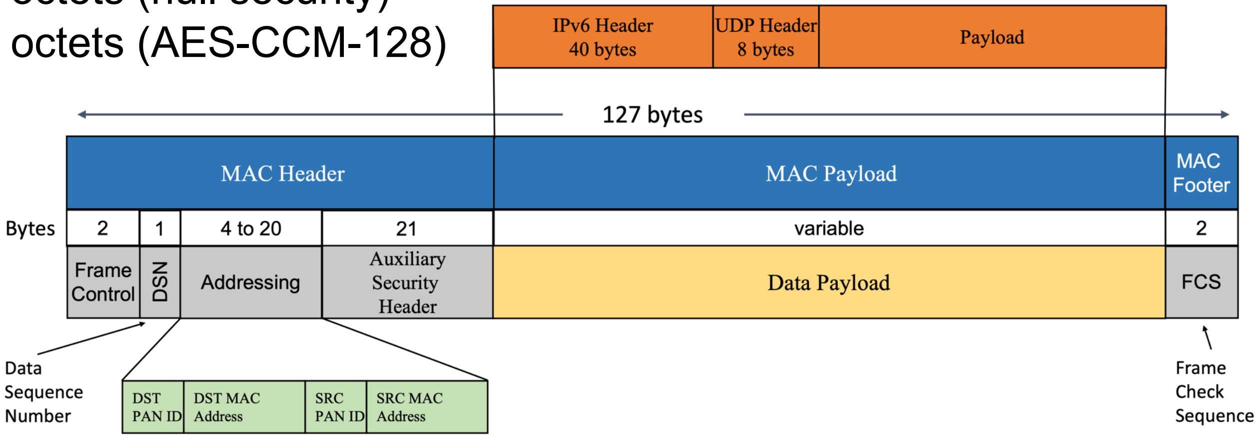
- The **6LoWPAN** is an adaptation layer that enables to transport IPv6 packets over IEEE Std 802.15.4 links:
  - Compression: reduces the size of 40-byte IPv6 header and higher protocols, i.e., UDP headers, RFC 6282.
  - Fragmentation: split (and reassembles) the IPv6 datagrams into smaller fragments, RFC 4944.
  - Fragment Delivery (Mesh-Under & Route-Over): RFC 4944.
  - 6LoWPAN Fragment Forwarding (6LFF): RFC 8930.
  - 6LoWPAN Selective Fragment Recovery mechanism: RFC 8931.

Problem Statement

- IEEE Std 802.15.4 has small MTU (i.e., 127 bytes).
- Header Size Calculation:
  - IPv6 header consists of 40 octets, while the UDP header of 8 octets.
  - IEEE Std 802.15.4 MAC header can be up to 25 octets (null security), **or** 25+21=46 octets (AES-CCM-128).
  - With the IEEE Std 802.15.4 frame size of 127 octets:

    127-25-40-8 = **54** octets (null security)

    127-46-40-8 = **33** octets (AES-CCM-128)



Encapsulation Header Format

- *encapsulated **IPv6 datagram***

IPv6 Dispatch	IPv6 Header	Payload

- *encapsulated LOWPAN_IPHC **compressed IPv6 datagram***

IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires **mesh addressing***

Mesh Type	Mesh Header	IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires **fragmentation***

Frag Type	Frag Header	IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires both **mesh addressing** and **fragmentation***

M Type	M Hdr	Frg Type	Frg Hdr	IPHC Dsp	IPHC Hdr	Payload

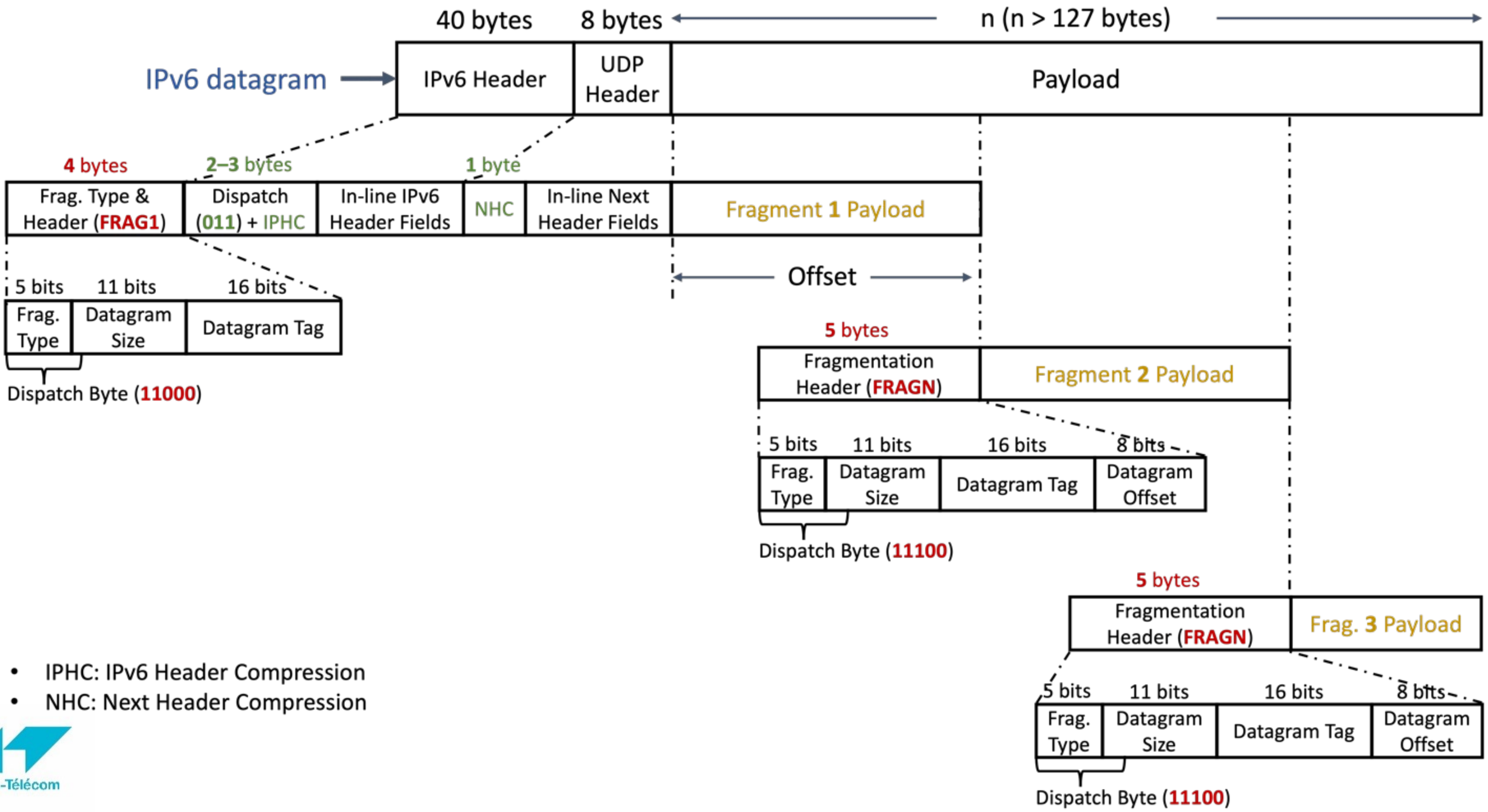
- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires both **mesh addressing** and a **broadcast header** to support mesh broadcast/multicast*

M Type	M Hdr	B Dsp	B Hdr	IPHC Dsp	IPHC Hdr	Payload

6LoWPAN Dispatch Codes

Bit Pattern	Short Code	Description
00 xxxxxx	NALP	Not A 6LoWPAN Packet
01 000001	IPv6	Uncompressed IPv6 address
~~01 000010~~	~~LOWPAN_HC1~~	~~HC1 Compressed IPv6 header~~ (***obsolete***)
01 010000	LOWPAN_BC0	BC0 Broadcast header
01 1	LOWPAN_IPHC	IPHC Compressed IPv6 header (***new version, RFC 6282***)
10 xxxxxx	MESH	Mesh routing header
11 000xxx	FRAG1	Fragmentation header (first fragment)
11 100xxx	FRAGN	Fragmentation header (subsequent fragment)

6LoWPAN Compression and Fragmentation Overview



- IPHC: IPv6 Header Compression
- NHC: Next Header Compression

SOMMAIRE

Institut Mines-Télécom

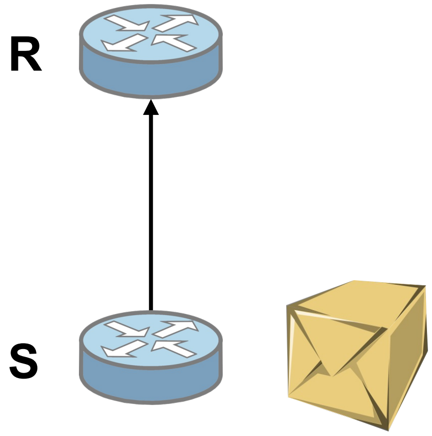
Chapter 1
RFC 4944: 6LoWPAN Fragmentation & Reassembly

Check the relevant video
"*6LoWPAN Fragmentation and Reassembly*"
on YouTube!

Institut Mines-Télécom

Context

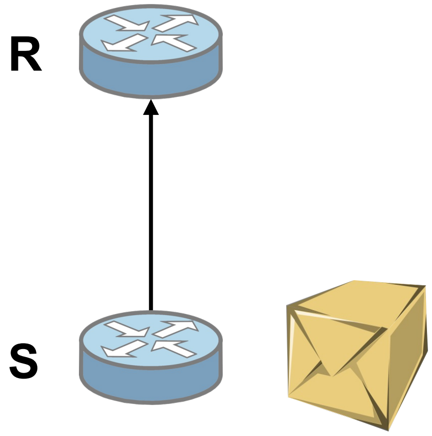
RFC 4944



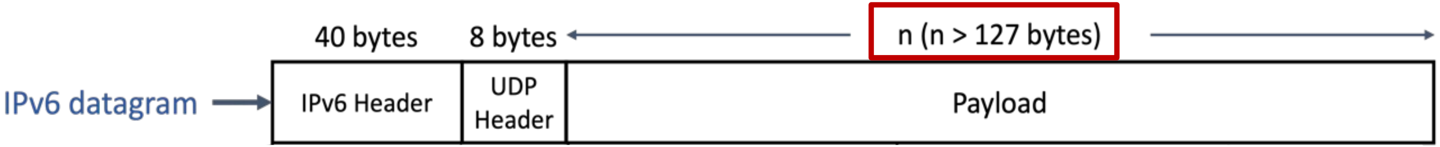
Institut Mines-Télécom

Context

RFC 4944



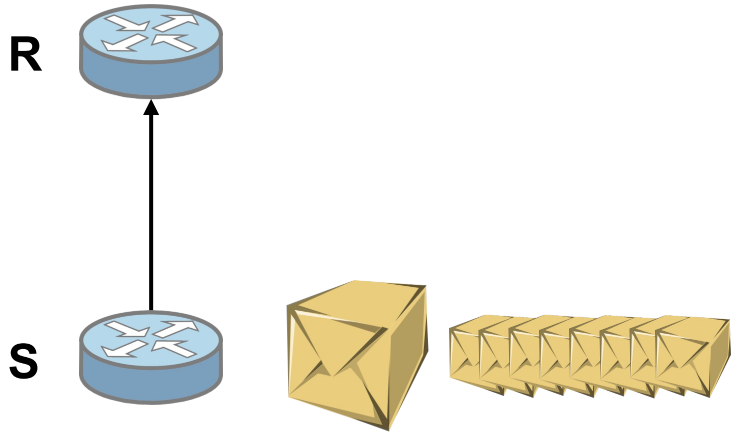
When an IPv6 packet is larger than IEEE 802.15.4 MTU (127 bytes)



Institut Mines-Télécom

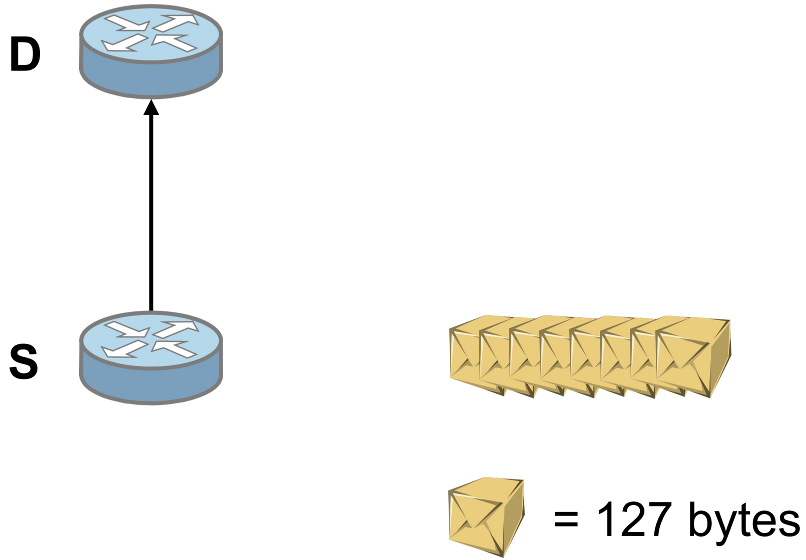
Context

RFC 4944



Context

RFC 4944



Encapsulation Header Format

- *encapsulated **IPv6 datagram***

IPv6 Dispatch	IPv6 Header	Payload

- *encapsulated LOWPAN_IPHC **compressed IPv6 datagram***

IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires **mesh addressing***

Mesh Type	Mesh Header	IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires **fragmentation***

Frag Type	Frag Header	IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires both **mesh addressing** and **fragmentation***

M Type	M Hdr	Frg Type	Frg Hdr	IPHC Dsp	IPHC Hdr	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires both **mesh addressing** and a **broadcast header** to support mesh broadcast/multicast*

M Type	M Hdr	B Dsp	B Hdr	IPHC Dsp	IPHC Hdr	Payload

Institut Mines-Télécom

6LoWPAN Dispatch Codes

Bit Pattern	Short Code	Description
00 xxxxxx	NALP	Not A 6LoWPAN Packet
01 000001	IPv6	Uncompressed IPv6 address
~~01 000010~~	~~LOWPAN_HC1~~	~~HC1 Compressed IPv6 header~~ (***obsolete***)
01 010000	LOWPAN_BC0	BC0 Broadcast header
01 1	LOWPAN_IPHC	IPHC Compressed IPv6 header (***new version, RFC 6282***)
10 xxxxxx	MESH	Mesh routing header
11 000xxx	FRAG1	Fragmentation header (first fragment)
11 100xxx	FRAGN	Fragmentation header (subsequent fragment)

Fragmentation Type and Header

To enable the fragmentation and reassembly operations, 6LoWPAN defines two fragment headers:

```
 0                   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 0 0 0 |    datagram_size    |         datagram_tag          |
+-----------+---------------------+-------------------------------+
```

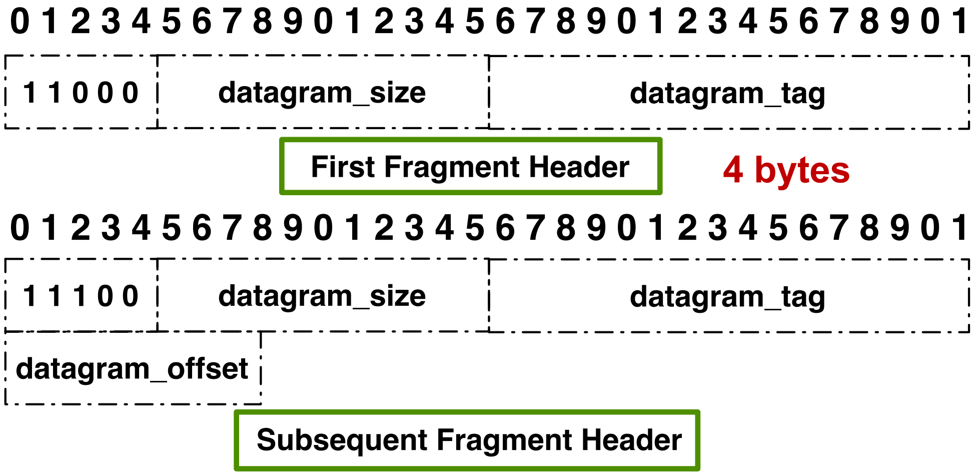
First Fragment Header

```
 0                   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 0 |    datagram_size    |         datagram_tag          |
+-----------+---------------------+-------------------------------+
|datagram_offset|
+---------------+
```

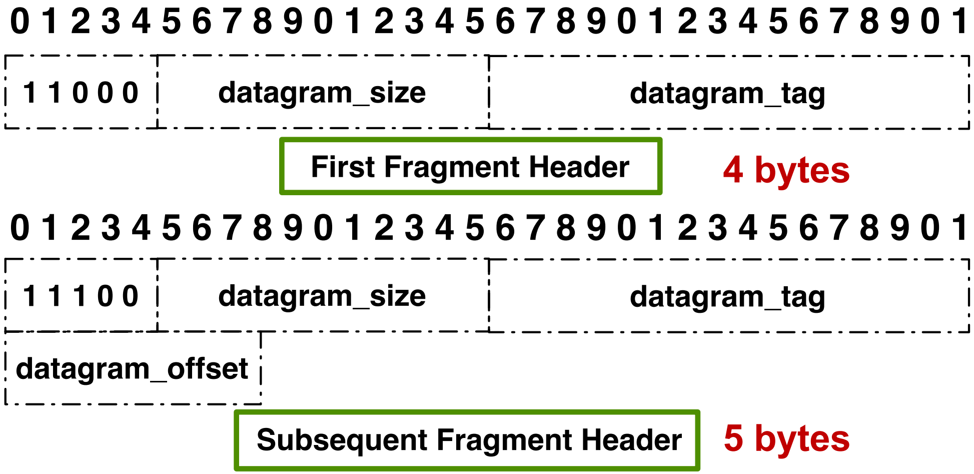
Subsequent Fragment Header

Fragmentation Type and Header

where the header for the first fragment consists of 4 bytes,

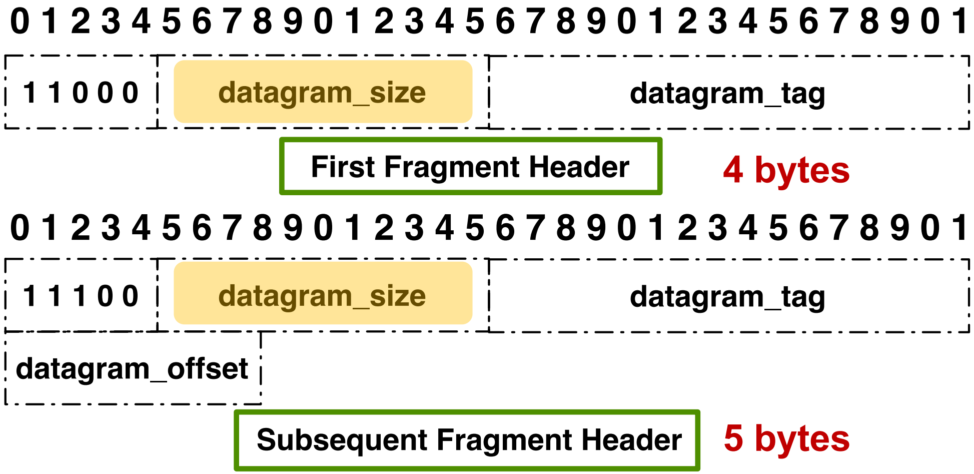


while the header for the subsequent fragments of 5 bytes.



Fragmentation Type and Header

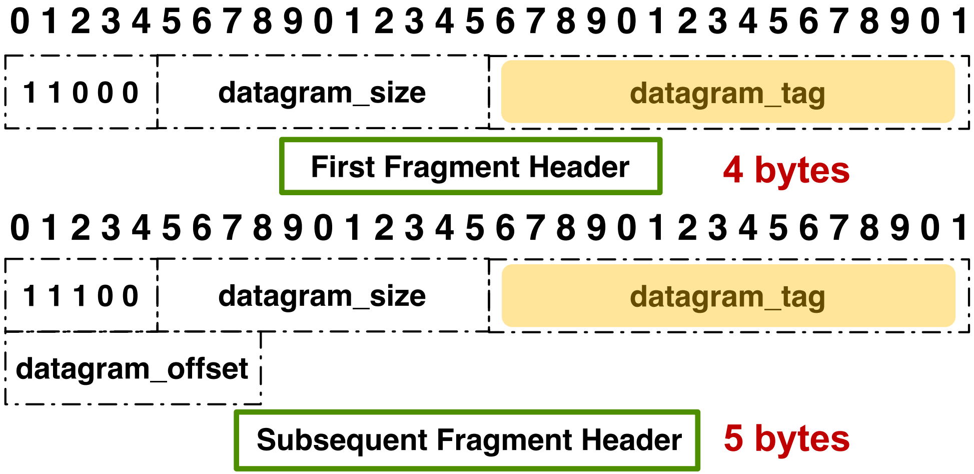
RFC 4944



1. The ***datagram_size*** to identify the size of the IPv6 datagram.

Fragmentation Type and Header

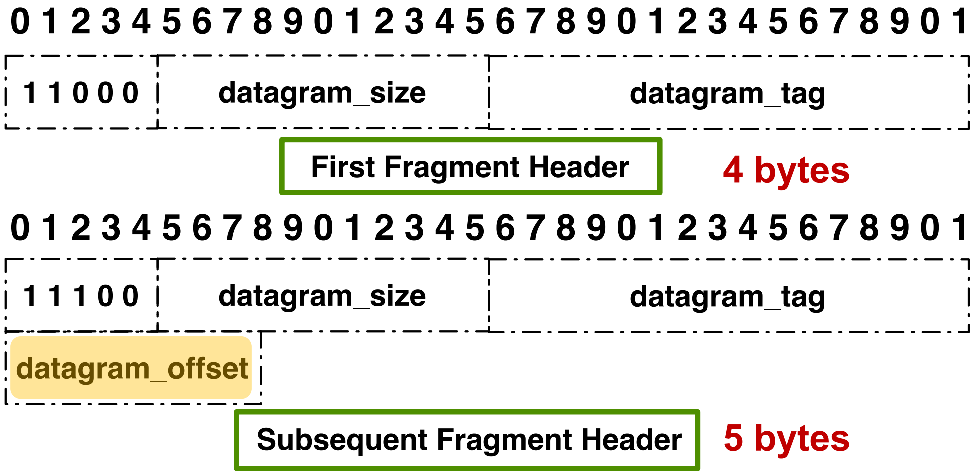
RFC 4944



2. The ***datagram_tag*** (in conjunction with the MAC source address) to identify all fragments of a single datagram.

Fragmentation Type and Header

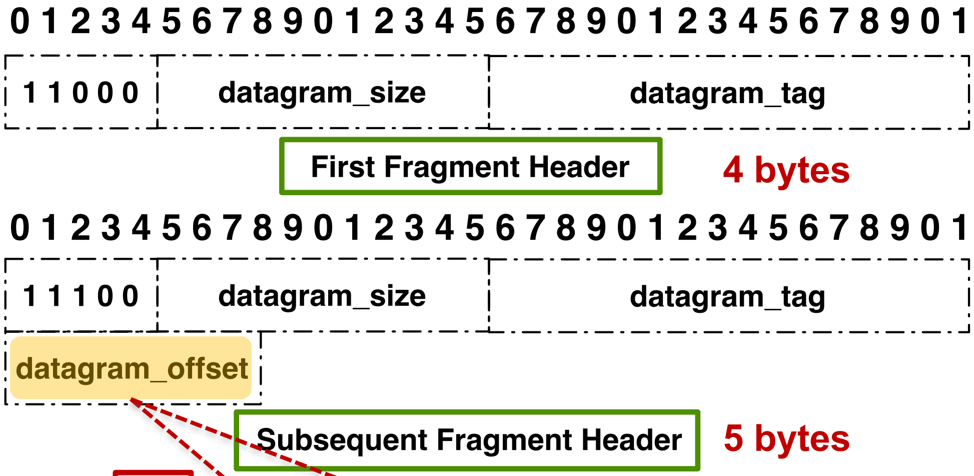
RFC 4944

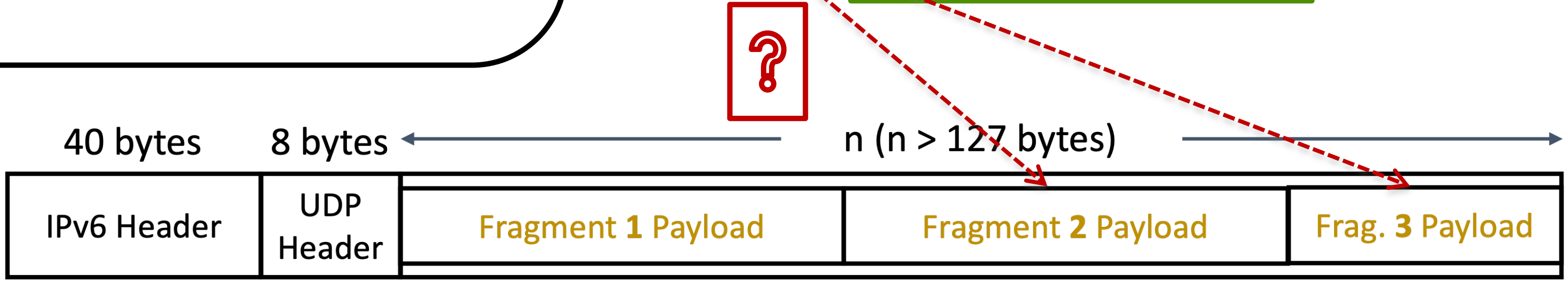


3. The ***datagram_offset*** to identify the location of the received fragment. This field is included **only** in the second and subsequent link-layer fragments of an IPv6 datagram, and it specifies the offset in increments of 8 bytes.

Fragmentation Type and Header

In other words, it identifies the relative position of the received link-layer fragment from the beginning of the payload datagram.

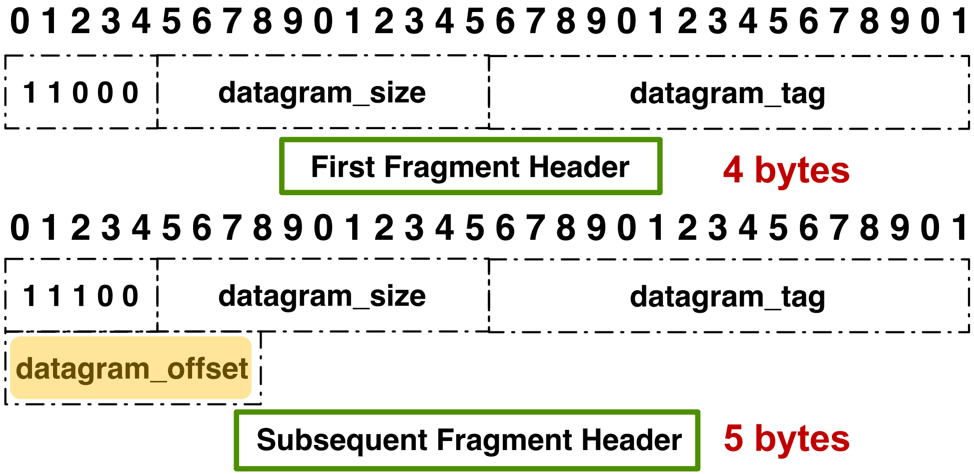




Institut Mines-Télécom

Fragmentation Type and Header

RFC 4944



The ***datagram_offset*** allows for out-of-sequence delivery!

Fragmentation Type and Header

Here, we have the Fragmentation Type and Header format for the first link-layer fragment,

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 0 0 0	datagram_size	datagram_tag

First Fragment Header **4 bytes**

Fragmentation Type and Header

where the first 5 bits of Dispatch Value Bit Pattern indicate the Fragmentation Type. In this case, it is equal to 11000 which represents the 1st fragment of an IPv6 datagram.

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

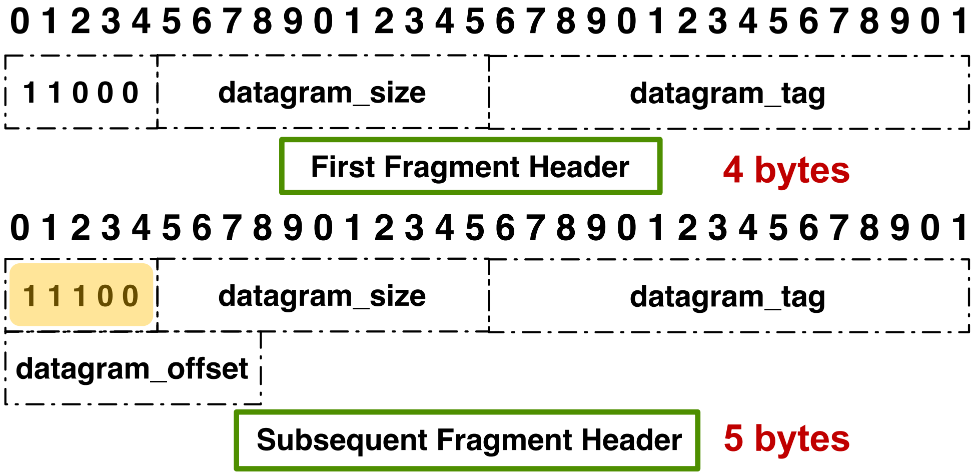
1 1 0 0 0	datagram_size	datagram_tag

First Fragment Header **4 bytes**

Dispatch Value Bit Pattern → the Fragmentation Type

Fragmentation Type and Header

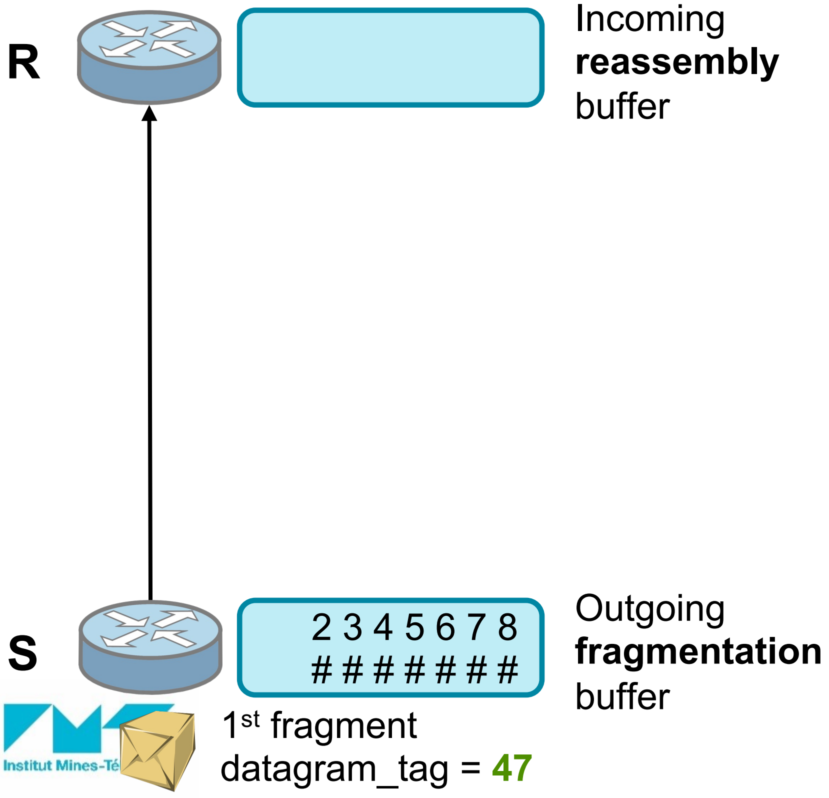
In the *Figure below*, the first 5 bits is equal to 11100 which represents the second or subsequent fragment of an IPv6 datagram.



Fragmentation Operation: Summary

- Fragmentation procedure takes place only when an IPv6 datagram, after the compression procedure, does not fit within an IEEE Std 802.15.4 frame, i.e., 127 bytes.
- The transmitter splits the datagram into multiple link fragments of 127 bytes.
- To enable the fragmentation procedure, and later the reassembly, the fragmentation header comes with the following fields:
  - The ***datagram size*** to identify the size of the IPv6 datagram.
  - The ***datagram tag*** to identify all fragments of a single datagram.
  - The ***datagram offset*** to identify the location of the received fragment.

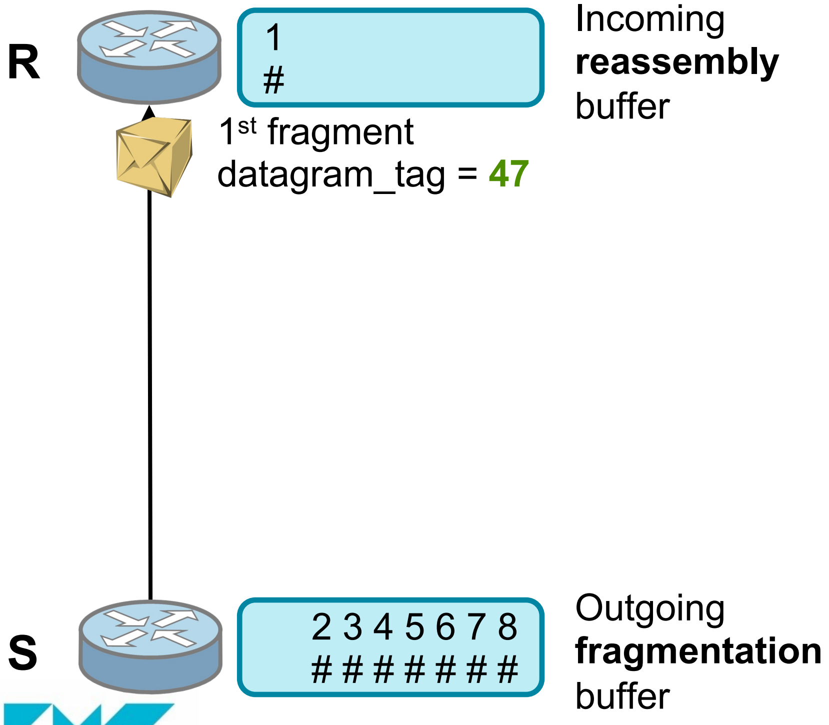
Reassembly Operation



The receiving node R:

(upon receipt of a link-layer fragment …)

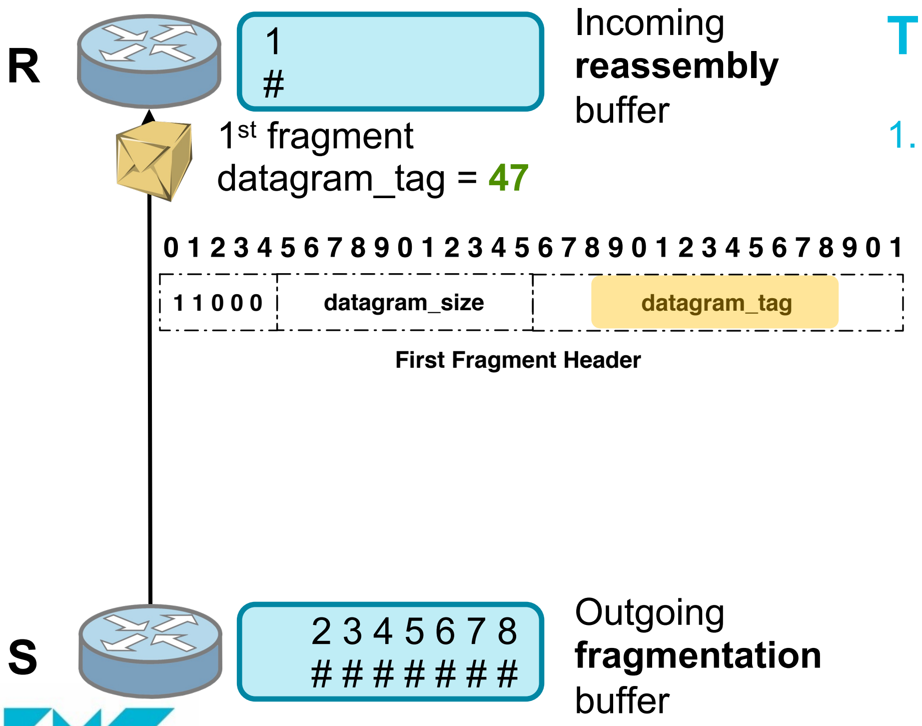
Reassembly Operation



The receiving node R:

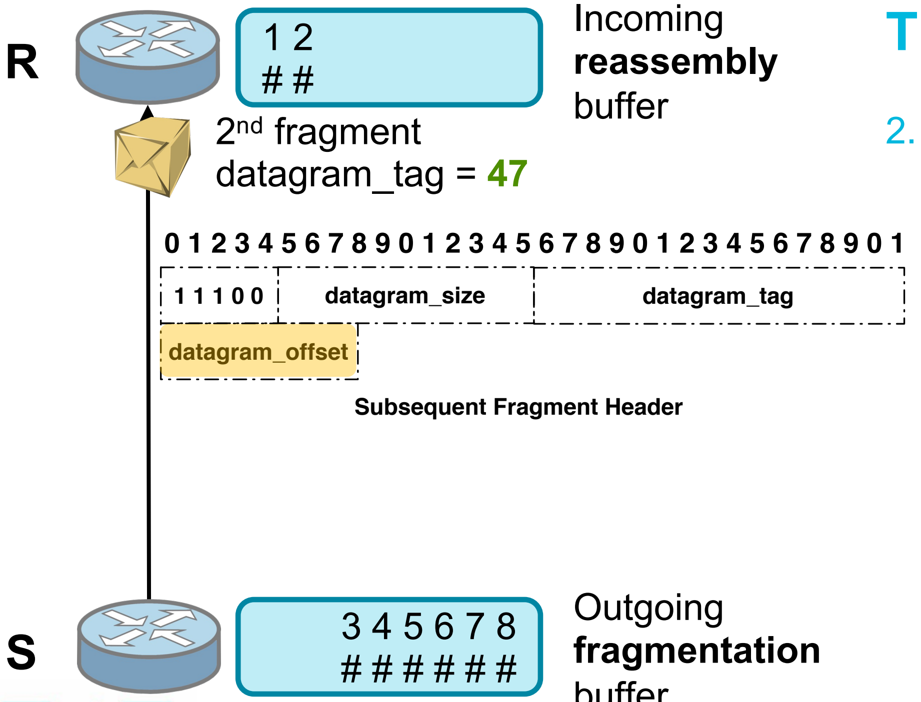
- Initiates the reassembly operation **to reconstruct the original IPv6 datagram**.

Reassembly Operation



The receiving node R:

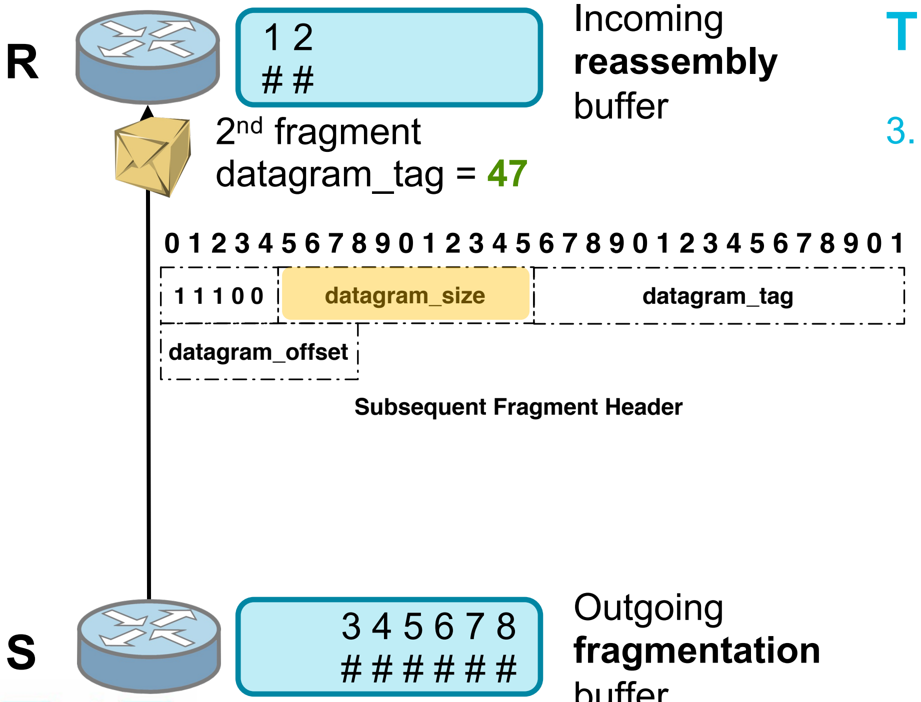
1. Checks the ***datagram_tag*** to identify the fragments that belong to a given IPv6 datagram.



The receiving node R:

2. Checks the ***datagram_offset*** to determine the location of the received individual fragment.

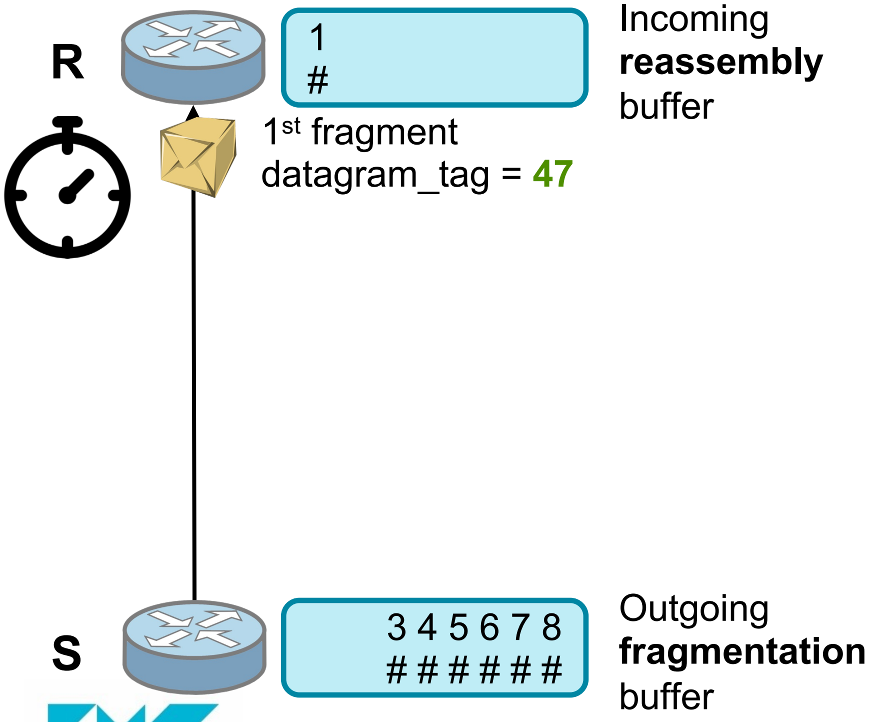
Reassembly Operation



The receiving node R:

3. Checks the ***datagram_size*** to identify the size of the original unfragmented datagram.

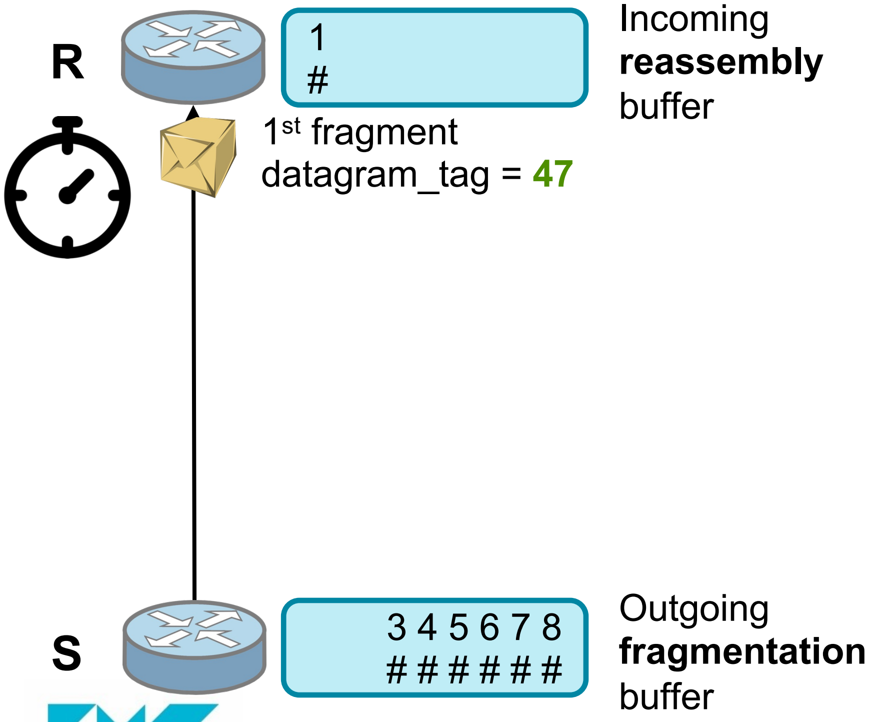
The Reassembly Timer



The receiving node R:

- ***Sets a reassembly timer***

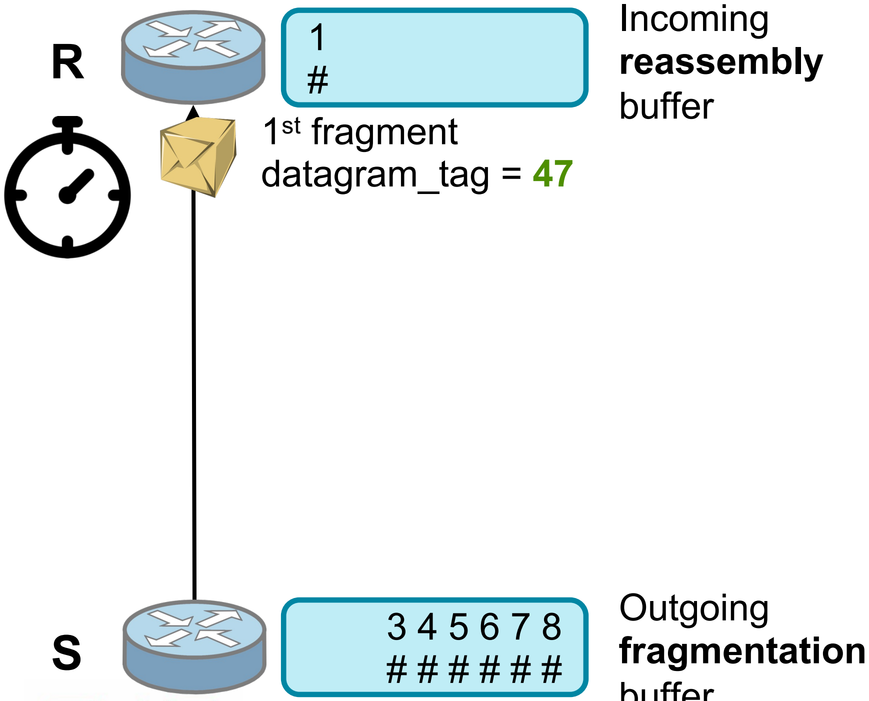
The Reassembly Timer



The receiving node R:

- Sets a reassembly timer, ***which is configured to a maximum of 60 seconds***

The Reassembly Timer



The receiving node R:

- Sets a reassembly timer, which is configured to a maximum of 60 seconds ***to allow discarding the partially reassembled packet after the timeout***.

Reassembly Operation: Summary

- The receiver, once it receives the first fragment, it initiates the reassembly procedure to construct the actual IPv6 data packet.
- It checks ***datagram_tag*** to identify the fragments that belong to a given IPv6 data packet.
- It checks the ***datagram_offset*** to identify the offset (i.e., location) of the received fragment within the original IPv6 data packet.
- The size of the unfragmented data packet as well as the size of the reassembly buffer can be identified by the ***datagram_size***.
- Finally, the receiver uses the source & final destination addresses to later forward the fragments to the next hop.

Chapter 2

RFC 4944: 6LoWPAN Frame Delivery Modes

Check the relevant video
"*6LoWPAN Frame Delivery Modes*"
on YouTube!

Institut Mines-Télécom

RFC 4944

6LoWPAN Frame Delivery modes:

- Mesh-Under.
- Router-Over.

RFC 4944

6LoWPAN Frame Delivery modes:

- Mesh-Under: Layer 2 based on MAC.
- Router-Over.

RFC 4944

6LoWPAN Frame Delivery modes:

- Mesh-Under: Layer 2 based on MAC.
- Router-Over: Layer 3 based on IP.

Mesh-Under:

- The ***routing*** and ***forwarding*** operations are performed at 6LoWPAN Adaptation layer.

RFC 4944

RFC 4944

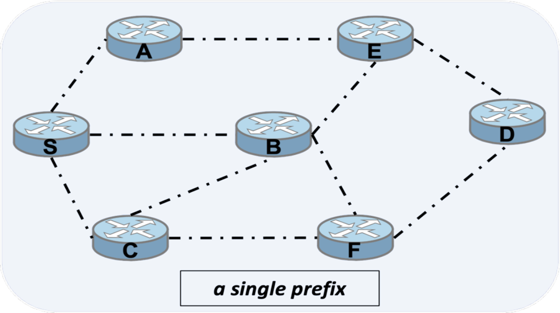
Mesh-Under:

- The ***routing*** and ***forwarding*** operations are performed at 6LoWPAN Adaptation layer.
- The IPv6 header ***does not*** need to be unpacked.

RFC 4944

Mesh-Under:

- The ***routing*** and ***forwarding*** operations are performed at 6LoWPAN Adaptation layer.
- The IPv6 header ***does not*** need to be unpacked.
- All nodes share the same prefix.



RFC 4944

To forward the received fragments, a node requires:

- The ***final destination address***, the link-layer address of the Final Destination.

Mesh-Under: Overview

RFC 4944

To forward the received fragments, a node requires:

- The ***final destination address***, the link-layer address of the Final Destination.
- The ***originator address***, the link-layer address of the Originator.

Mesh-Under: Overview

RFC 4944

To forward the received fragments, a node requires:

- The ***final destination address***, the link-layer address of the Final Destination.
- The ***originator address***, the link-layer address of the Originator.

originator address, final address

NB: Considering that at each forwarding step, the link-layer destination and source addresses are overwritten by the addresses of the next hop and by the node performing the forwarding, **this information regarding the final destination address and the originator address needs to be stored somewhere else**.

Mesh-Under: Mesh Addressing Type and Header

Towards this aim, 6LoWPAN defines the Mesh Header in RFC 4944, where 1 and 0 are the first two bits.

To forward the received fragments, a node requires:

- The ***final destination address***, the link-layer address of the Final Destination.
- The ***originator address***, the link-layer address of the Originator.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	0	1	2	3
1	0							originator address, final address (16 or 64-bit)											

The Mesh Addressing Type & Header

Attention: Mesh Type is defined by 1 and 0 as the first two bits.

Mesh-Under: Mesh Addressing Type and Header

In addition to the addresses, the Mesh Header stores a Layer-2 equivalent of an IPv6 Hop Limit. This value must be decremented by a forwarding node before sending the frame on its next hop; if the value reaches zero, the frame is discarded silently.

To forward the received fragments, a node requires:

- ► The ***final destination address***, the link-layer address of the Final Destination.
- ► The ***originator address***, the link-layer address of the Originator.
- ► ***Hops Left*** (4 bits), a layer-2 equivalent of an IPv6 Hop Limit.

0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	0	1	2	3
1	0			Hops Left				originator address, final address (16 or 64-bit)											

The Mesh Addressing Type & Header

Attention: Mesh Type is defined by 1 and 0 as the first two bits.

Mesh-Under: Mesh Addressing Type and Header

Finally, the mesh header defines the V and F bits that indicate whether the Originator (or “Very first”) address and the Final Destination address, respectively, are 16-bit short or 64-bit EUI-64 addresses.

To forward the received fragments, a node requires:

- The ***final destination address***, the link-layer address of the Final Destination.
- The ***originator address***, the link-layer address of the Originator.
- ***Hops Left*** (4 bits), a layer-2 equivalent of an IPv6 Hop Limit.
- ***V*** and ***F bits*** to indicate whether the addresses are 16-bit short or 64-bit EUI-64.

0	1	2	3	4 5 6 7	8 9 0 1 2 3 4 5 0 1 2 3
1	0	V	F	Hops Left	originator address, final address (16 or 64-bit)

The Mesh Addressing Type & Header

Attention: Mesh Type is defined by 1 and 0 as the first two bits.

RFC 4944: 6LoWPAN Frame Delivery Modes

Encapsulation Header Format

- *encapsulated **IPv6 datagram***

IPv6 Dispatch	IPv6 Header	Payload

- *encapsulated LOWPAN_IPHC **compressed IPv6 datagram***

IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires **mesh addressing***

Mesh Type	Mesh Header	IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires **fragmentation***

Frag Type	Frag Header	IPHC Dispatch	IPHC Header	Payload

- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires both **mesh addressing** and **fragmentation***

M Type	M Hdr	Frg Type	Frg Hdr	IPHC Dsp	IPHC Hdr	Payload

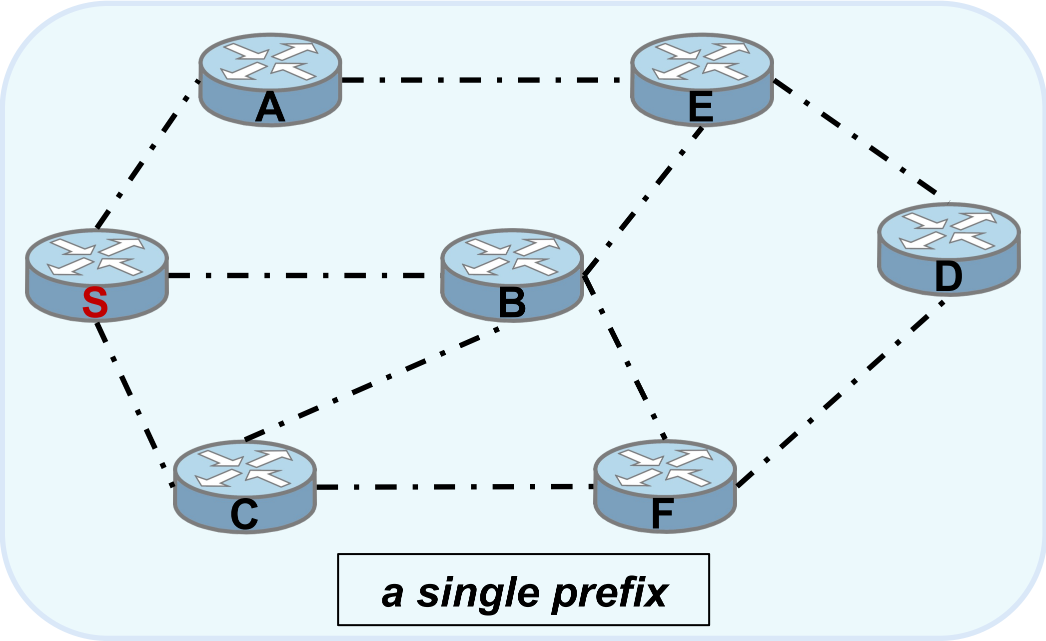
- *encapsulated LOWPAN_IPHC compressed IPv6 datagram that requires both **mesh addressing** and a **broadcast header** to support mesh broadcast/multicast*

M Type	M Hdr	B Dsp	B Hdr	IPHC Dsp	IPHC Hdr	Payload

6LoWPAN Dispatch Codes

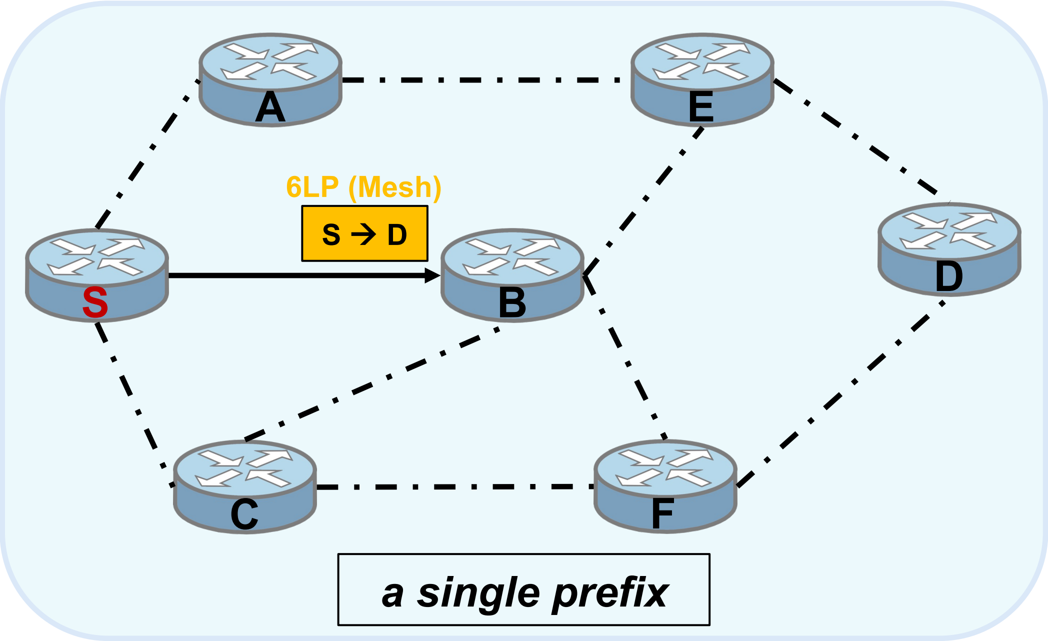
Bit Pattern	Short Code	Description
00 xxxxxx	NALP	Not A 6LoWPAN Packet
01 000001	IPv6	Uncompressed IPv6 address
~~01 000010~~	~~LOWPAN_HC1~~	~~HC1 Compressed IPv6 header~~ (*obsolete*)
01 010000	LOWPAN_BC0	BC0 Broadcast header
01 1	LOWPAN_IPHC	IPHC Compressed IPv6 header (***new version, RFC 6282***)
10 xxxxxx	MESH	Mesh routing header
11 000xxx	FRAG1	Fragmentation header (first fragment)
11 100xxx	FRAGN	Fragmentation header (subsequent fragment)

Mesh-Under: Operation Example



Let's assume that node S uses the Mesh-Under approach to deliver a frame.

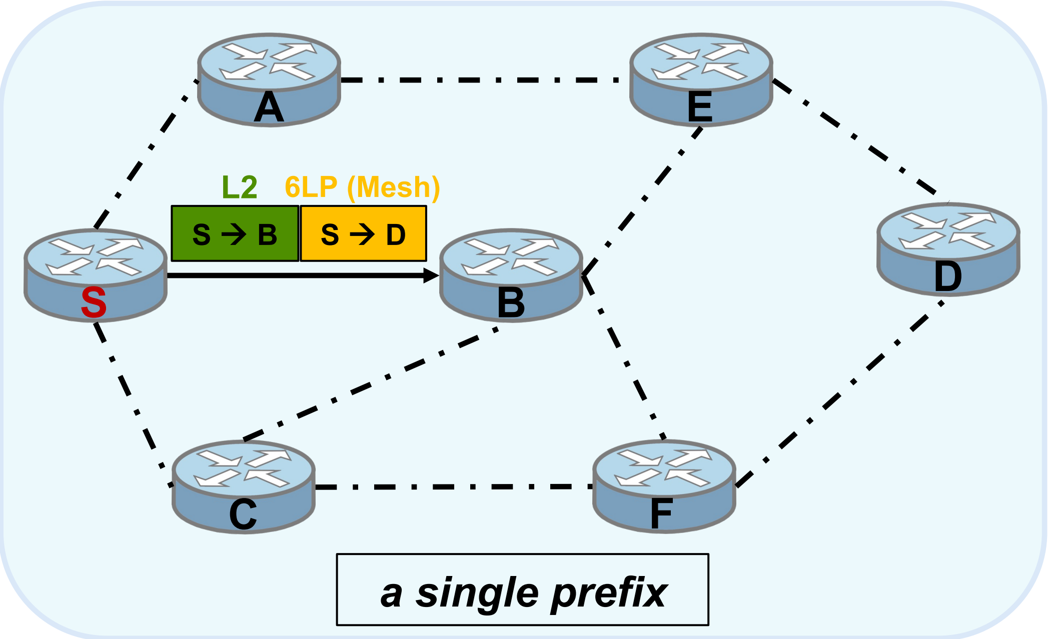
Mesh-Under: Operation Example



Then, it MUST include a Mesh Addressing header with the Originator's link-layer address set to its own, and the Final destination's link-layer address set to the frame's ultimate destination, in this case of node D.

6LP: **6LoWPAN Adaptation Layer** (i.e., Mesh Addressing Type & Header)

Mesh-Under: Operation Example

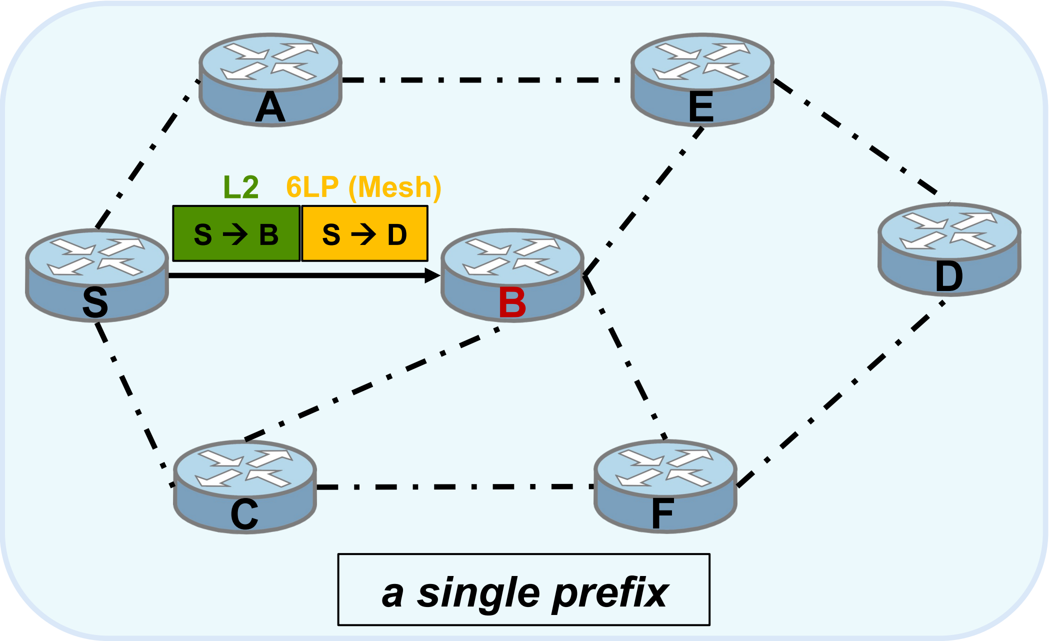


Furthermore, it sets in the L2 header's source address field, its own link-layer address, and it includes the forwarder's, node's B, link-layer address in the L2 header's destination address field.

6LP: **6LoWPAN Adaptation Layer** (i.e., Mesh Addressing Type & Header)

L2: **Layer 2** (e.g., IEEE Std 802.15.4)

Mesh-Under: Operation Example

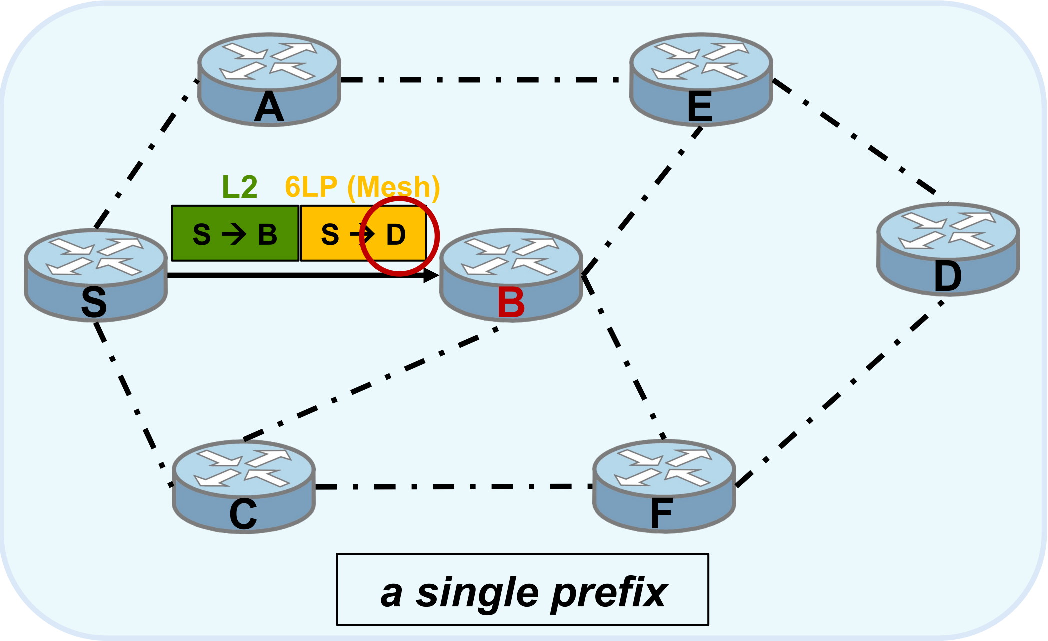


Similarly, when a forwarding node receives a frame,

6LP: **6LoWPAN Adaptation Layer (**i.e., Mesh Addressing Type & Header**)**

L2: **Layer 2 (**e.g., IEEE Std 802.15.4**)**

Mesh-Under: Operation Example

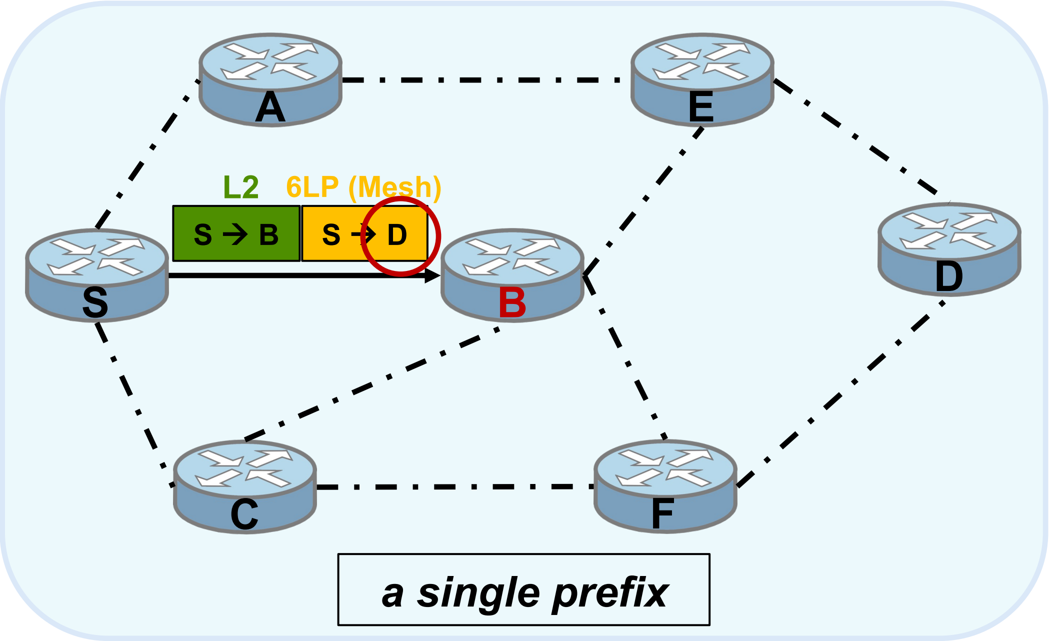


Similarly, when a forwarding node receives a frame, in this case node B, it checks the Mesh Addressing header's "Final Address" field to determine the Final destination according to the following three options:

6LP: 6LoWPAN Adaptation Layer (i.e., Mesh Addressing Type & Header)

L2: Layer 2 (e.g., IEEE Std 802.15.4)

Mesh-Under: Operation Example

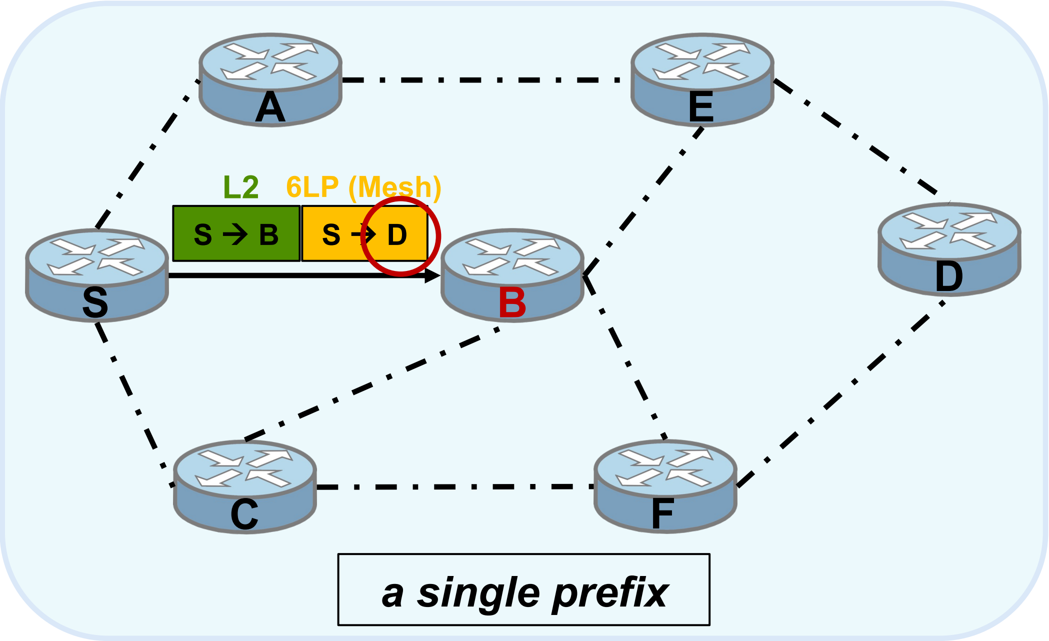


1. If the node *is* the final destination, → it initiates the reassembly of the IPv6 packet.

6LP: **6LoWPAN Adaptation Layer** (i.e., Mesh Addressing Type & Header)

L2: **Layer 2** (e.g., IEEE Std 802.15.4)

Mesh-Under: Operation Example

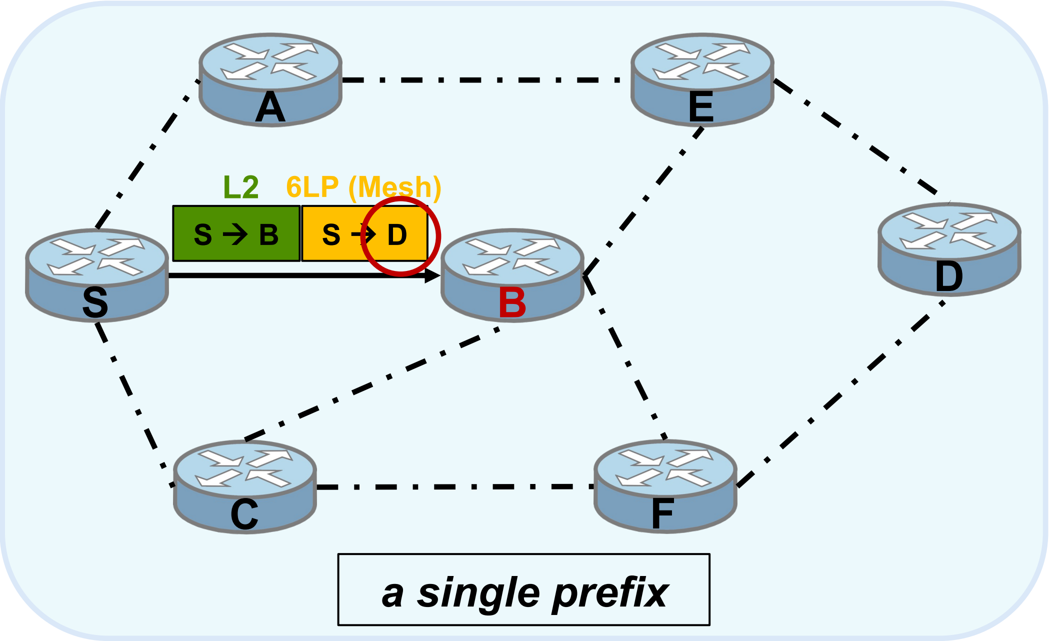


2. If it ***is not***, → it reduces the "Hops Left",

6LP: **6LoWPAN Adaptation Layer (**i.e., Mesh Addressing Type & Header**)**

L2: **Layer 2 (**e.g., IEEE Std 802.15.4**)**

Mesh-Under: Operation Example

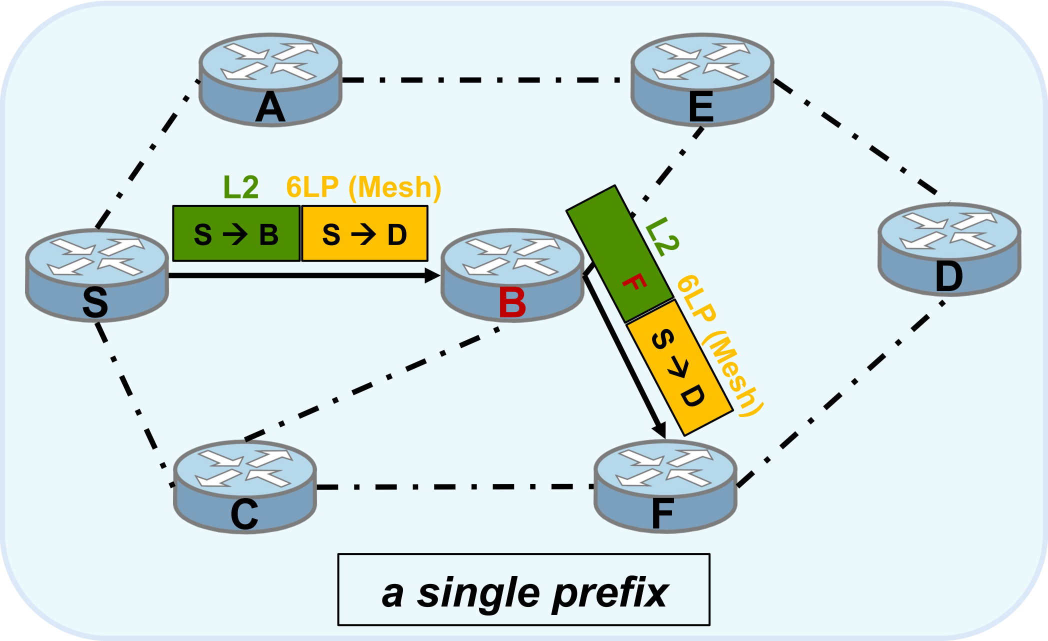


2. If it ***is not***, →
   it reduces the "Hops Left",
   and if it ***decremented to zero***, →
   it **discards** the frame.

6LP: **6LoWPAN Adaptation Layer (**i.e., Mesh Addressing Type & Header**)**

L2: **Layer 2 (**e.g., IEEE Std 802.15.4**)**

Mesh-Under: Operation Example

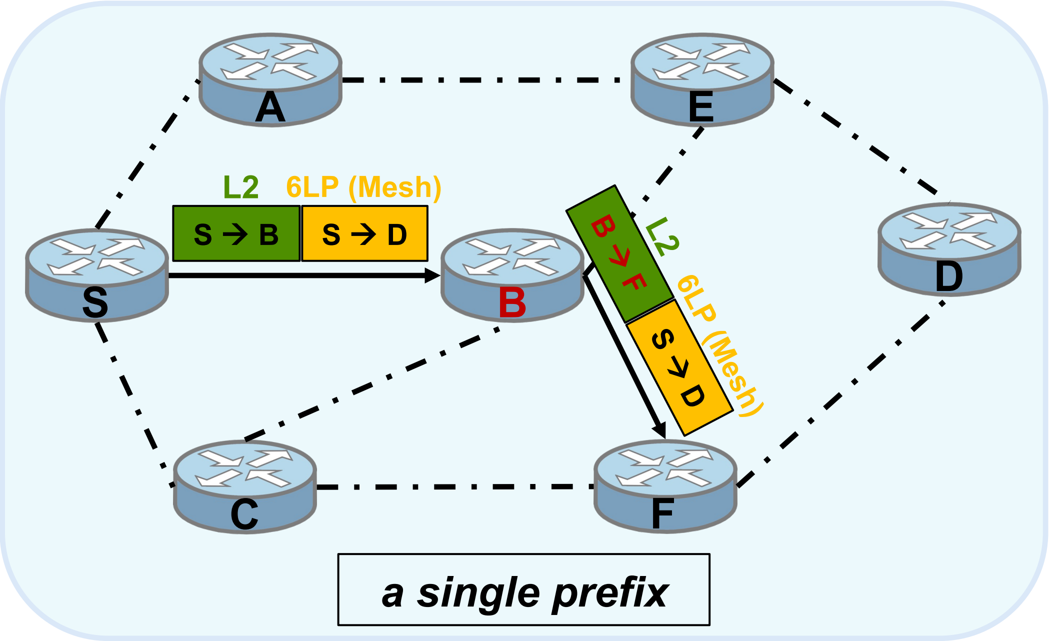


3. Otherwise,
   - it sets **F** in the destination address field **of the L2 header**.

6LP: **6LoWPAN Adaptation Layer** (i.e., Mesh Addressing Type & Header)

L2: **Layer 2** (e.g., IEEE Std 802.15.4)

Mesh-Under: Operation Example

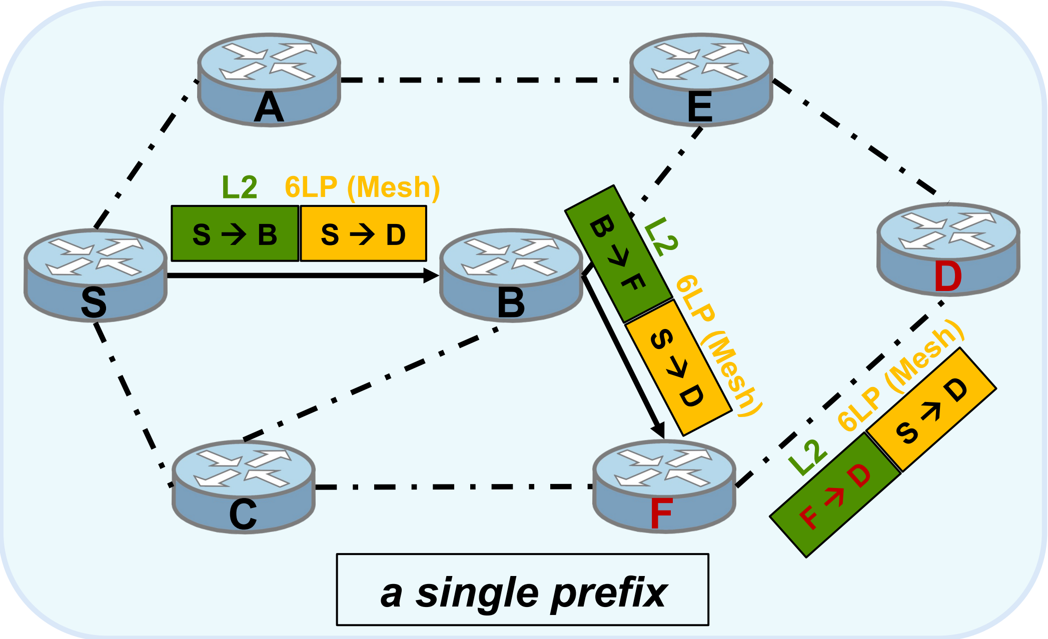


3. Otherwise,
   - it sets **F** in the destination address field **of the L2 header**.
   - Finally, it sets the **L2 source address to its own** and transmits the frame.

6LP: **6LoWPAN Adaptation Layer** (i.e., Mesh Addressing Type & Header)

L2: **Layer 2** (e.g., IEEE Std 802.15.4)

Mesh-Under: Operation Example



These operations are performed for each frame in each intermediate node before an IPv6 packet reaches its destination.

6LP: **6LoWPAN Adaptation Layer** (i.e., Mesh Addressing Type & Header)

L2: **Layer 2** (e.g., IEEE Std 802.15.4)

- 6LoWPAN adaptation layer executes the routing and forwarding:
  - ***No IPv6 header unpacking*** !!!
  - From IPv6, the network (L2) appears as a link (single prefix).

- Forwarding is done via mesh header → ***link layer*** source and destination addresses are employed.

- Originating and Final node specified by either short (16-bit) or EUID (64-bit) IEEE Std 802.15.4 address.

Mesh-Under: Summary (2/2)

- The ***mesh type*** is defined by a 1-bit and 0-bit as the first two bits.
- V:
  - 0: if the Originator (or "Very first") Address is an IEEE extended 64-bit address (EUI-64)
  - 1: if it is a short 16-bit addresses.
- F:
  - 0: if the Final Destination Address is an IEEE extended 64-bit address (EUI-64).
  - 1: if it is a short 16-bit addresses.
- Hops Left*:
  - 4-bit field SHALL be decremented by each forwarding node before sending this packet towards its next hop.
  - The packet is not forwarded if Hops Left is decremented to zero.
- Originator Address: This is the link-layer address of the Originator.
- Final Destination Address: This is the link-layer address of the Final Destin.

* The value 0xF is reserved and signifies an 8-bit Deep Hops Left field immediately following, and allows a source node to specify a hop limit greater than 14 hops.

RFC 4944

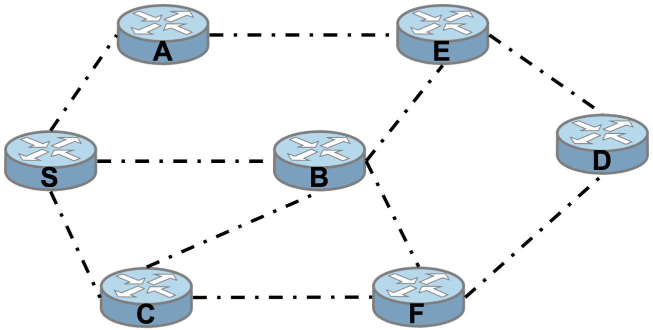
Route-Over:

- The ***routing*** and ***forwarding*** operations are performed at Layer 3 based on IP.

RFC 4944

Route-Over:

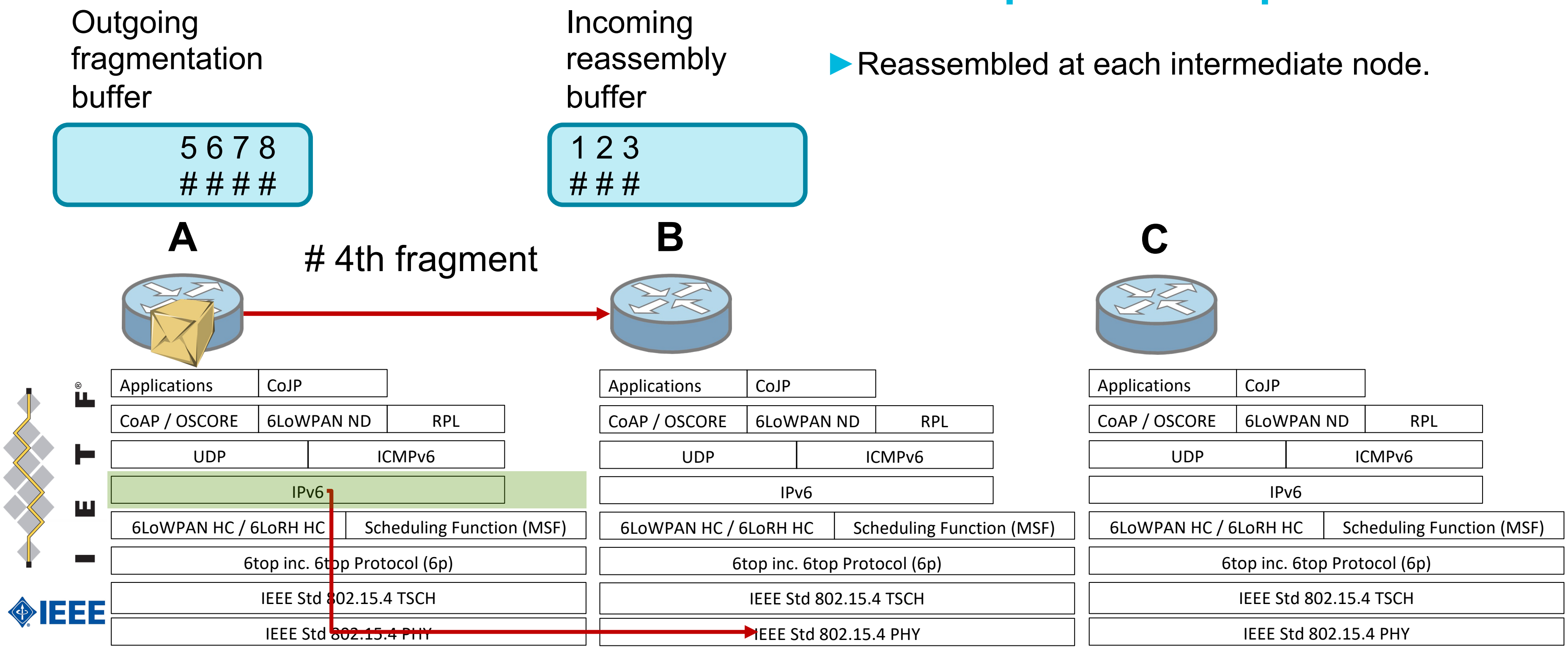
- The ***routing*** and ***forwarding*** operations are performed at Layer 3 based on IP.
- Each link layer hop is an IP hop.



Route-Over: Overview

An IPv6 packet is expected to be:

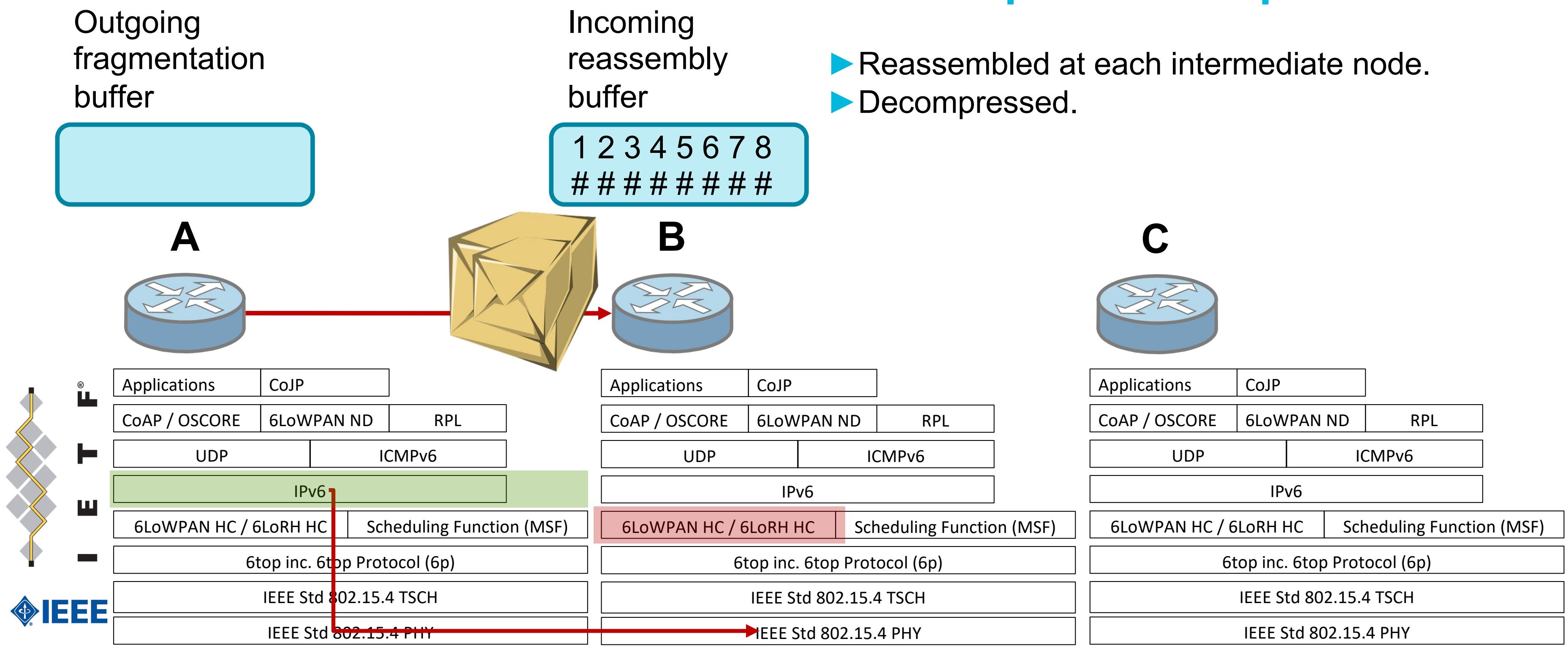
- Reassembled at each intermediate node.



Route-Over: Overview

An IPv6 packet is expected to be:

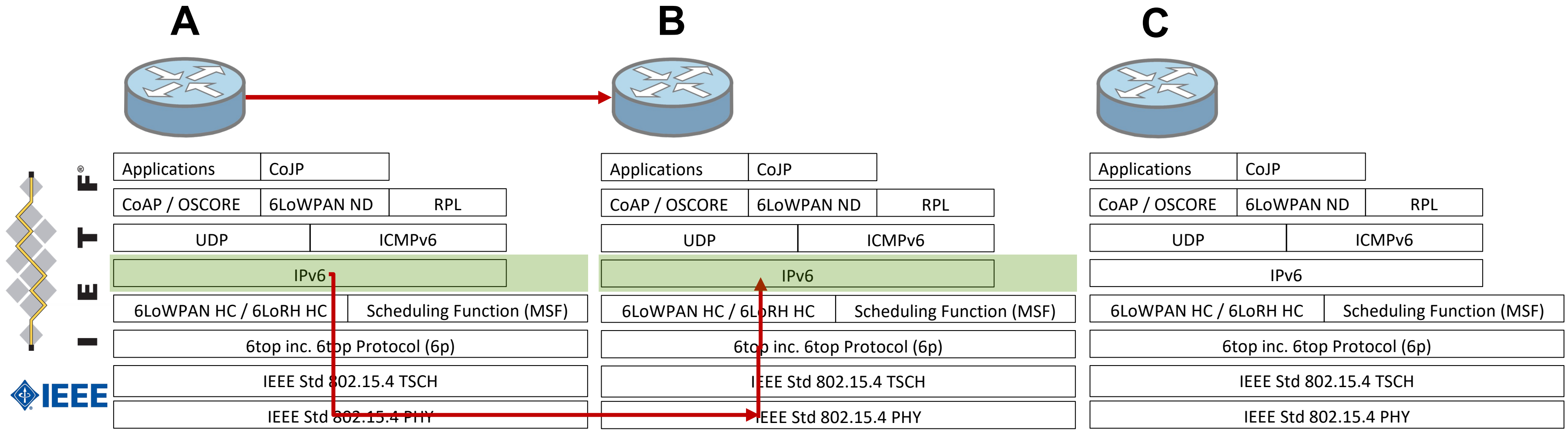
- Reassembled at each intermediate node.
- Decompressed.



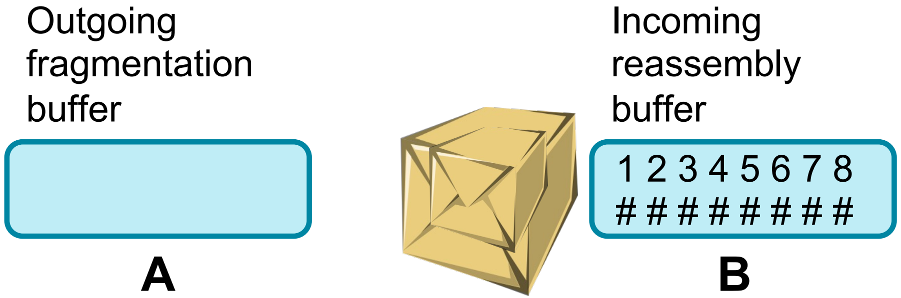
Route-Over: Overview

An IPv6 packet is expected to be:

- ▶ Reassembled at each intermediate node.
- ▶ Decompressed.
- ▶ Pushed to Layer 3 to be routed, i.e., **RPL**.

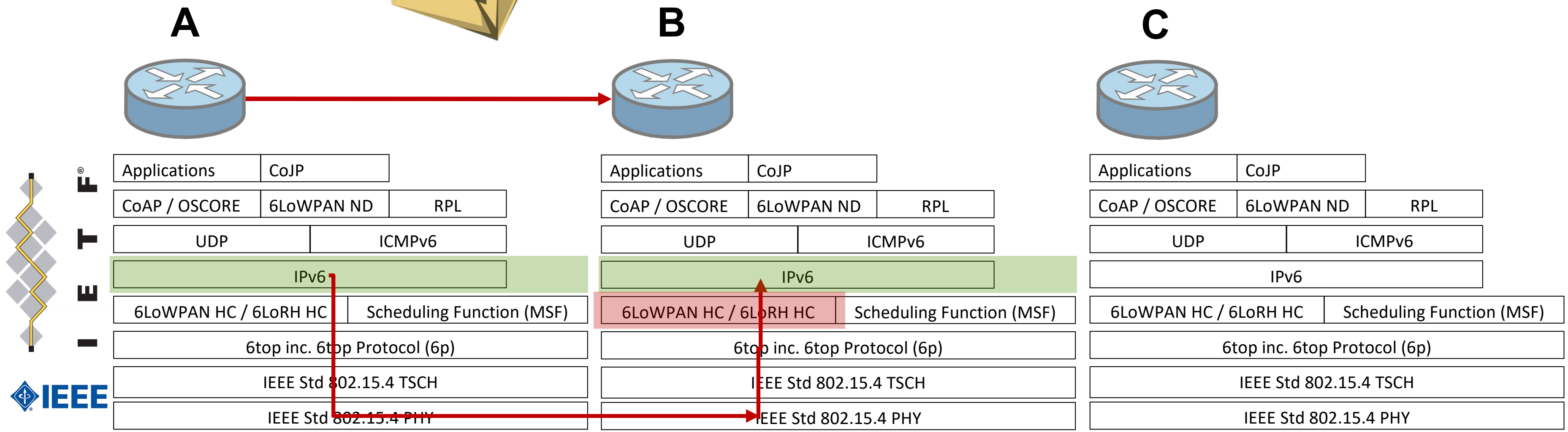


Route-Over: Overview

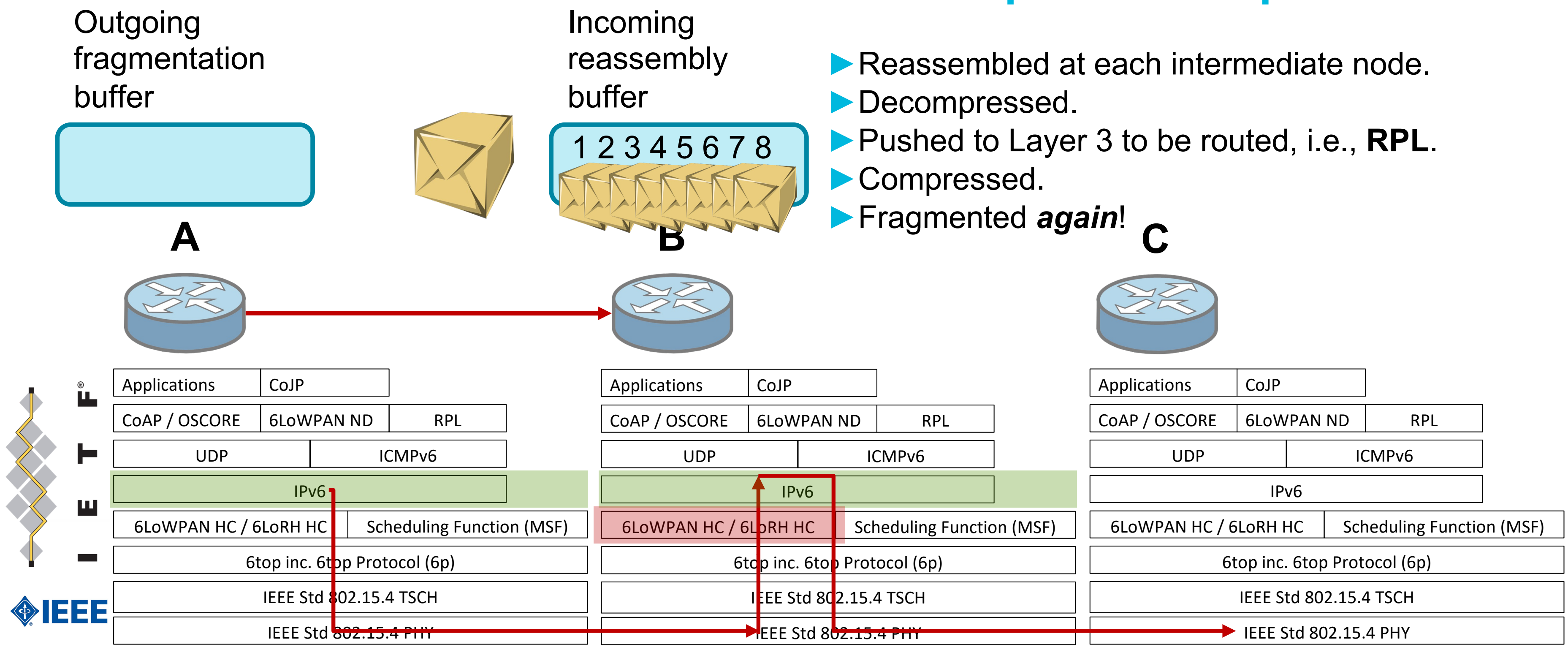


An IPv6 packet is expected to be:

- Reassembled at each intermediate node.
- Decompressed.
- Pushed to Layer 3 to be routed, i.e., **RPL**.
- Compressed.



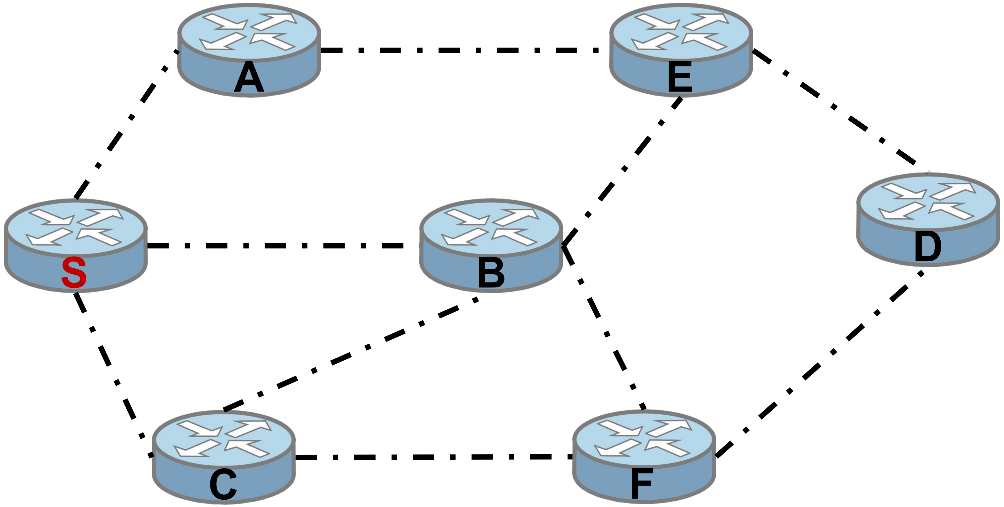
Route-Over: Overview



An IPv6 packet is expected to be:

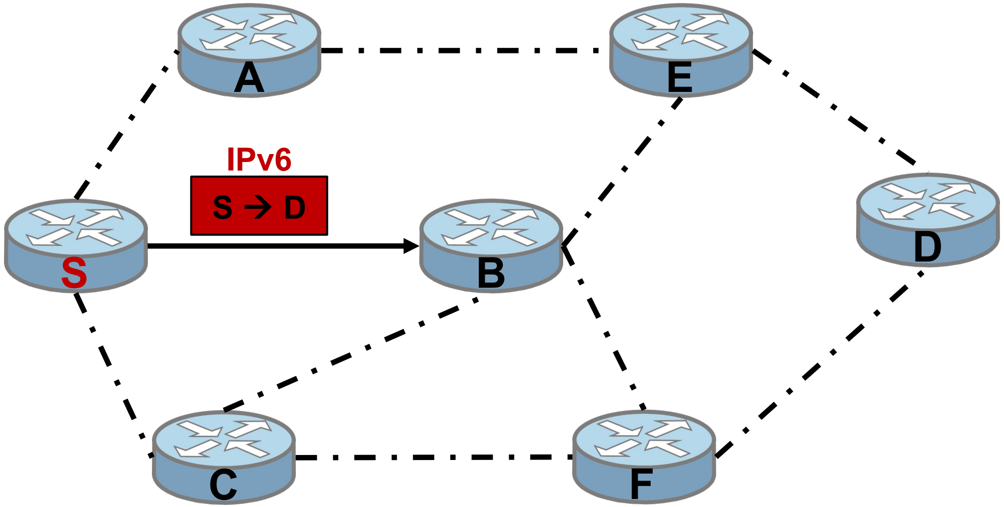
- Reassembled at each intermediate node.
- Decompressed.
- Pushed to Layer 3 to be routed, i.e., **RPL**.
- Compressed.
- Fragmented ***again***!

Route-Over: Operation Example



Considering again the same topology where node S is the source node.

Route-Over: Operation Example

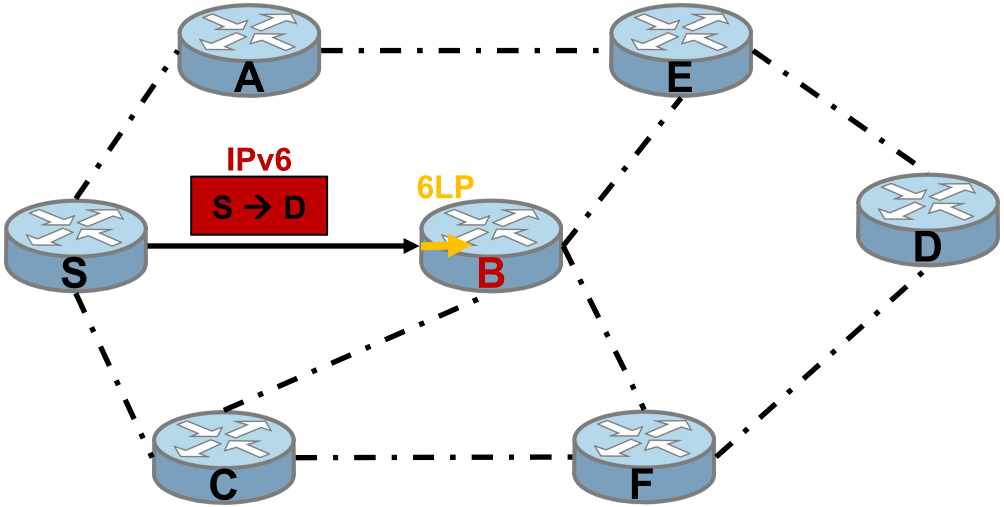


The fragmented IPv6 packet is transmitted to the next hop node B based on the IPv6 header and the routing protocol that is running on top.

IPv6: IPv6 Layer

Institut Mines-Télécom

Route-Over: Operation Example

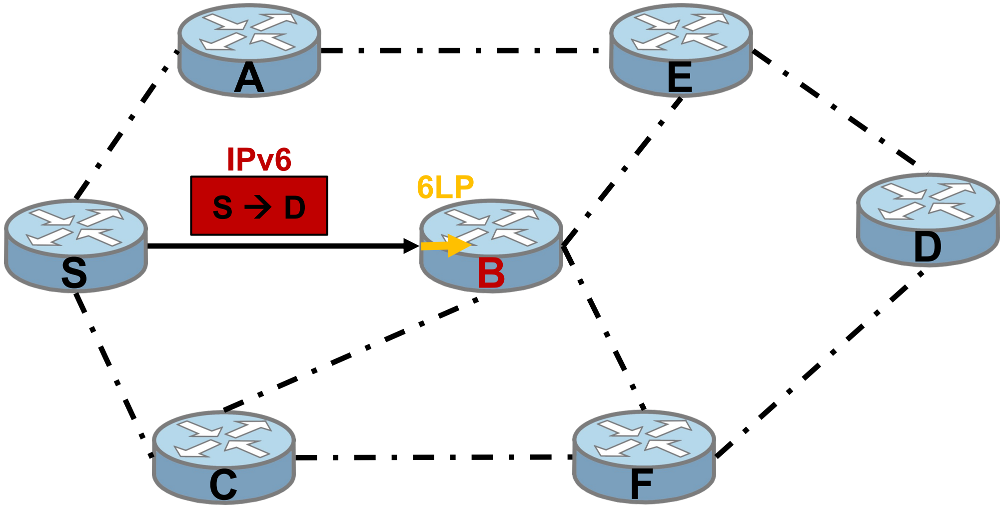


1. Upon the reception of the 1st fragment:
   - node B stores the received fragments.

IPv6: IPv6 Layer

6LP: 6LoWPAN Adaptation Layer

Route-Over: Operation Example

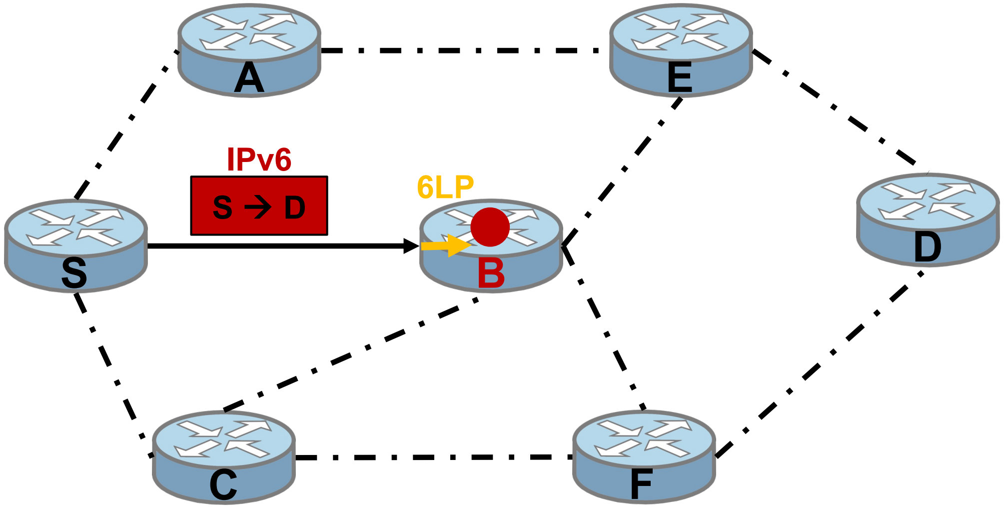


1. Upon the reception of the 1st fragment:
   - node B stores the received fragments.
   - then, it initiates the Reassembly and Decompression operations.

IPv6: IPv6 Layer

6LP: 6LoWPAN Adaptation Layer

Route-Over: Operation Example

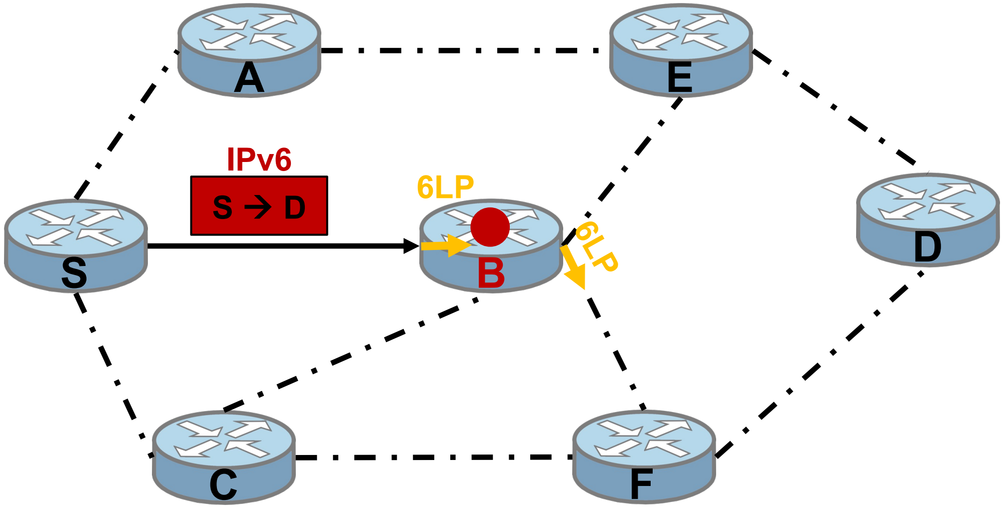


2. Then, it checks the IPv6 original source and final destination headers.

IPv6: **IPv6 Layer**

6LP: **6LoWPAN Adaptation Layer**

Route-Over: Operation Example

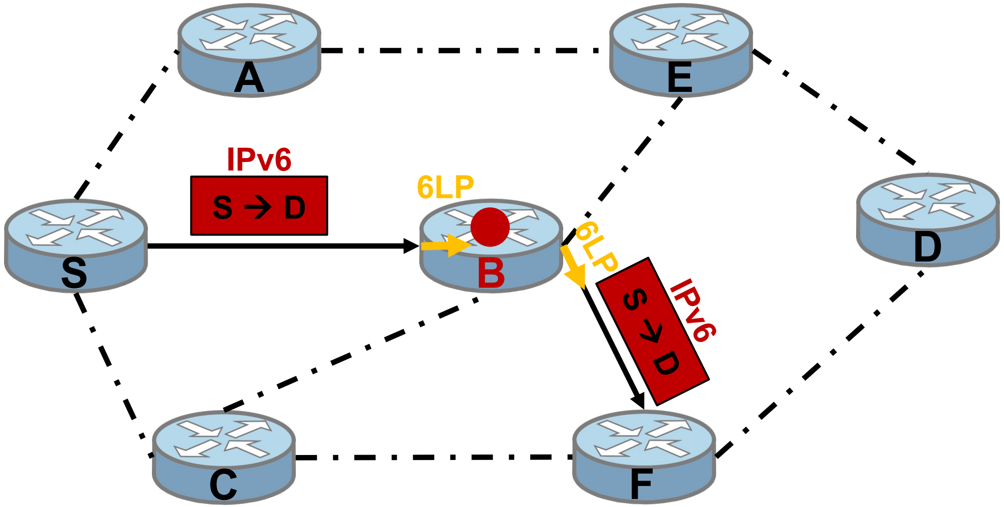


3. If node ***B is not*** the ***final destination***, it pushes to the 6LoWPAN layer:
   - for Compression and Fragmentation operations.

IPv6: **IPv6 Layer**

6LP: **6LoWPAN Adaptation Layer**

Route-Over: Operation Example

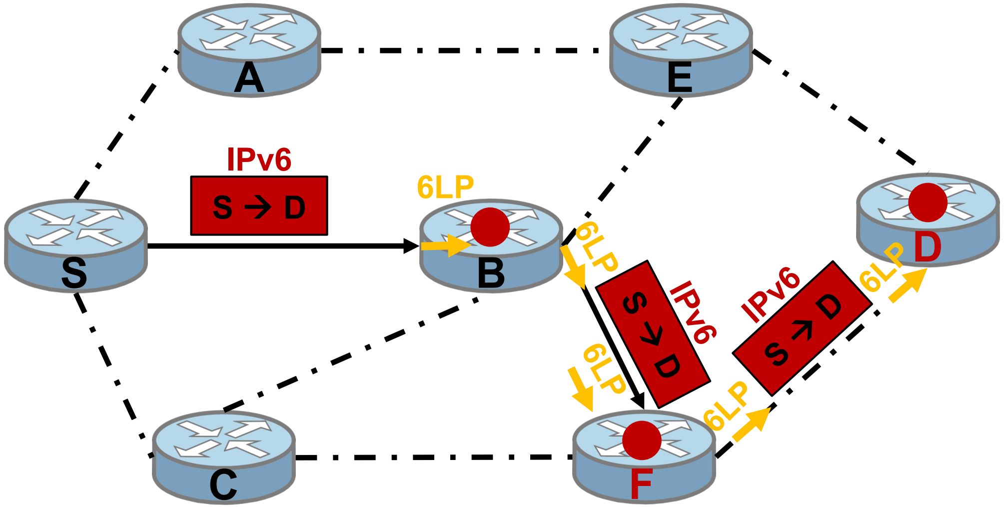


4. Based on its L3 routing table:
   - it will forward the packet to node F.

IPv6: IPv6 Layer

6LP: 6LoWPAN Adaptation Layer

Route-Over: Operation Example



The same procedure will take place in the rest of the path until the final destination is reached.

IPv6: **IPv6 Layer**
6LP: **6LoWPAN Adaptation Layer**

- The frames are routed at the network layer, thus:
  - 6LoWPAN adaptation layer at each hop reassembles the original IPv6 datagram.
  - Then fragment again, before forwarding to the next hop.
  - Routing (L3) protocol is running on the relay/forwarding nodes.

Chapter 3

Route Over (Per-hop Fragmentation & Reassembly): Issues

Check the relevant video "*6LoWPAN Frame Delivery Modes*", from 07:28! on YouTube!

Route-Over: Issues

For more details, please check the following **articles**:

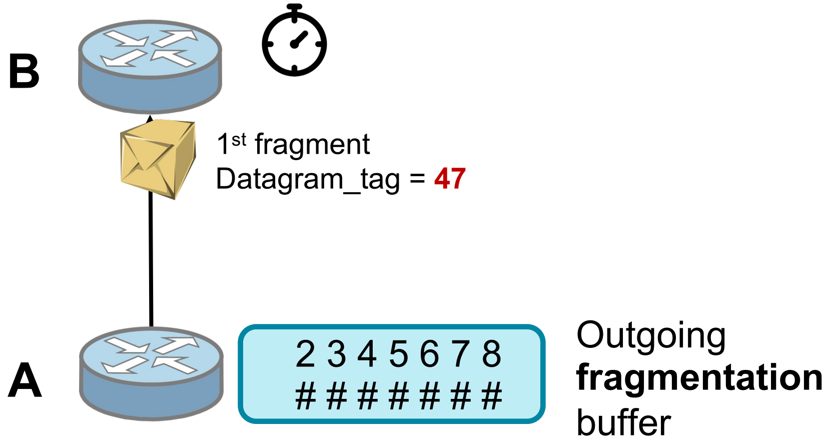
1. *G. Z. Papadopoulos, R. Jadhav, P. Thubert and N. Montavont, "Updates on RFC 4944: Fragment Forwarding and Recovery," In IEEE Communications Standards Magazine, vol. 3, pp. 54-59, June 2019.*
2. *G. Z. Papadopoulos, P. Thubert, S. Tsakalidis and N. Montavont, "RFC 4944: Per-hop Fragmentation and Reassembly Issues" In Proc. IEEE CSCN 2018 - Paris, France, October 2018*

Route-Over: Issues (Reliability)

RFC 4944

Reliability:

- When a node receives the 1st fragment, it initiates the ***reassembly timer*** (i.e., 60 seconds).



RFC 4944

Reliability:

- When a node receives the 1st fragment, it initiates the ***reassembly timer*** (i.e., 60 seconds).
- If **all fragments are not received** during this *reassembly time period*:
  - **the reassembly** of the IPv6 packet **is not possible**.
  - **the received fragments** will be **discarded** while the buffer will be cleared.
  - **even if only one fragment is missing** at the receiver side, **it can not reassemble** the original IPv6 data packet.

Route-Over: Issues (Latency)

RFC 4944

Latency:

- The reassembly at each hop increases the delay:
  - *each device is required to "wait" for 60 seconds.*

	Source	Forwarder 1	Forwarder 2	Destination
T = 0	F F F			
T = 1	F F (F)	F		
T = 2	F (F)	F F		
T = 3	(F)	F F F		
T = 4		F F (F)	F	
T = 5		F (F)	F F	
T = 6		(F)	F F F	
T = 7			F F (F)	F
T = 8			F (F)	F F
T = 9			(F)	F F F

Route-Over: Issues (Latency)

RFC 4944

Latency:

- The reassembly at each hop increases the delay:
  - *each device is required to "wait" for 60 seconds*.
- There is an additional computation time to fragment **again** the previously reassembled IPv6 packet.

	Source	Forwarder 1	Forwarder 2	Destination
T = 0	F F F			
T = 1	F F (F)	F		
T = 2	F (F)	F F		
T = 3	(F)	F F F		
T = 4		F F (F)	F	
T = 5		F (F)	F F	
T = 6		(F)	F F F	
T = 7			F F (F)	F
T = 8			F (F)	F F
T = 9			(F)	F F F

Institut Mines-Télécom

RFC 4944

Latency:

- The reassembly at each hop increases the delay:
  - *each device is required to "wait" for 60 seconds*.
- There is an additional computation time to fragment **again** the previously reassembled IPv6 packet.
- Thus, **the more hops** in the path toward the destination is the **higher** the end-to-end **delay**.

	Source	Forwarder 1	Forwarder 2	Destination
T = 0	F F F			
T = 1	F F (F)	F		
T = 2	F (F)	F F		
T = 3	(F)	F F F		
T = 4		F F (F)	F	
T = 5		F (F)	F F	
T = 6		(F)	F F F	
T = 7			F F (F)	F
T = 8			F (F)	F F
T = 9			(F)	F F F

RFC 4944

Resource Usage:

- The fragmentation process requires **large** incoming reassembly **buffers** (1280 bytes or more).

Route-Over: Issues (Resource Usage)

RFC 4944

Resource Usage:

- The fragmentation process requires **large** incoming reassembly **buffers** (1280 bytes or more).
- To perform complete reassembly at each hop, the forwarder requires at least 1280 bytes to buffer per complete IPv6 datagram.

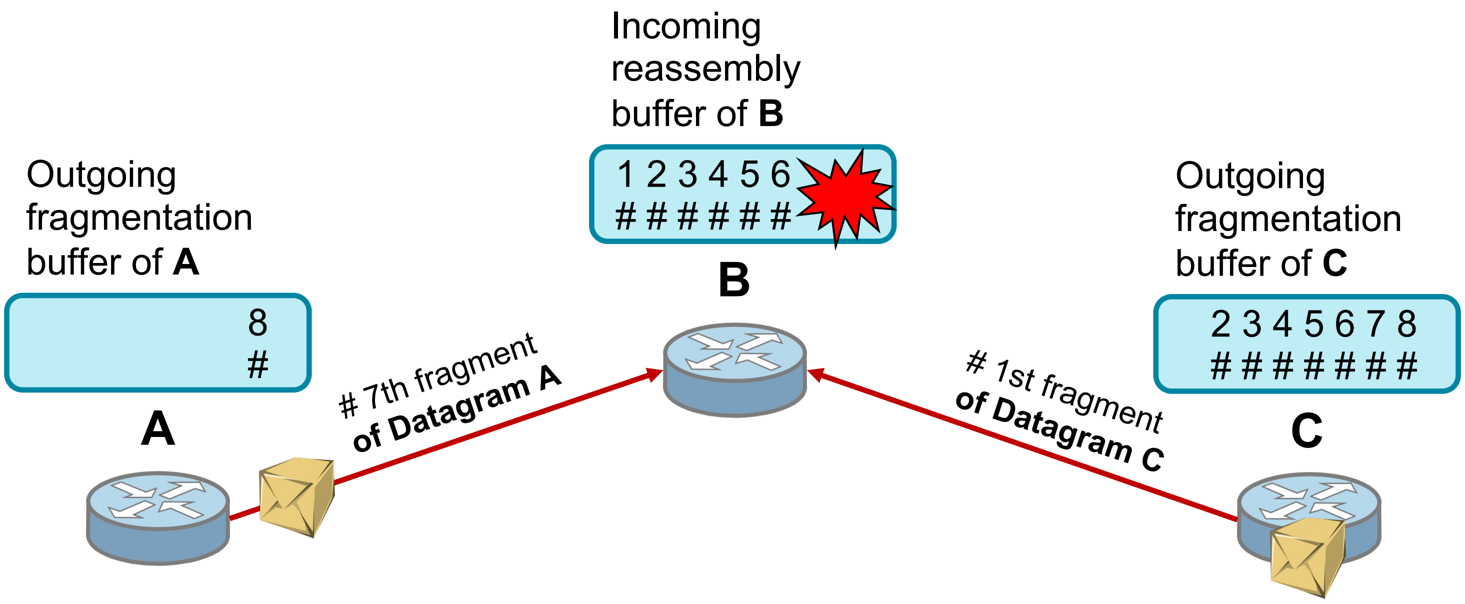
Route-Over: Issues (Resource Usage)

RFC 4944

Resource Usage:

- The fragmentation process requires **large** incoming reassembly **buffers** (1280 bytes or more).
- To perform complete reassembly at each hop, the forwarder requires at least 1280 bytes to buffer per complete IPv6 datagram.
- Considering that the sensor devices are extremely constrained in terms of memory:
  - It will be possible to reassemble only very few complete IPv6 packets.

Route-Over: Issues (Resource Usage)



NB: Therefore, given several consecutive datagrams in the wireless multi-hop network, an ongoing reassembled datagram A may be discarded when a new fragment of datagram C is received, while the previous datagram A has not yet entirely been reassembled.

RFC 4944

Resource Usage:

- The fragmentation process requires **large** incoming reassembly **buffers** (1280 bytes or more).
- To perform complete reassembly at each hop, the forwarder requires at least 1280 bytes to buffer per complete IPv6 datagram.
- Considering that the sensor devices are extremely constrained in terms of memory:
  - It will be possible to reassemble only very few complete IPv6 packets.
- Thus, such issues introduces more losses in a multi-hop network.

Route-Over: Issues (Implementation or *datagram_tag*)

RFC 4944

Implementation:

- Only the **1st fragment** of an IPv6 packet **contains** the source and destination **IPv6 addresses**:

RFC 4944

Implementation:

- Only the **1st fragment** of an IPv6 packet **contains** the source and destination **IPv6 addresses**:
- The subsequent fragments are routed based on the *datagram_tag*.

Route-Over: Issues (Implementation or *datagram_tag*)

RFC 4944

Implementation:

- Only the **1st fragment** of an IPv6 packet **contains** the source and destination **IPv6 addresses**:
- The subsequent fragments are routed based on the *datagram_tag*.
- A *datagram_tag* **is unique only** to the 6LoWPAN original **source** and final **destination nodes**.

RFC 4944

Implementation:

- Only the **1st fragment** of an IPv6 packet **contains** the source and destination **IPv6 addresses**:
- The subsequent fragments are routed based on the *datagram_tag*.
- A *datagram_tag* **is unique only** to the 6LoWPAN original **source** and final **destination nodes**.
- Thus, two different traffic flows may be tagged with the same value.

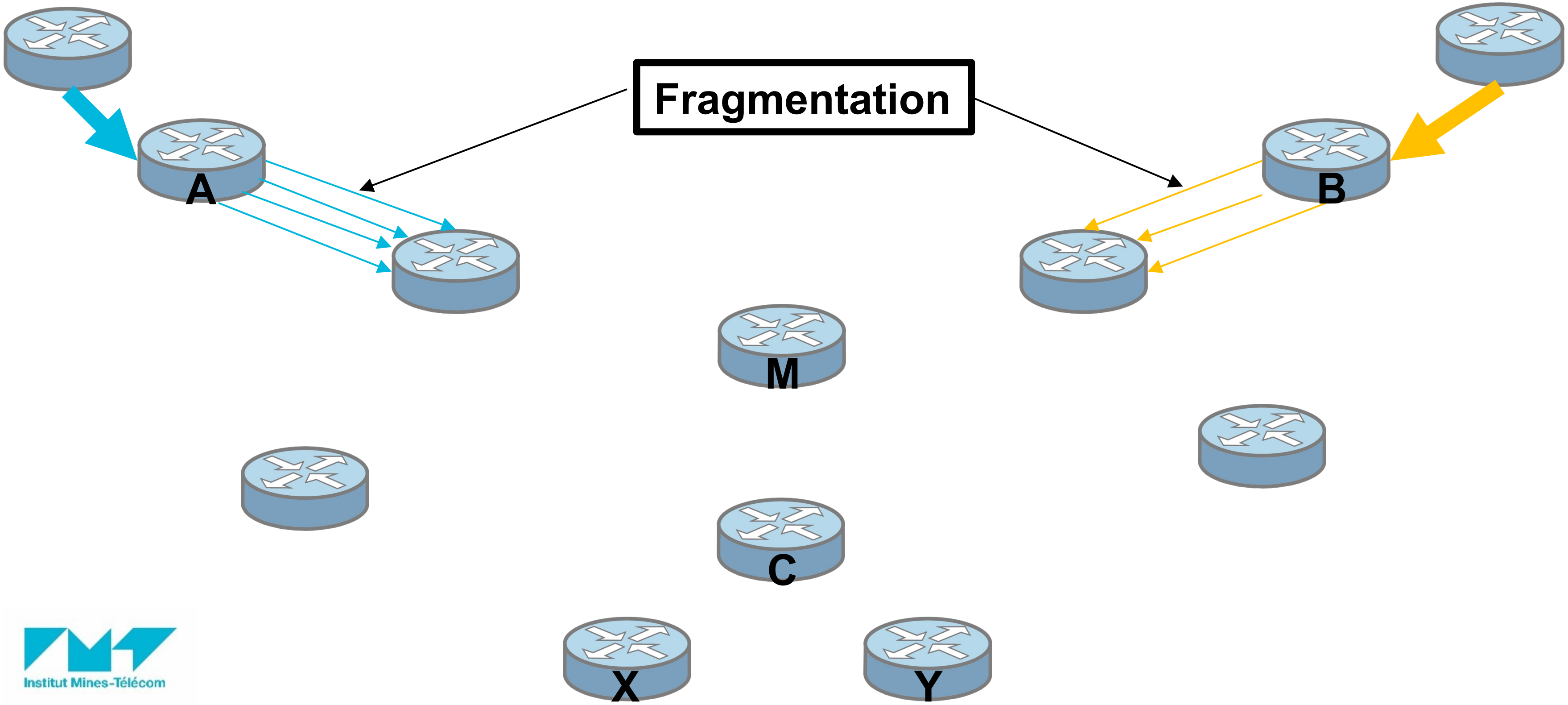
RFC 4944

Implementation:

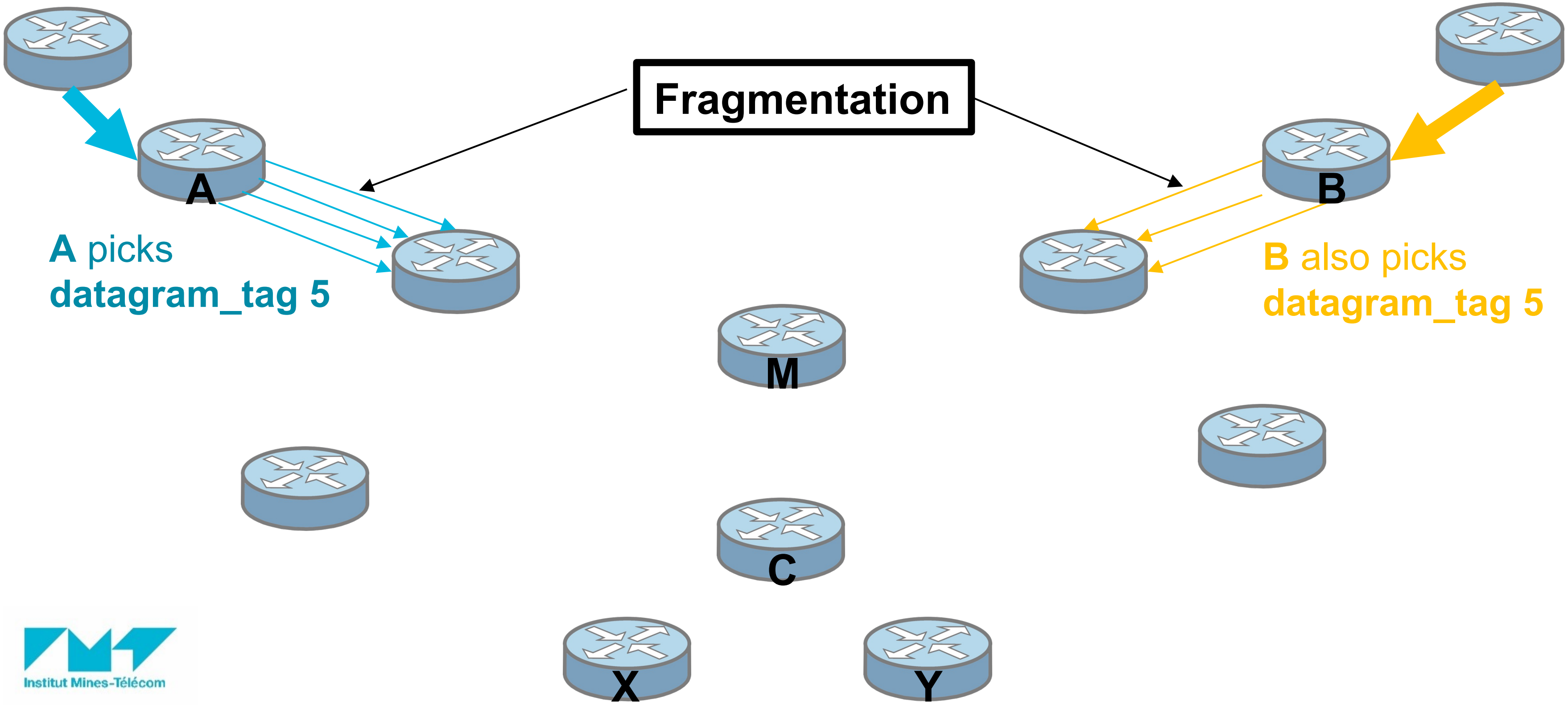
- Only the **1st fragment** of an IPv6 packet **contains** the source and destination **IPv6 addresses**:
- The subsequent fragments are routed based on the *datagram_tag*.
- A *datagram_tag* **is unique only** to the 6LoWPAN original **source** and final **destination nodes**.
- Thus, two different traffic flows may be tagged with the same value.
- As a result, it may cause implementation issues during the IPv6 packet reassemble operation and, thus, during the forwarding phase!

Institut Mines-Télécom

Route-Over: Issues (Implementation or *datagram_tag*)

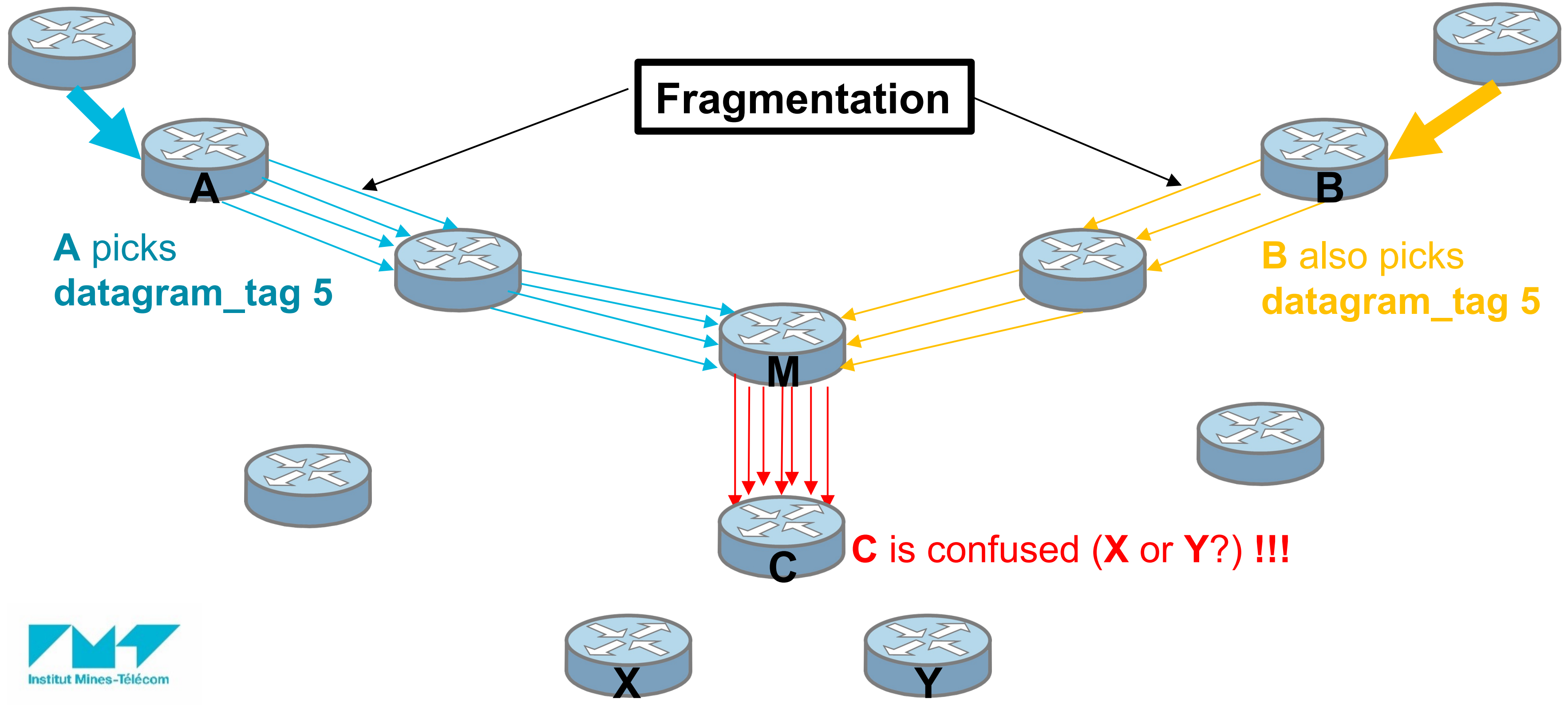


Route-Over: Issues (Implementation or *datagram_tag*)



Institut Mines-Télécom

Route-Over: Issues (Implementation or *datagram_tag*)



Institut Mines-Télécom

Route-Over: Issues (Summary)

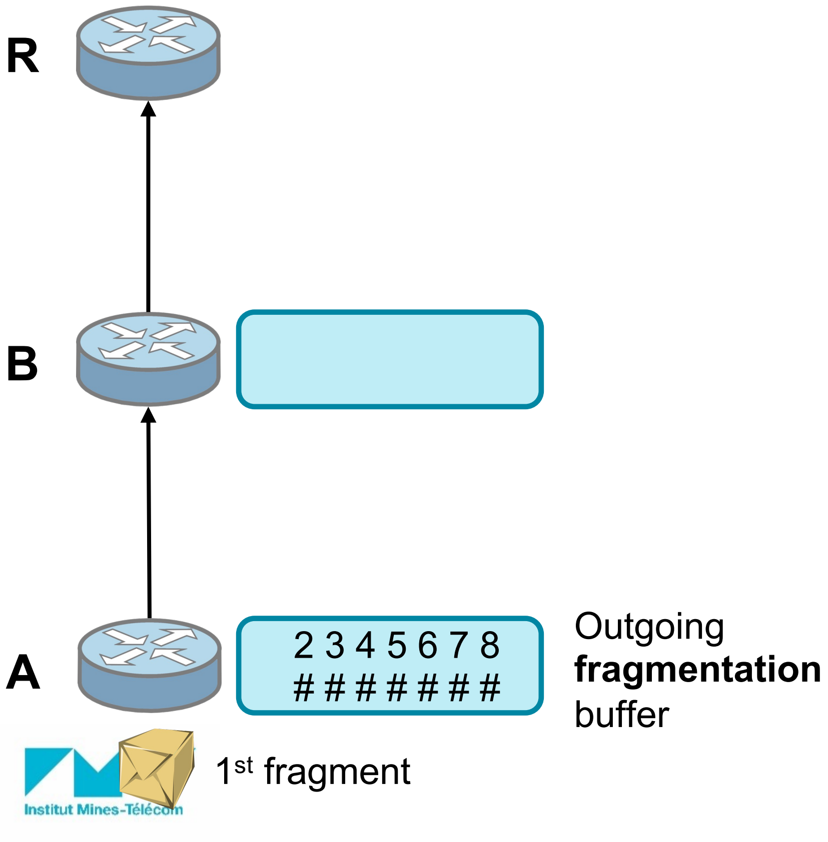
- Network Reliability
- Latency Performance
- Resource Usage
- Implementation of datagram_tag

Chapter 4
RFC 8930: 6LoWPAN Fragment Forwarding (6LFF)

Check the relevant video
"*6LoWPAN Fragment Forwarding*"
on YouTube!

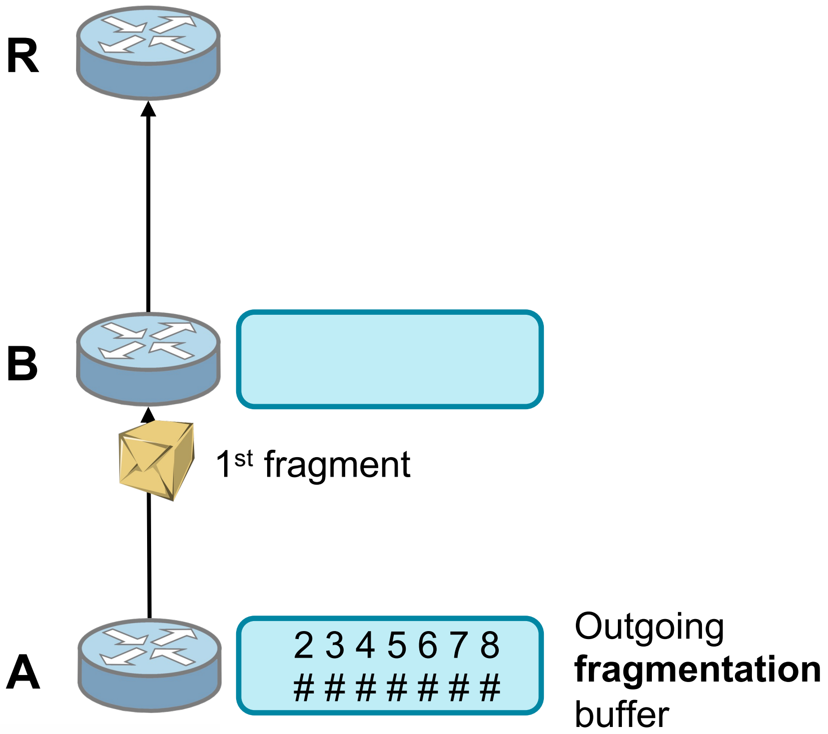
Institut Mines-Télécom

YouTube



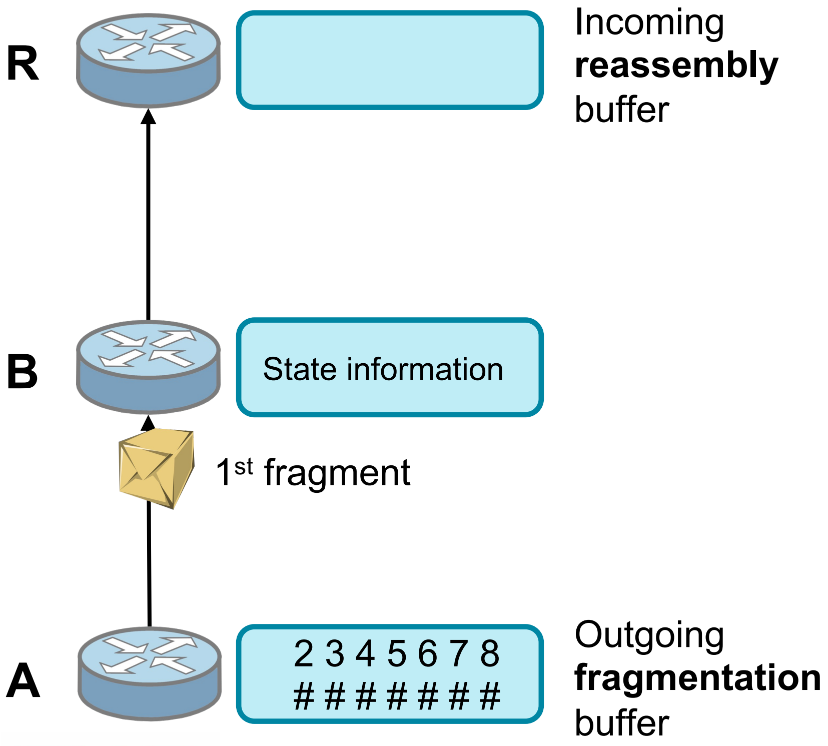
6LFF at a glance:

- The routing decision is made at the 1st fragment.



6LFF at a glance:

- The routing decision is made at the 1st fragment.
- The 1st fragment carries the *IPv6 compressed header*, ***i.e., destination address***.

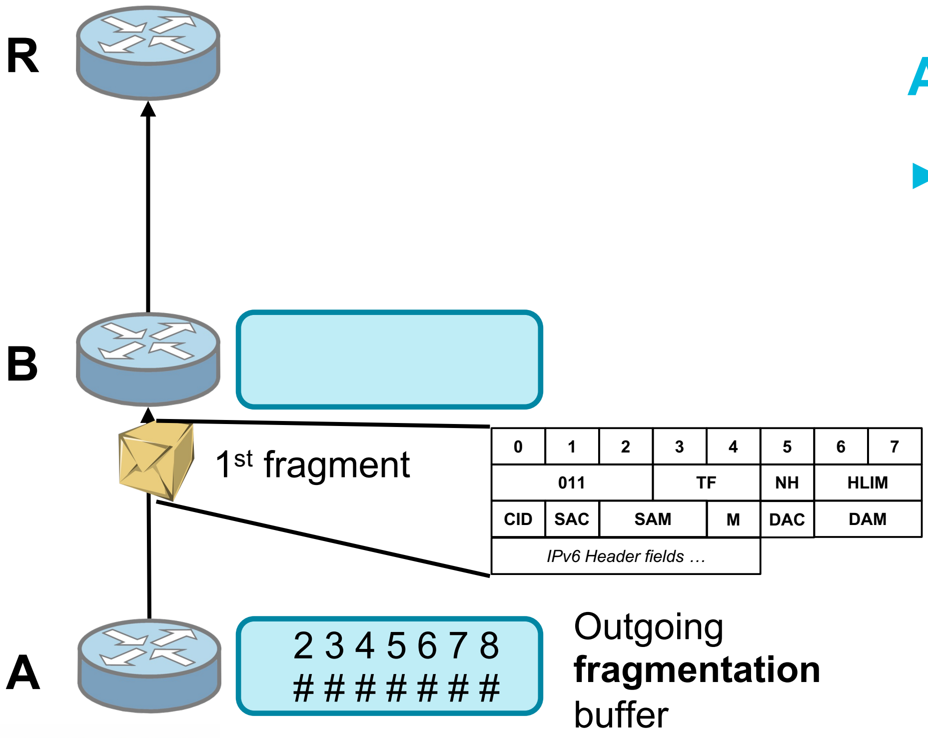


6LFF at a glance:

- The routing decision is made at the 1st fragment.
- The 1st fragment carries the *IPv6 compressed header*, ***i.e., destination address***.
- The received fragments are forwarded immediately! And some state is kept in the intermediate nodes to enable forwarding the subsequent fragments along the same path toward the destination node.

Institut Mines-Télécom

The Operation of the 6LFF Mechanism

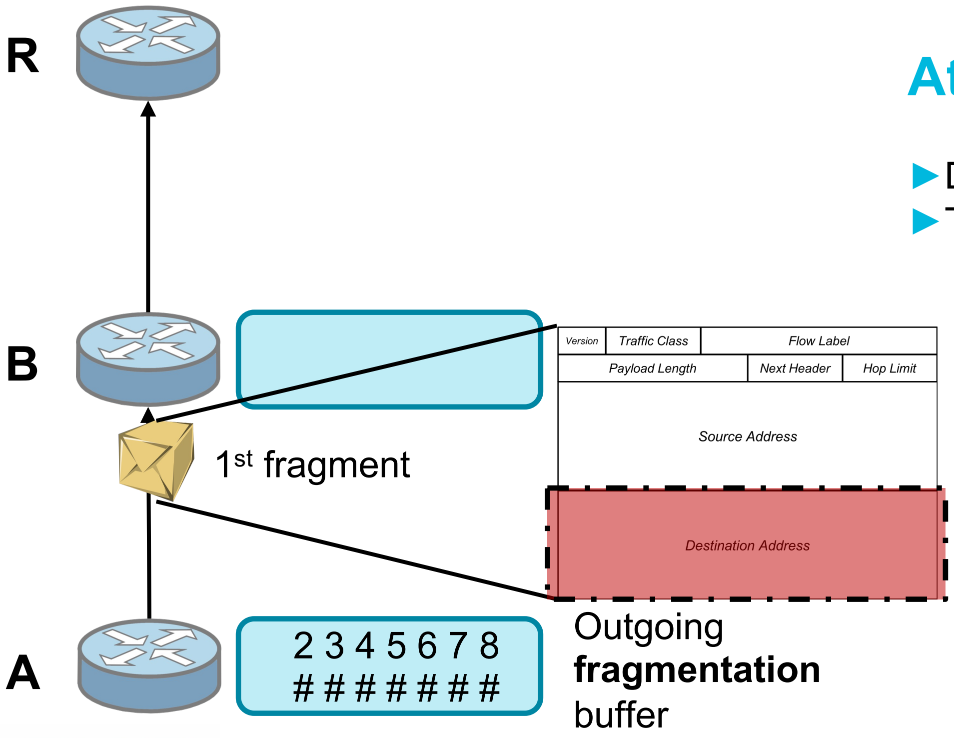


0	1	2	3	4	5	6	7
011			TF		NH	HLIM	
CID	SAC	SAM		M	DAC	DAM	
IPv6 Header fields …							

At the reception of the 1st fragment:

- ►Decompress the ***IPv6 compressed header***.

The Operation of the 6LFF Mechanism

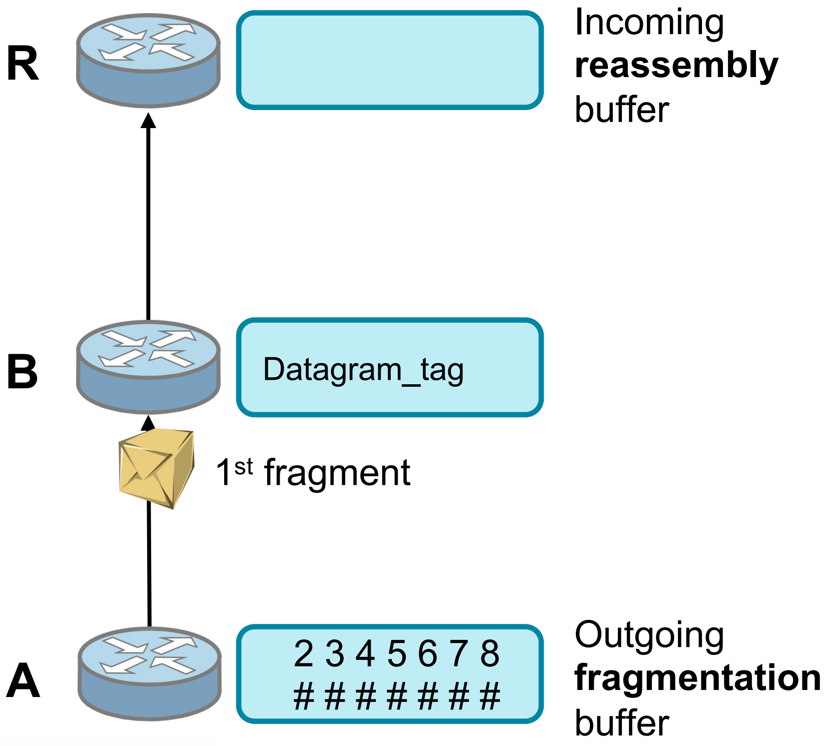


At the reception of the 1st fragment:

- Decompress the ***IPv6 compressed header***.
- To identify the IPv6 address of the next hop.

Institut Mines-Télécom

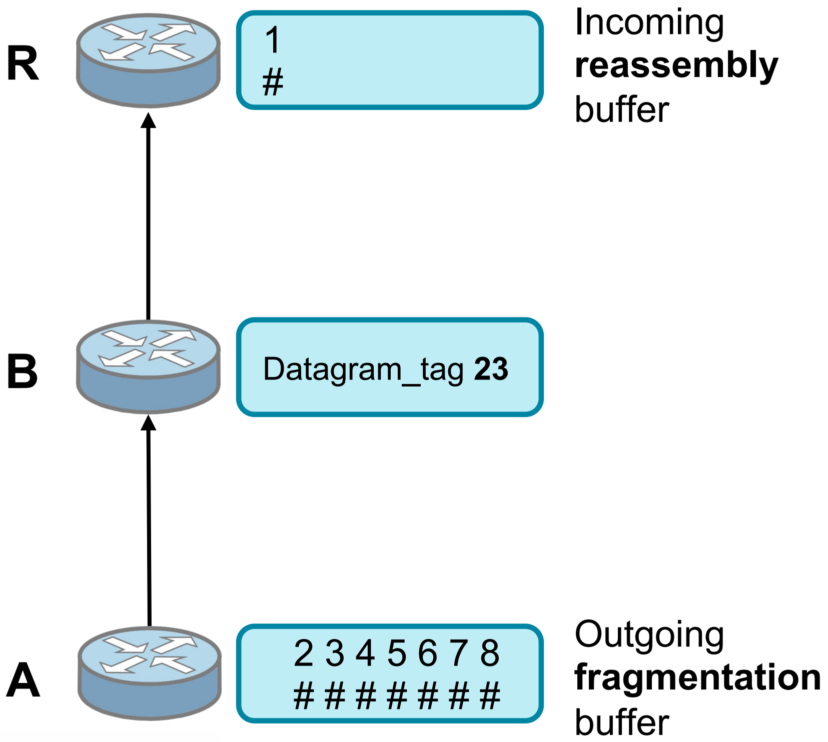
The Operation of the 6LFF Mechanism



At the reception of the 1st fragment:

- Decompress the ***IPv6 compressed header***.
- To identify the IPv6 address of the next hop.
- Forward the fragment to the next hop.

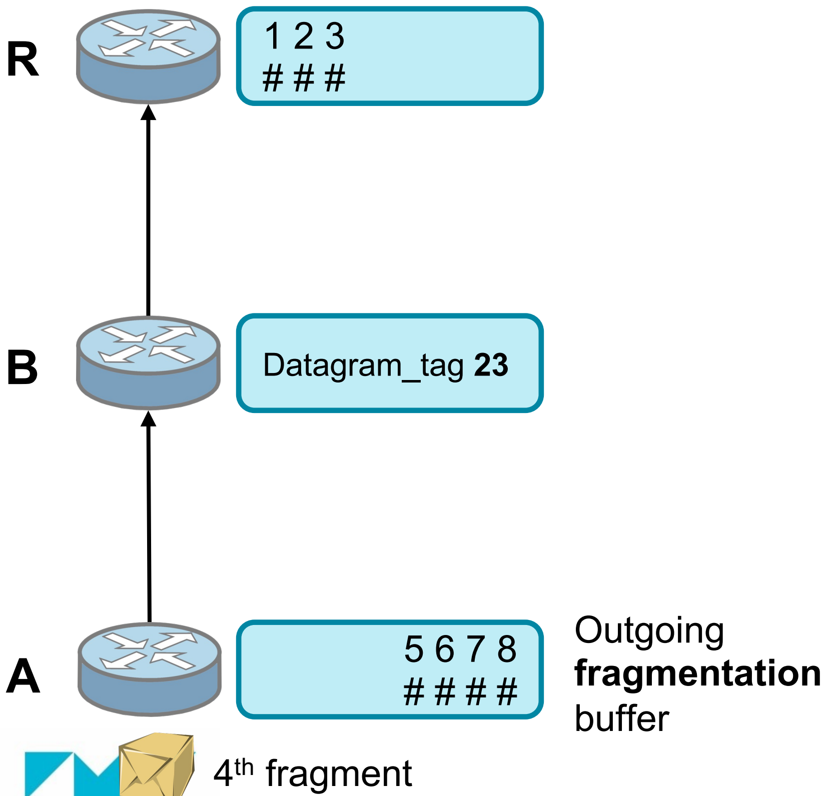
The Operation of the 6LFF Mechanism



At the reception of the 1st fragment:

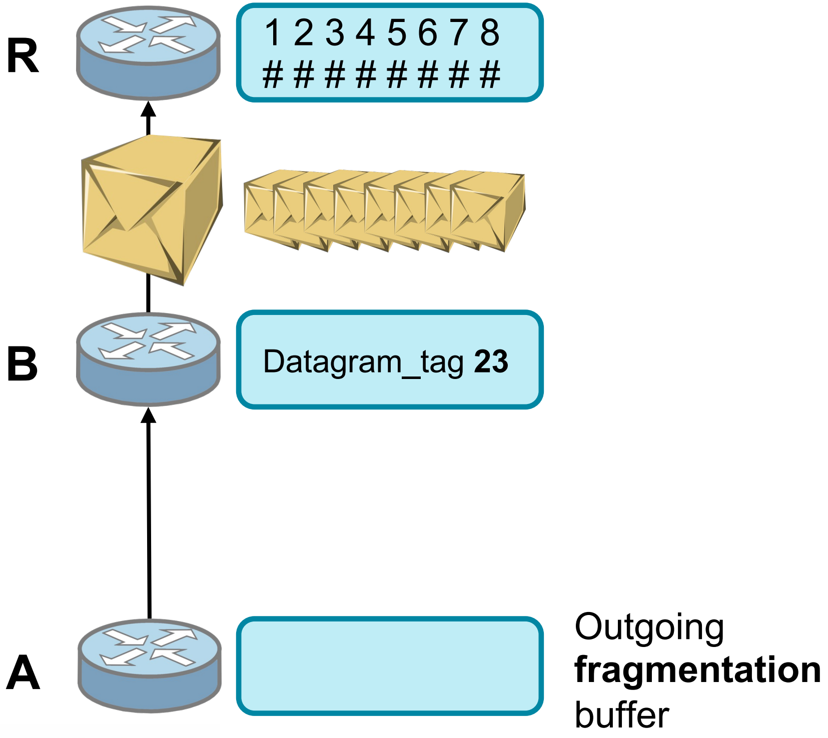
- Decompress the ***IPv6 compressed header***.
- To identify the IPv6 address of the next hop.
- Forward the fragment to the next hop.
- Register the ***datagram_tag*** of the fragment.

The Operation of the 6LFF Mechanism



At the reception of the subsequent fragments:

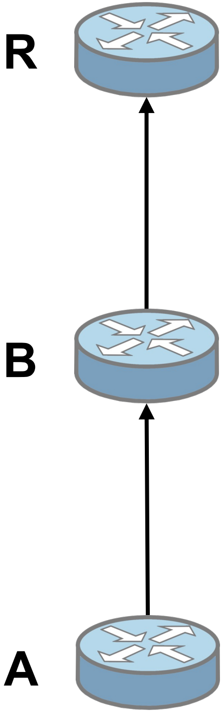
- When receiving subsequent fragments of the same IPv6 packet, which have the same datagram_tag, the node forwards them to the same next hop.



At the reception of the subsequent fragments:

- When receiving subsequent fragments of the same IPv6 packet, which have the same datagram_tag, the node forwards them to the same next hop.
- The IPv6 packet is reassembled when **all** the fragments have arrived at the destination.

Virtual Reassembly Buffer (VRB)



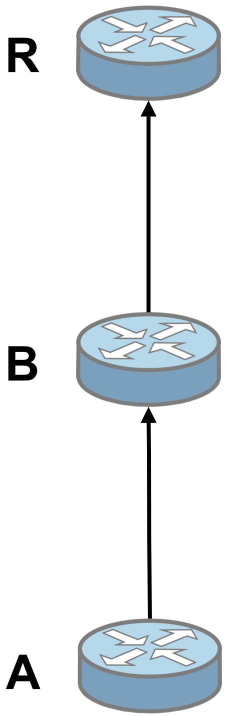
Node **B**'s VRB table.

incoming		outgoing	

VRB at a glance:

- The operation is similar to ***switching table***.

Virtual Reassembly Buffer (VRB)



Node **B**'s VRB table.

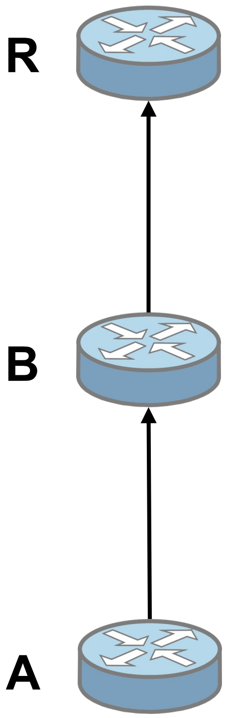
incoming	outgoing
entry 1 (IPv6 packet 1)	
entry 2 (IPv6 packet 2)	
entry … (IPv6 packet …)	
entry *n* (IPv6 packet *n*)	

VRB at a glance:

- The operation is similar to ***switching table***.
- The entries correspond to IPv6 packets.

Institut Mines-Télécom

Virtual Reassembly Buffer (VRB)



Node **B**’s VRB table.

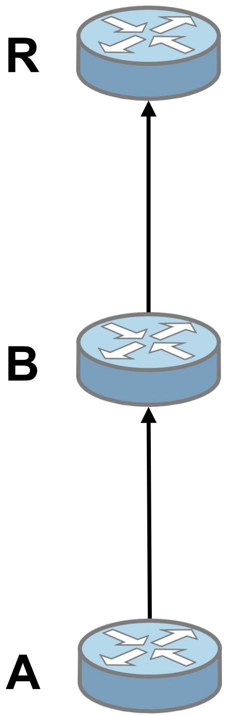
incoming		outgoing	
entry 1 (empty)			
entry 2 (empty)			
entry … (empty)			
entry *n* (empty)			

VRB at a glance:

- ▶The operation is similar to ***switching table***.
- ▶The entries correspond to IPv6 packets.
- ▶The VRBs have a maximum pre-allocated memory.

Institut Mines-Télécom

Virtual Reassembly Buffer (VRB)



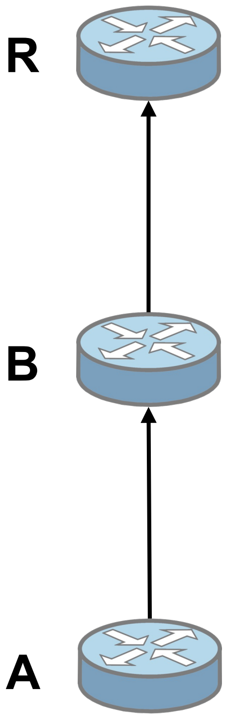
Node **B**’s VRB table.

incoming		outgoing	

A VRB entry:

- Each VRB entry is a tuple of 4 elements:

Virtual Reassembly Buffer (VRB)



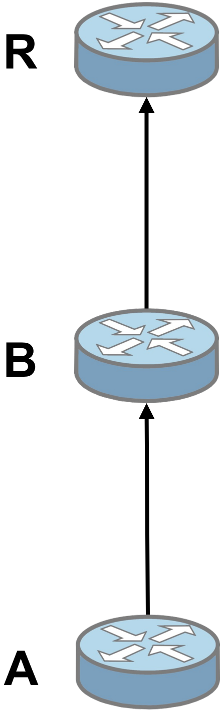
Node **B**'s VRB table.

incoming		outgoing	
L2 src			

A VRB entry:

- Each VRB entry is a tuple of 4 elements:
  - link-layer address of the previous hop.

Virtual Reassembly Buffer (VRB)



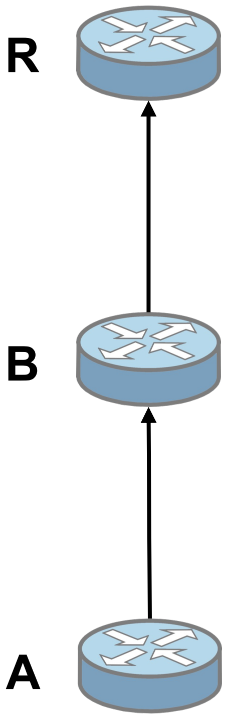
Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag		

A VRB entry:

- Each VRB entry is a tuple of 4 elements:
  - link-layer address of the previous hop.
  - locally unique datagram_tag of the incoming fragment.

Virtual Reassembly Buffer (VRB)



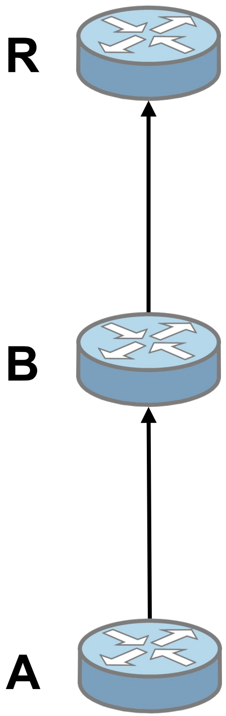
Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	

A VRB entry:

- Each VRB entry is a tuple of 4 elements:
  - link-layer address of the previous hop.
  - locally unique datagram_tag of the incoming fragment.
  - link-layer address of the next hop.

Virtual Reassembly Buffer (VRB)



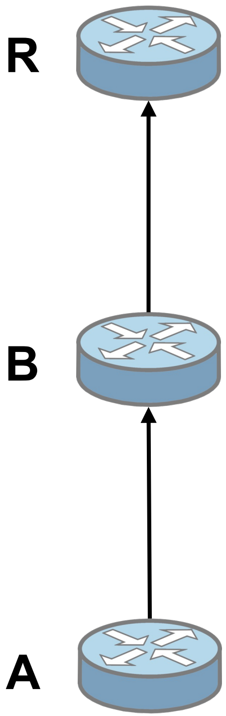
Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag

A VRB entry:

- Each VRB entry is a tuple of 4 elements:
  - link-layer address of the previous hop.
  - locally unique datagram_tag of the incoming fragment.
  - link-layer address of the next hop.
  - locally unique datagram_tag for the outgoing fragment.

Virtual Reassembly Buffer (VRB)



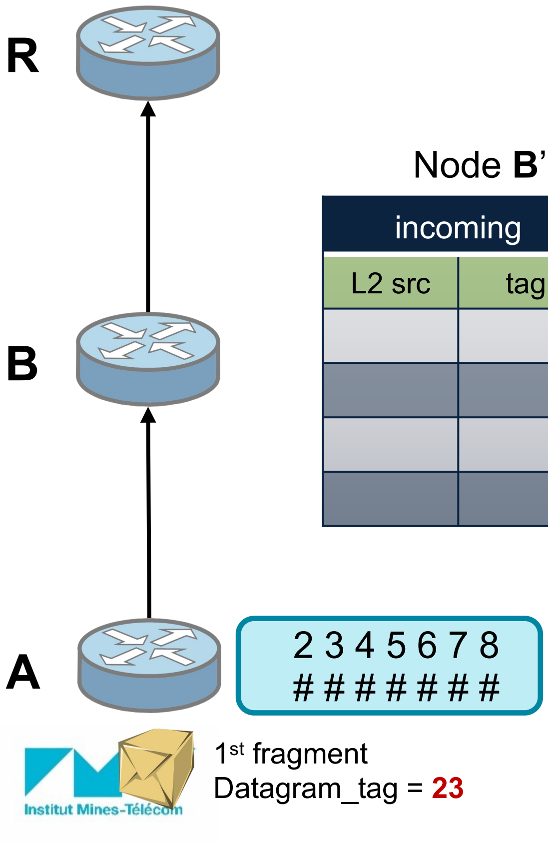
Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag

A VRB entry:

- Each VRB entry is a tuple of 4 elements:
  - link-layer address of the previous hop.
  - locally unique datagram_tag of the incoming fragment.
  - link-layer address of the next hop.
  - locally unique datagram_tag for the outgoing fragment.
- Each VRB entry requires 20 bytes.

VRB Operation



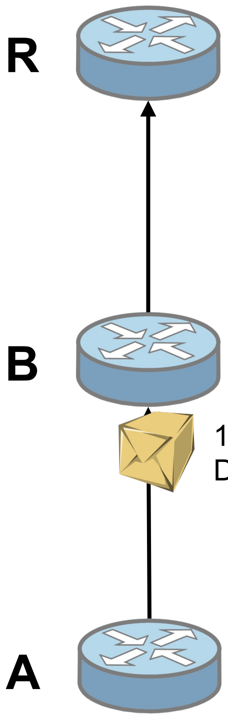
Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag

Outgoing **fragmentation** buffer

At the reception of the 1st fragment:

VRB Operation



Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23		

1st fragment
Datagram_tag = 23

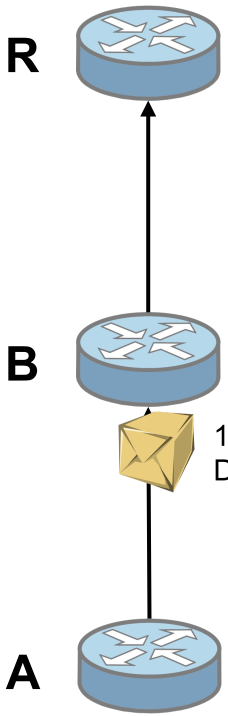
2 3 4 5 6 7 8
\# # # # # # #

Outgoing **fragmentation** buffer

At the reception of the 1st fragment:

- Record the *link-layer address* and *datagram_tag* of the **previous hop**.

VRB Operation



Node **B**’s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	

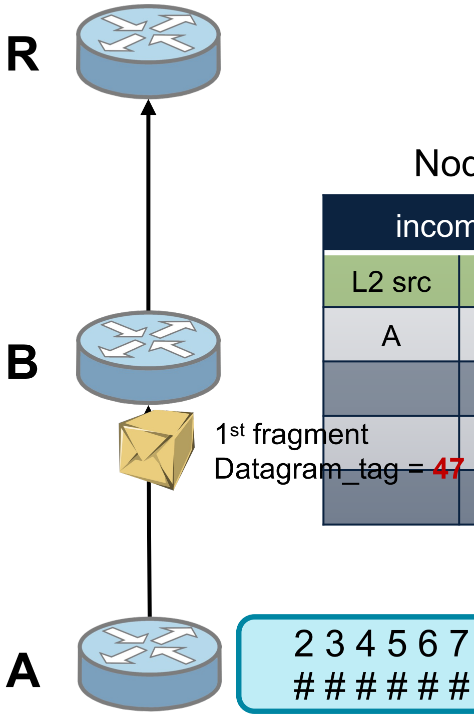
2 3 4 5 6 7 8
# # # # # # #

Outgoing **fragmentation** buffer

At the reception of the 1st fragment:

- Record the *link-layer address* and *datagram_tag* of the **previous hop**.
- Determine the *link-layer address* of the **next hop**.

VRB Operation



Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	47

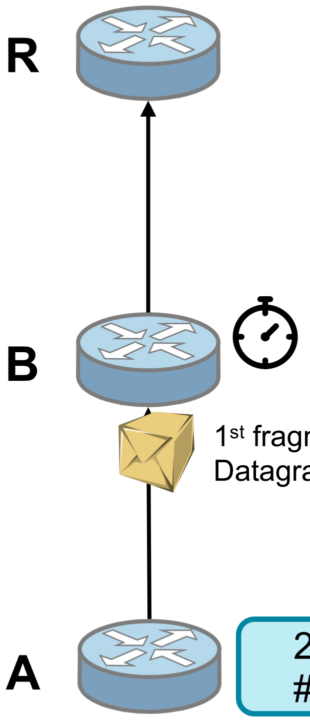
2 3 4 5 6 7 8
# # # # # # #

Outgoing **fragmentation** buffer

At the reception of the 1st fragment:

- Record the *link-layer address* and *datagram_tag* of the **previous hop**.
- Determine the *link-layer address* of the **next hop**.
- Pick a **new unique** *datagram_tag* for the **next hop**.

VRB Operation



Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	47

2 3 4 5 6 7 8
\# # # # # # #

Outgoing **fragmentation** buffer

At the reception of the 1st fragment:

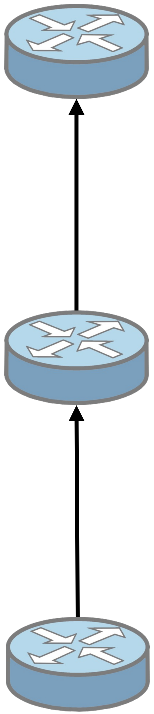
- Record the *link-layer address* and *datagram_tag* of the **previous hop**.
- Determine the *link-layer address* of the **next hop**.
- Pick a **new unique** *datagram_tag* for the **next hop**.
- Set a timer to allow discarding a **partially** reassembled packet after some timeout.

VRB Operation

R

B

A



1 2 3
# # #

Incoming **reassembly** buffer

Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	47

5 6 7 8
# # # #

Outgoing **fragmentation** buffer

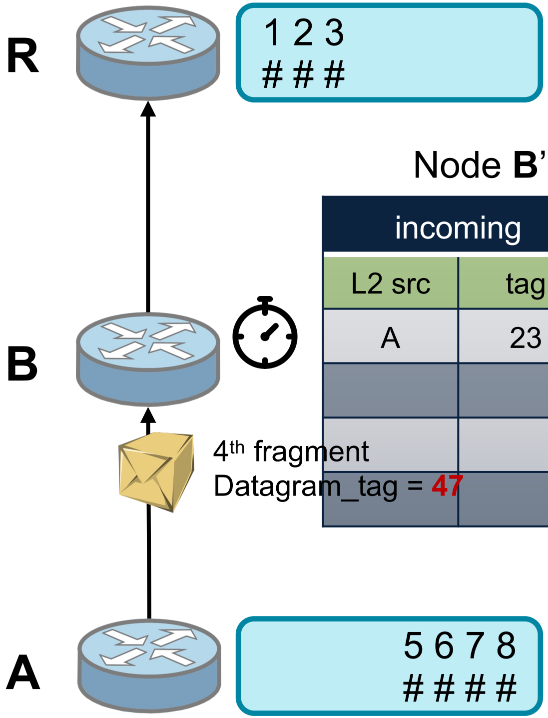
4th fragment
Datagram_tag = **23**

RFC 8930

Then, all subsequent fragments of the same IPv6 packet will go through the same process!

Institut Mines-Télécom

VRB Operation



Incoming **reassembly** buffer

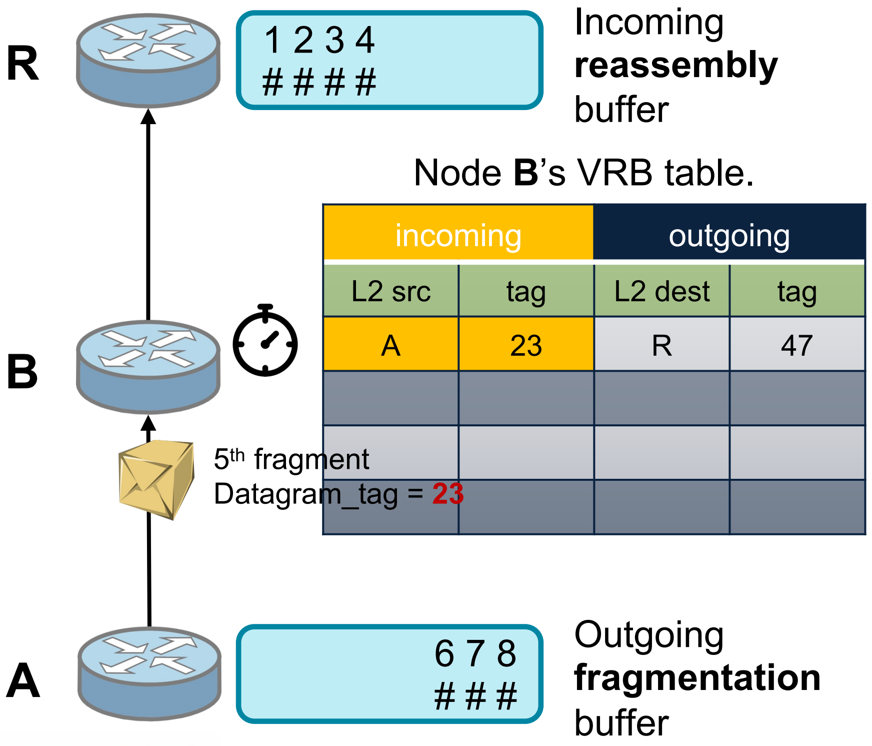
Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	47

Outgoing **fragmentation** buffer

RFC 8930

VRB Operation

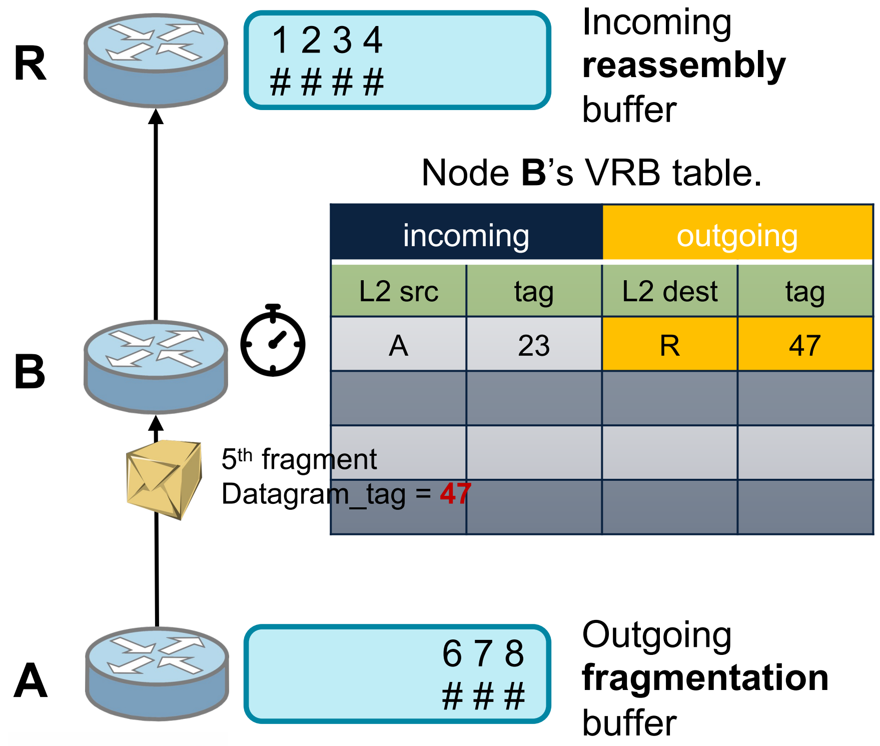


Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	47

More specifically, the intermediate node for each subsequent fragment will search its source link-layer address and datagram tag in the “incoming” columns of the VRB table

VRB Operation

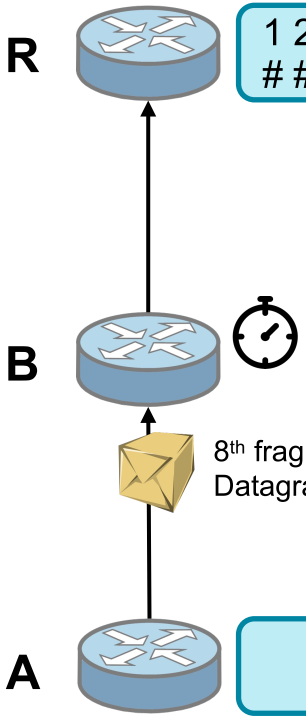


Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	47

More specifically, the intermediate node for each subsequent fragment will search its source link-layer address and datagram tag in the "incoming" columns of the VRB table *and forward it based on the "outgoing" columns*.

VRB Operation



1 2 3 4 5 6 7
# # # # # # #

Incoming **reassembly** buffer

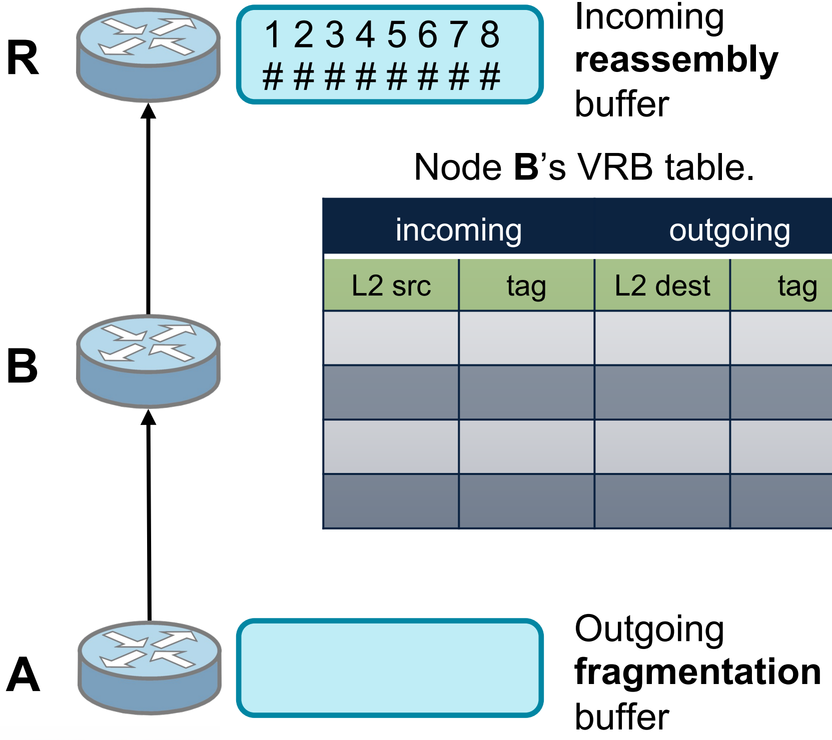
Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag
A	23	R	47

Outgoing **fragmentation** buffer

Finally, upon forwarding the last fragment,

VRB Operation

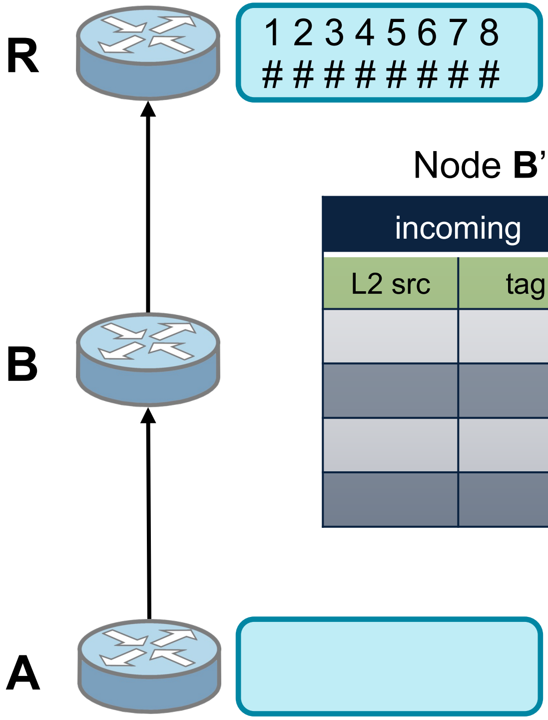


Node **B**'s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag

Finally, upon forwarding the last fragment, *the node clears the VRB entry from its table.*

VRB Operation



Incoming **reassembly** buffer

Node **B**’s VRB table.

incoming		outgoing	
L2 src	tag	L2 dest	tag

Outgoing **fragmentation** buffer

As a result, the VRB technique allows intermediate nodes to immediately forward the received fragments, without reassembling the complete IPv6 packet first.

Advantages

- The **end-to-end latency** should be greatly reduced.

RFC 8930

Advantages

RFC 8930

- The **end-to-end latency** should be greatly reduced.
- The **end-to-end network reliability** should be improved.

Advantages

RFC 8930

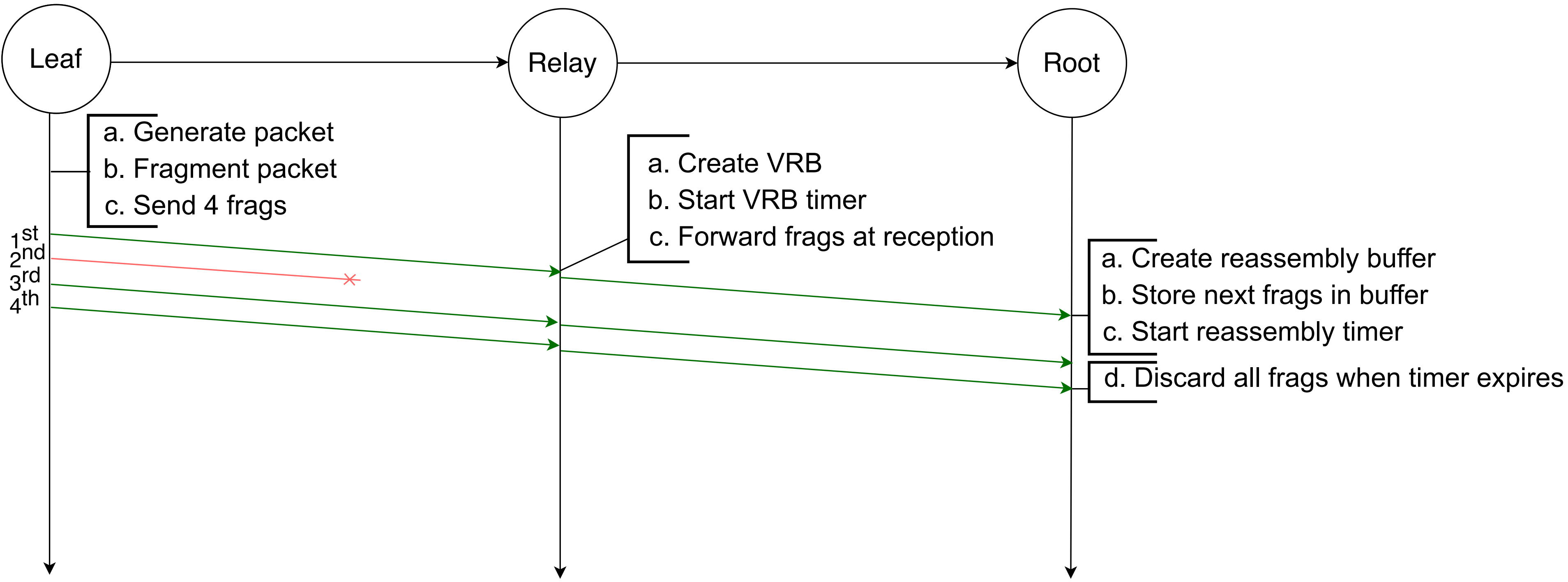
- The **end-to-end latency** should be greatly reduced.
- The **end-to-end network reliability** should be improved.
- The **datagram_tag issue** is solved.

- **No Fragment Recovery**: in case a single fragment is lost along the multi-hop path, then there is no mechanism for the node that reassembles an IPv6 packet to request for it.

- Thus, missing even a single fragment will eventually introduce unnecessary traffic, since the remaining fragments are forwarded toward the destination, even it can never reassemble the data packet.

- Moreover, it will require the whole IPv6 packet (i.e., all fragments) to be retransmitted from the source node.

Drawbacks: No Fragment Recovery **(1/3)**



Drawbacks: No Per-Fragment Routing **(2/3)**

- **No Per-Fragment Routing**: all follow-up fragments must follow the same path to the destination as the first fragment, since only the 1st fragment contains the IPv6 destination address.

- A side effect is that the first fragment must always be forwarded first.

- **Non-zero Packet Drop Probability**: the size of the VRB table is necessarily finite.
  - Thus, data packets are dropped, when a node must concurrently forward more data packets than the entries in its VRB table.

RFC 8930

To conclude, I want you to remember that 6LoWPAN Fragment Forwarding (6LFF) is an implementation technique, not a new protocol. That makes it fully compatible with the Per-hop Fragmentation and Reassembly of the original 6LoWPAN Route Over mode of RFC 4944.

- VRB allows a node to immediately forward the received fragments, without reassembling the complete IPv6 packet first.

- Each node in the network maintains a VRB table where the entries correspond to forwarding IPv6 packets.

- In the beginning, all VRB tables of all nodes are empty, though they do have a maximum pre-allocated memory.

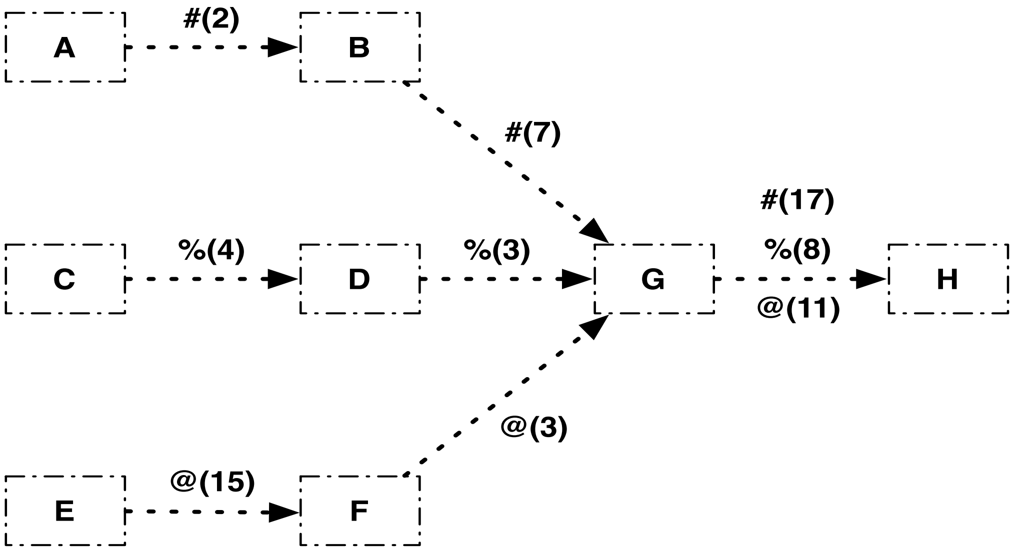
6LFF: Summary **(2/4)**

- A relay node, once receiving the first fragment of a data packet, it will register the datagram_tag of the fragment in its pre-allocated memory (VRB), and it will determine the next hop based on the IPv6 address contained in that fragment, as well as it will pick a new datagram_tag.

- Then, all subsequent fragments of this packet (with the same received datagram_tag) will be forwarded through the same outgoing address with the new datagram_tag.

- Finally, only at the final destination, the fragments are reassembled.

6LFF: Summary (3/4)

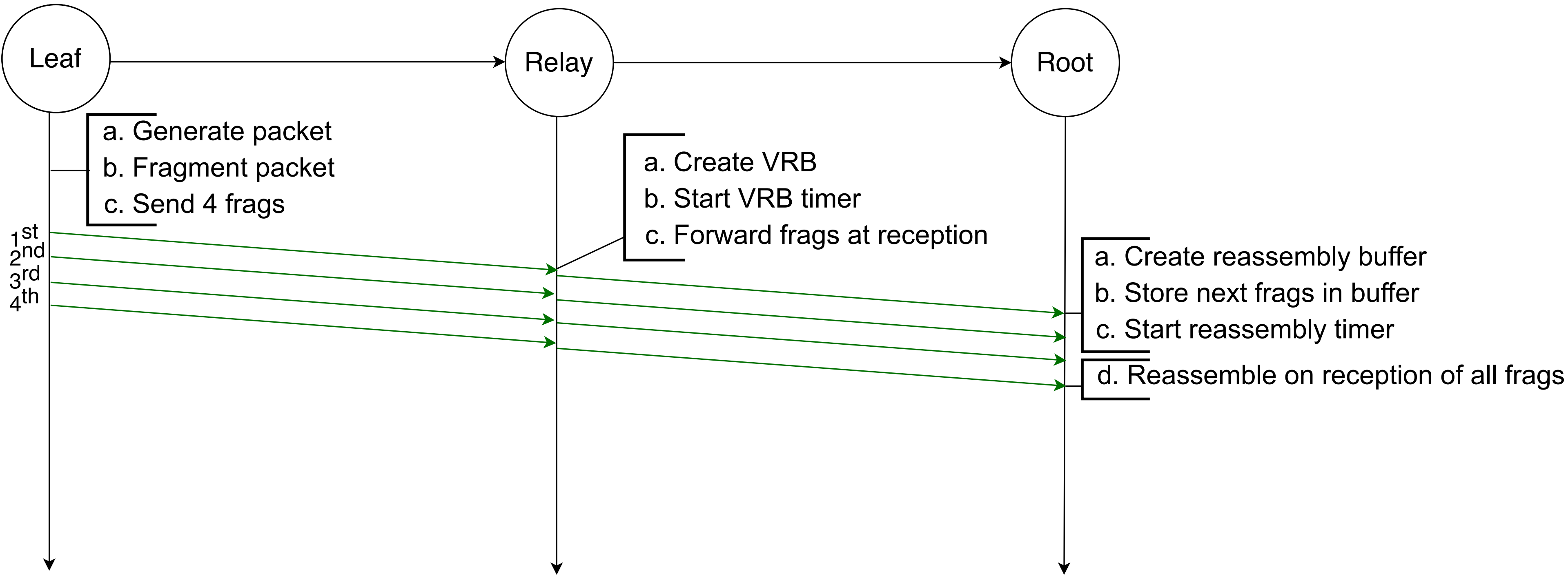


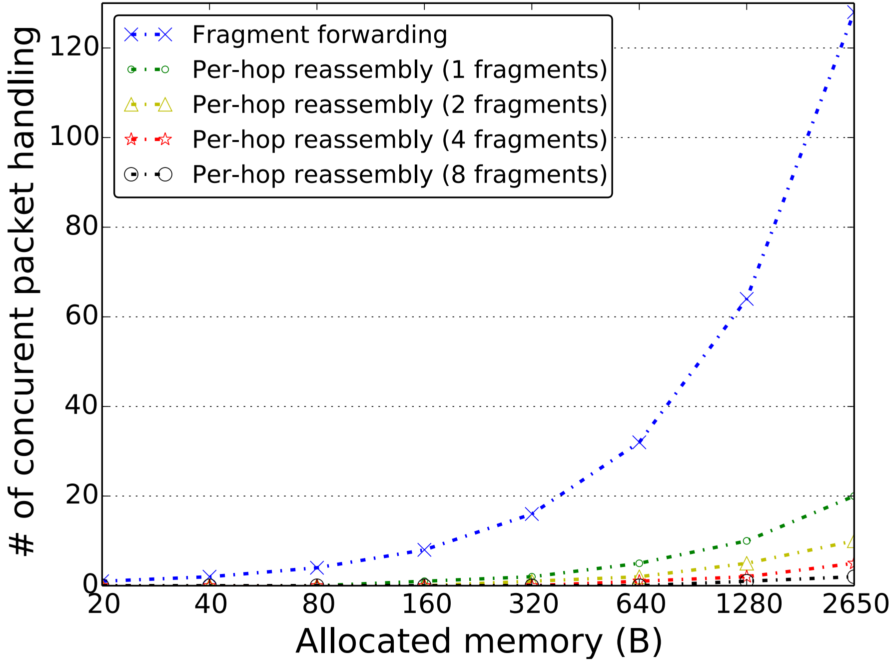
incoming		outgoing	
L2 src	tag	L2 dest	tag
B	7	H	17
D	3	H	8
F	3	H	11
empty			

Node G's VRB table

A typical operation of the VRB scheme. Node G receives fragments from B, D and F that originally are transmitted by nodes A, C and E, with datagram_tag configured to 7, 3 and 3, respectively. To solve the datagram_tag issue, when G receives two fragments with the same datagram_tag (i.e., datagram_tag 3 from nodes D and F), at the outgoing columns, it picks a new datagram_tag that is unique for the next hop node.

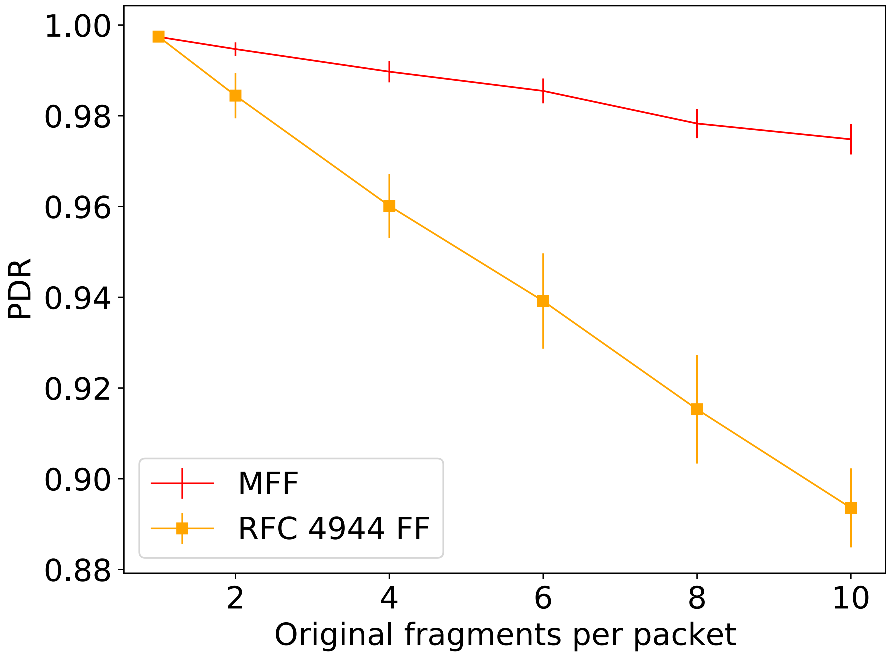
6LFF: Summary (4/4)

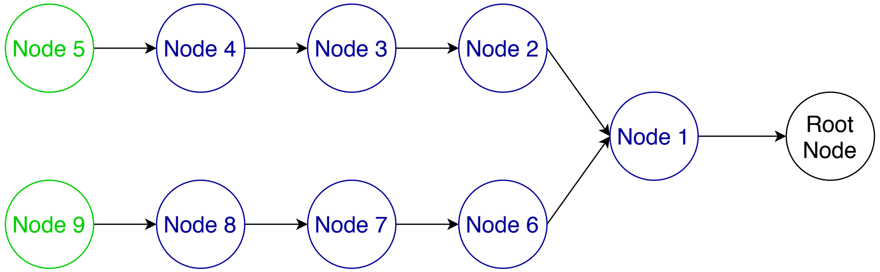




Number of concurrent data packet management at a potential relay node. When comparing the 6LFF (VRB-based) scheme against Per-hop Reassembly (Route-Over) with various fragment sizes.

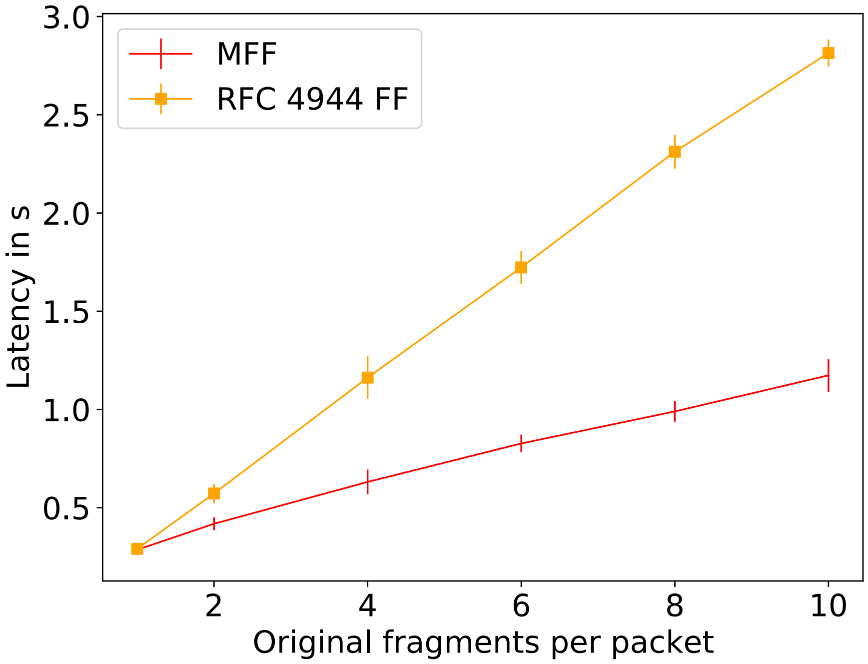
Performance Evaluation: Network Reliability

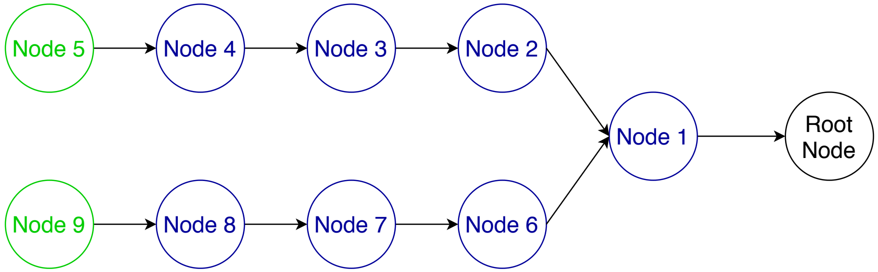




Comparing end-to-end reliability with per-hop reassembly and fragment forwarding. Results are averaged over 100 simulation runs and plotted with a 95% confidence interval. Leaf nodes 5 and 9 are transmitting IPv6 data packets to the root node every 40 seconds with a link quality at every hop of 0.85%. With per-hop reassembly, frames are dropped at node I because it runs out of memory for the reassembly buffer.

Performance Evaluation: Network Latency





Comparing end-to-end latency with per-hop reassembly and fragment forwarding. Results are averaged over 100 simulation runs and plotted with a 95% confidence interval. Leaf nodes 5 and 9 are transmitting IPv6 data packets to the root node every 40 seconds with a link quality at every hop of 0.85%. In this scenario, using fragment forwarding reduces end-to-end latency by roughly 50% when compared to per-hop reassembly.

Institut Mines-Télécom

6LoWPAN: Fragmentation & Reassembly, Frame Delivery Modes, & Fragment Forwarding

Georgios Z. PAPADOPOULOS, Professor, IMT Atlantique
georgios.papadopoulos@imt-atlantique.fr
www.georgiospapadopoulos.com
www.youtube.com/c/gzpapadopoulos