Dimensioning of Reassembly Buffers at the OLT

Glen Kramer, Broadcom

Introduction

- Reassembly buffers are not directly specified in 802.3ca.
- In 802.3ca spec, reassembly is assumed to happen automatically when the MAC receives the final fragment of a frame.
- However, real implementations have to deal with reassembly buffers and 802.3ca may need to provide special hooks to allow the OLT to manage the memory better.

Problem Statement

- 100G-EPON supports up to 56K ULIDs and 4K PLIDs (see kramer_3ca_1b_0916.pdf)
- 100G-EPON needs to support jumbo frames (10KBytes)
- Static memory allocation is not feasible
  - 60K x 10KB = 600 MBytes

- **How to support a large number of ULIDs/PLIDs while keeping memory requirements reasonable?**
- **What is the reasonable reassembly buffer size?**
  - 2MB, 4MB?
  - Examples in this presentation will assume 4MB reassembly buffer per PON port

Method #1 – Static Flow Limit

- Limit the number of fragmentable flows
- Add a field to the ULID assignment attribute to indicate whether that ULID is allowed to send fragments or not.

- Each fragmentable ULID gets a 10K reassembly slot permanently assigned to it.
  - With 4MB buffer and 10KB max frame size, we can have 400 fragmentable ULIDs per PON.

- 400 ULIDs out of 56K is just 0.7%.
- Unfragmentable ULIDs constitute the majority and will cause transmission inefficiency because OLT does not know the frame boundaries

- Hooks required from 802.3ca: a flag (boolean field) in the ULID assignment attribute

Method #1a – Static Flow+Frame Limit

- Not all flows will ever carry jumbo frames.
- Method #1a is like method #1, but the ULID assignment attribute, in addition to fragmentability flag, also carries the maximum allowed frame size on this ULID.
- All UNI ingress frames exceeding this size are dropped.

- Each fragmentable ULID gets either a 10K or 2K reassembly slot permanently assigned to it.
  - If among all the fragmentable ULIDs, only half should carry jumbo frames, then with 4MB buffer, we can have 668 fragmentable ULIDs
    - 333 ULIDs with max frame size of 10KB.
    - 335 ULIDs with max frame size of 2KB.

- 668 ULIDs out of 56K is just 1.16% - not a significant improvement.

- Hooks required from 802.3ca: two additional fields (boolean + integer) in the ULID assignment attribute

Method #2 – Dynamic Slot Reservation

- While the maximum possible number of ULIDs per standard is 56K, deployed systems generally don't need to configure/use that many.
- Also, not all configured ULIDs are active all at once.
- It may be possible to allow fragmentation on any ULID and still not over-run the buffer (most of the time).
  - Need a mechanism to handle a statistically-rare situation where more ULIDs have fragmented traffic than the buffer can handle.
- With a 4MB reassembly buffer, the OLT can handle ~400 fragments at any time.

Method #2 – Operation

OLT behavior

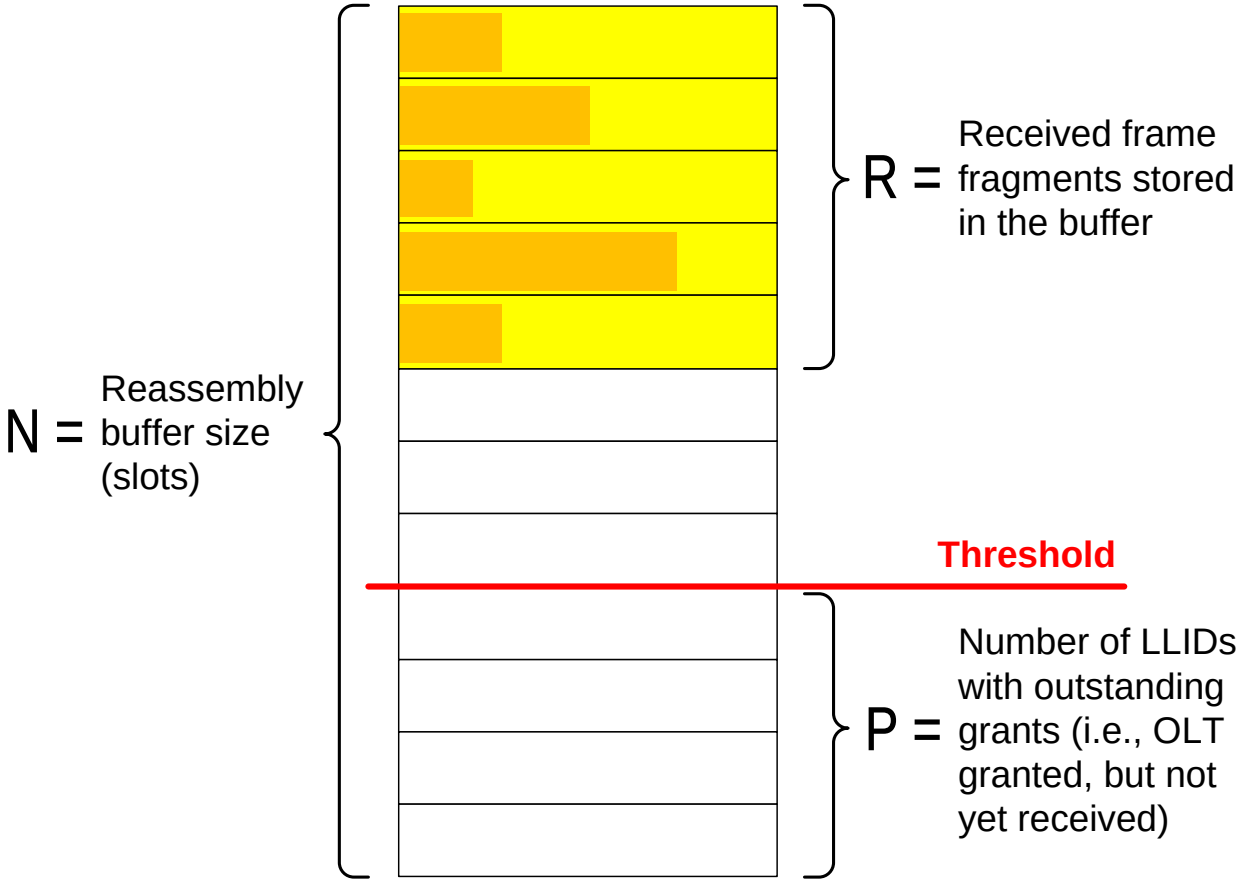
- When a frame header is received, the OLT allocates 10KB slot in the reassembly buffer for the given ULID.
- When the tail fragment is received, the complete frame is moved out and the 10K slot is freed for use by this or another ULID.
- When the buffer occupancy exceeds a certain threshold, the OLT sends a flag to the ONU to disallow fragmentation
- When the buffer occupancy decreases, the OLT sends a command to restart fragmentation
  - Use hysteresis (separate on/off thresholds) to avoid oscillations

ONU behavior

- If a grant to a ULID is received with the fragmentation disabled, the ONU will send the remaining fragment on that ULID, but will not fragment later frames.

- Decisions to make about the new ***fragment*** / ***do_not_fragment*** flag:
  - Delivered as part of the GATE message or a separate message?
  - Is per ULID, or per single ONU, or broadcast to all ONUs?
- Hooks required from 802.3ca: a new MAC Ctrl message or new field(s) in GATE MPCPDU

Method #2 operation

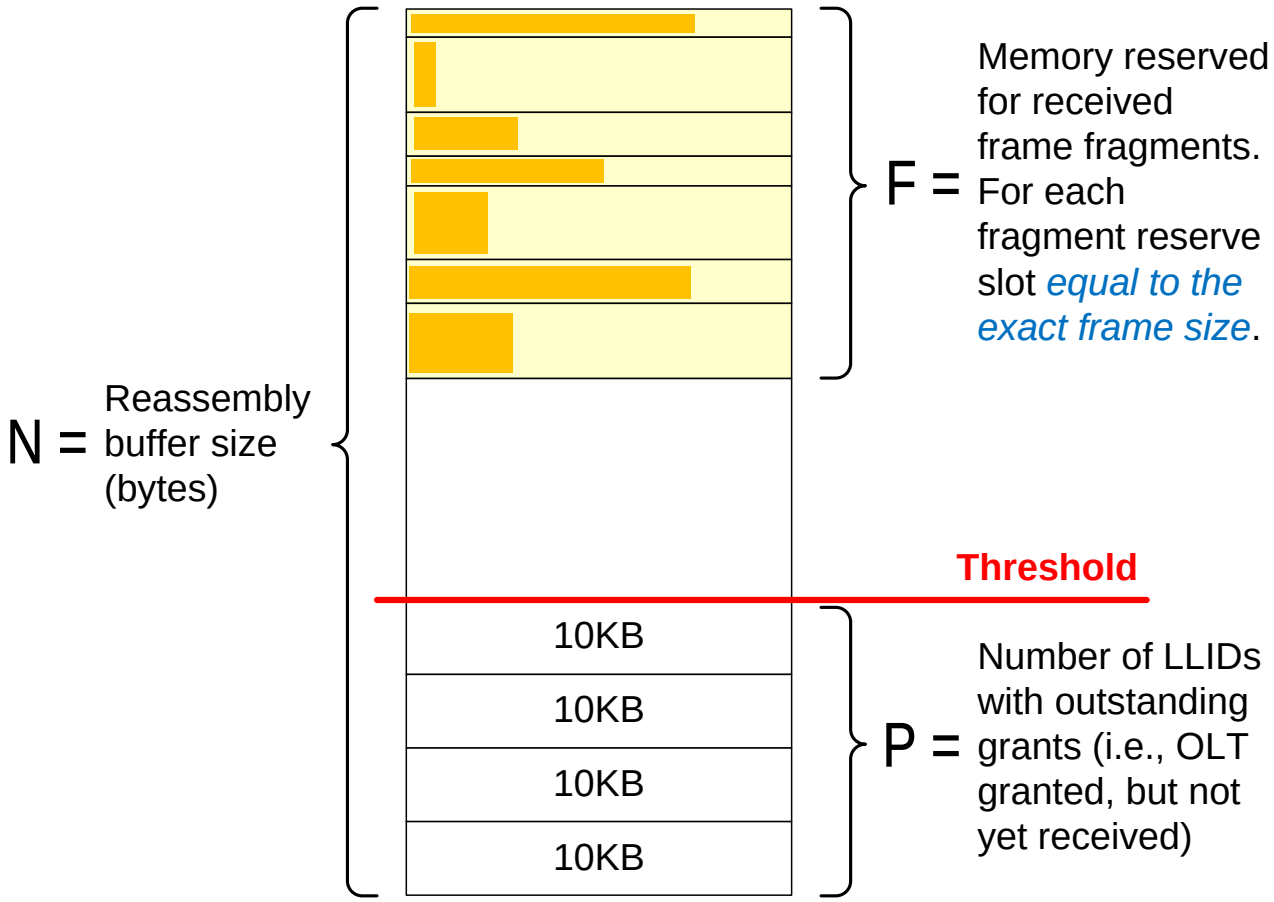



```
If( R+P < N )
    Grant with fragmentation_flag = enabled
else
    Grant with fragmentation_flag = disabled
```

Method #2a – Dynamic Allocation

- Even if an ULID can carry jumbo frames, a particular frame fragment may belong to a small frame. In such case, it is wasteful to allocate full 10KB slot in the reassembly buffer.
- Method #2a is an extension of Method #2:
  - The ONU tells the OLT how big is the full frame that is being received. The OLT allocates appropriately-sized reassembly buffer slot.
  - The OLT tells the ONU whether to fragment or not the last frame in each envelope.
- In this method, the envelope header (or frame's preamble) carries the information about the actual length of the full frame.
- When the OLT receives the beginning fragment of the frame, it will allocate the space in the reassembly buffer to fit the entire frame, but not more.

Method #2a operation




```
If( F+Px10KB < N )
    Grant with fragmentation_flag = enabled
else
    Grant with fragmentation_flag = disabled
```

Method #2a – Dynamic Allocation

- Hooks required from 802.3ca:
  - a new field in envelope header or in frame preamble to carry the frame length
  - a new downstream MAC Ctrl message or new field(s) in GATE MPCPDU

Issues

- Envelope headers (or preambles) are written in RS, below MAC, but MAC frame length is only available in MPCP.
  How to pass this info to RS? A new MPRS_CTRL.request primitive?
- Memory fragmentation issues – Reassembly buffer may be left with many small free slots instead of one contiguous empty space.
  - Can be solved if Reassembly buffer is build as a linked list.

Method #3 – Static Frame / Dynamic Slot

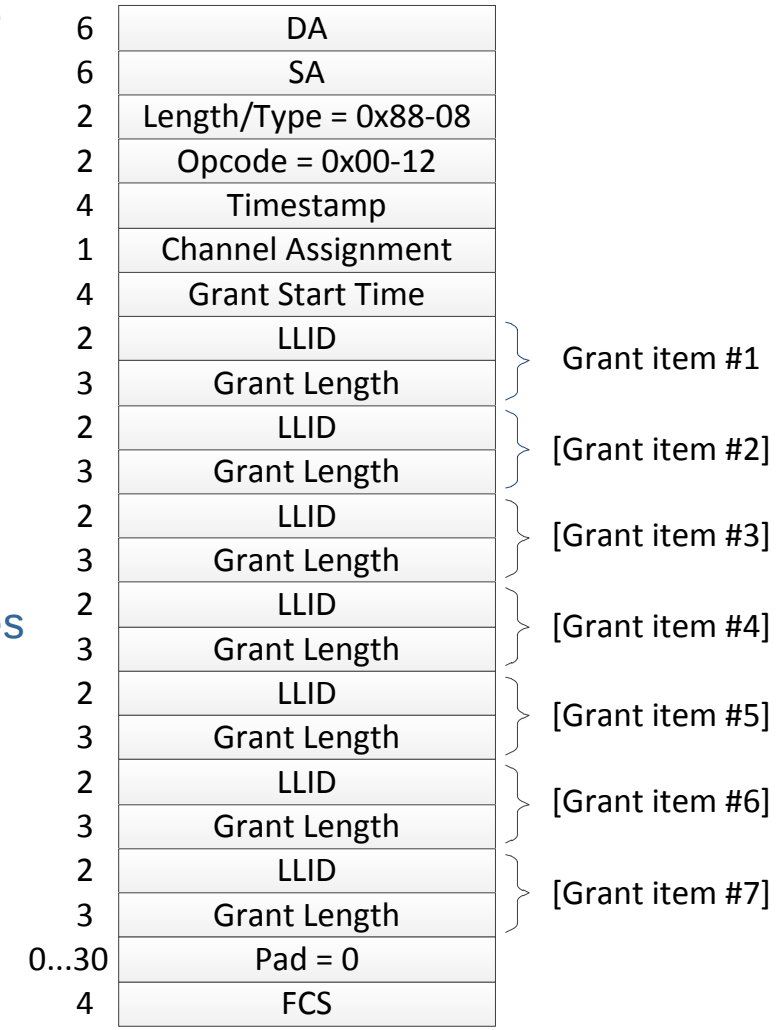
- Method #3 = Static Frame Limit + Dynamic Slot Reservation

- Upon registration, each ULID is provisioned the maximum allowed frame size as in Method #1a.
- OLT tells the ONU whether to fragment or not the last frame in each envelope as in Method #2

- Hooks required from 802.3ca:
  - two additional fields (boolean + integer) in the ULID assignment attribute
  - a new MAC Ctrl message or new field(s) in GATE MPCPDU

Method #4 – Token Granting

- A completely new twist on granting:
  - Grant Length represents not the envelope length, but simply a number of tokens given to an LLID.
  - The ONU adds this number to a token bucket for this LLID.
  - ONU serves (mostly) whole frames and decrements the corresponding token bucket.
  - Token bucket can accrue deficit (go negative) after serving a frame, but negative buckets don't get to serve frames until they receive enough tokens to become positive.
- Hooks required from 802.3ca: none, but a completely different behavior of MAC control client at the ONU (not 802.3ca spec).



Method #4 (continued)

- ONU shall comply with the overall burst length, so a given burst may have one or few fragments, but not a fragment per every LLID.
- The maximum number of outstanding fragments per ONU can be predefined, or configured via a management message.
  - If the OLT has the reassembly buffer big enough to hold 256 full frames, and it has registered 32 ONUs, it would allow 8 fragments per ONU.
  - If it registered 64 ONUs, it would allow 4 fragments per ONU.

- A corner case where this scheme breaks:
  1. OLT grants an LLID as a separate burst.
  2. Burst size is smaller than head-of-line frame.
  3. ONU currently has the maximum number of fragments outstanding.

  So
  - ONU cannot send a full frame from the granted LLID without violating the burst size.
  - ONU cannot fragment given frame without exceeding the number of outstanding fragments.
  - ONU can serve a different LLID in this burst, but that makes PON operation too unpredictable.
- In general, the OLT cannot completely control the QoS, since the amount and order of transmission for each ULID is determined by the ONU.
  - The OLT may be granting an ULID to ensure latency bounds, but the ONU serves a different ULID because of the fragmentation limits.

Summary

- Method #1 – Static Flow Limit
- Method #1a – Static Flow+Frame Limit
- Method #2 – Dynamic Slot Reservation
- Method #2a – Dynamic Allocation
- Method #3 – Static Frame/Dynamic Slot
- Method #4 – Token Granting

- Any other ideas?

- Straw Polls

Thank You