Dimensioning of Reassembly Buffers at the OLT

Glen Kramer, Broadcom

Recap of the previous discussion

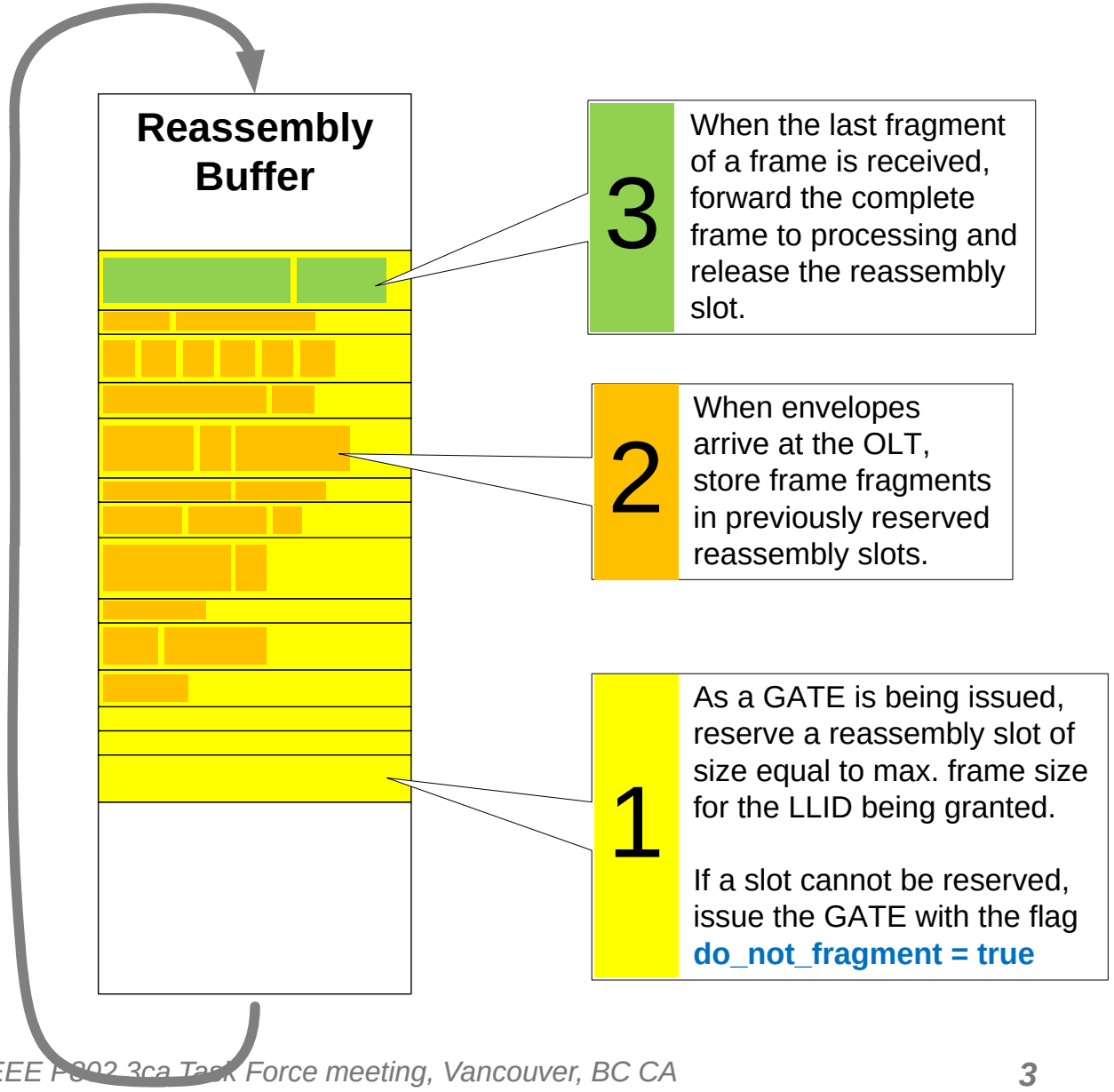
- For problem statement, see kramer_3ca_1a_0117.pdf
- In Huntington Beach in January, we discussed six methods of reassembly buffer management:
  - Method #1 – Static Slot Allocation
  - Method #1a – Static Slot Allocation + Static Frame Limit
  - Method #2 – Dynamic Slot Reservation
  - Method #2a – Dynamic Slot Allocation
  - Method #3 – Static Frame Limit + Dynamic Slot Allocation
  - Method #4 – Token Granting
- Suggestion from the floor: Focus more on Method #3
  - For each grant in flight, reserve a slot of size equal to the max. frame size allowed on a given LLID
  - When envelope is received, send every complete frame to packet processing and leave the remaining fragment in the reassembly buffer.
  - If the received envelope did not end with a frame fragment (i.e., the reassembly slot is empty), release the reassembly slot for use by other LLIDs.

Method #3 – High Level Steps

Step 1: Reservation

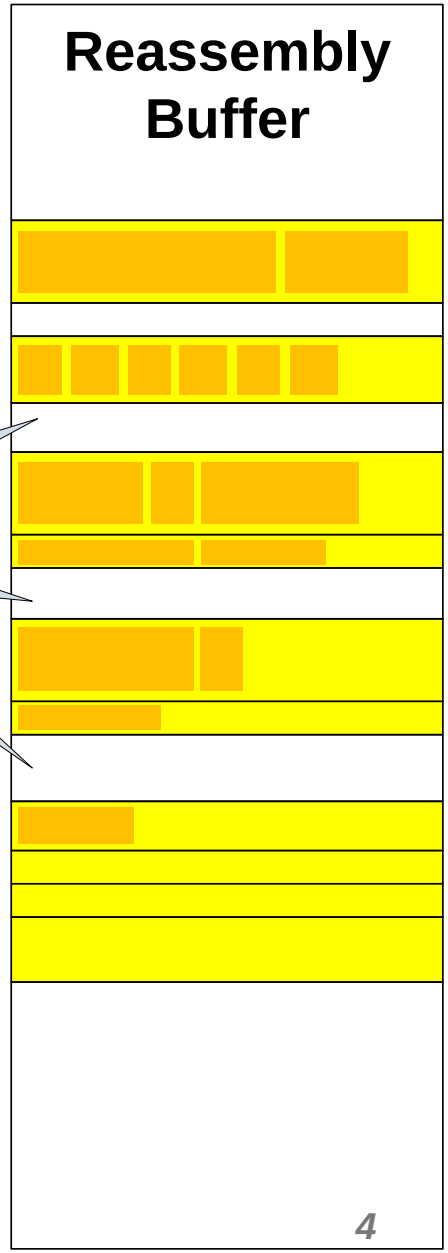
Step 2: Reassembly

Step 3: Release



Memory Fragmentation

- Typically, shorter frames will be split into fewer fragments than longer frames.
- Slots for LLIDs with smaller max. frame size often will be released sooner than the slots for LLIDs with large max. frame size, i.e., larger reassembly slots will generally be occupied longer than smaller slots.
- This will lead to many free small slots surrounded by busy slots.
- When the OLT needs to allocate the next large reassembly slot, it may not be able to do so, because the buffer's empty space consists of many disjoint small empty slots.



Mitigating Memory Fragmentation

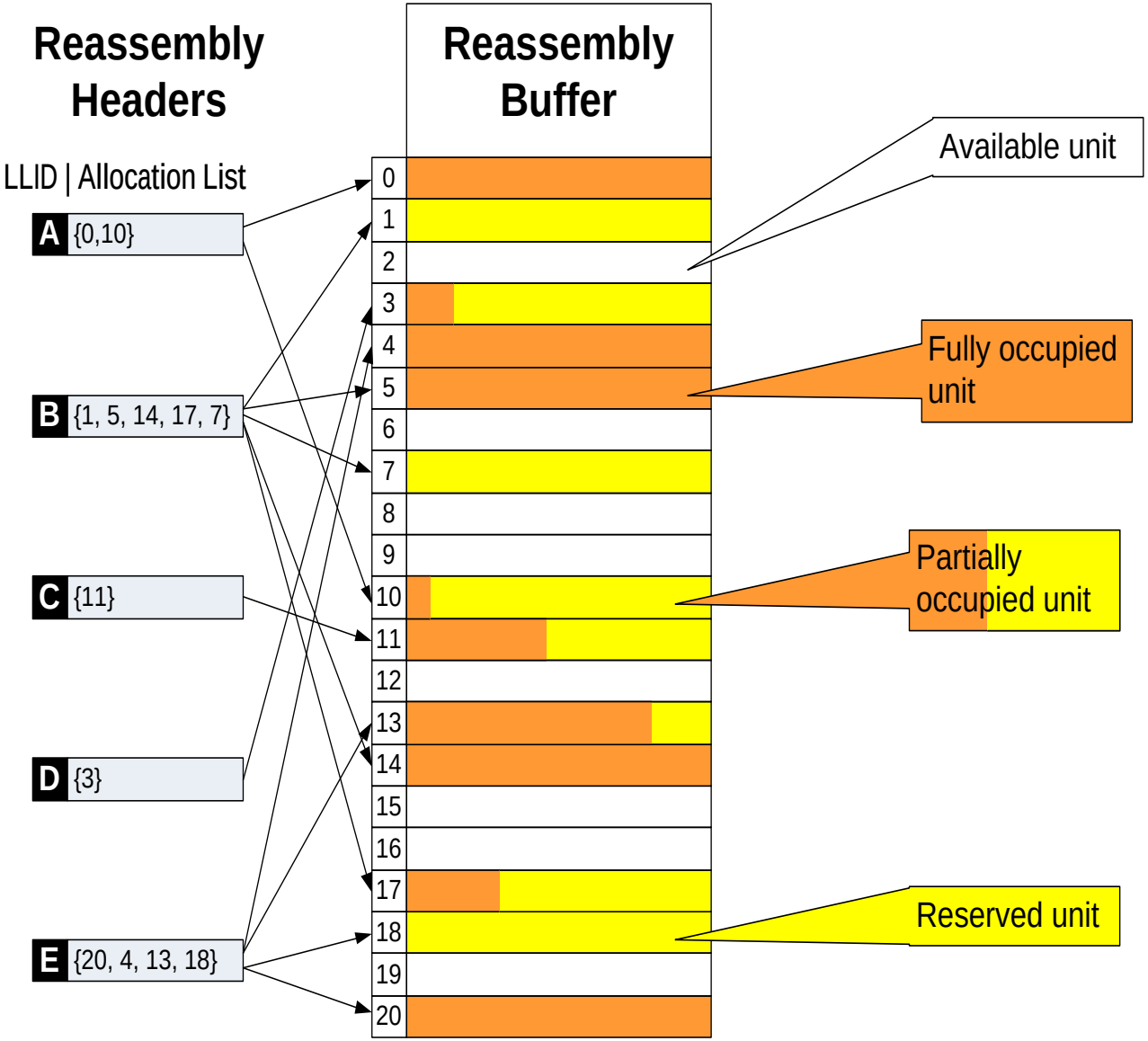
- Define fixed allocation unit (*U* EQs)
- Smallest reassembly slot will get just one allocation unit
  - Smallest slot = *U* EQs
- Large reassembly slot will get *N* allocation units
  - Large slot = *N*×*U* EQs

- Allocations slots for each LLID do not need to be contiguous.

- In addition to reassembly buffer, the OLT will have to store reassembly header per LLID containing the following information:
  - LLID value
  - List of allocated units
  - Write Pointer (points to first available EQ where the next received data to be written)

Allocation Units

- A new LLID would reserve a number of units sufficient to hold the maximum size frame permitted for this LLID.
- The allocation units need not be contiguous



Allocation Unit Size

- If provisioned max. frame sizes per LLID include only 2KB and 10KB frames, then unit size may be 251 EQs (for 2000 bytes of data + 8 bytes for preamble). The LLID with 10K frames will get allocated 5 units for a total of 1040 bytes.

- If provisioned max. frame sizes for different LLIDs include many different values, selecting the optimal allocation unit size may be more complicated. This is an optimization problem:
  - A smaller allocation unit size will improve buffer memory utilization, as overhead per LLID would be **< U**
    (i.e., $overhead = U - frm\_size \; \boldsymbol{MOD} \; U$**)**
  - A smaller allocation unit size will require a longer allocation unit list in the LLID header, i.e., it will increase header size.
    The number of units in the list is $\lceil frm\_size/U \rceil$

- Reassembly slot allocation unit size is an implementation choice
  - Not visible outside OLT
  - Does not affect interoperability

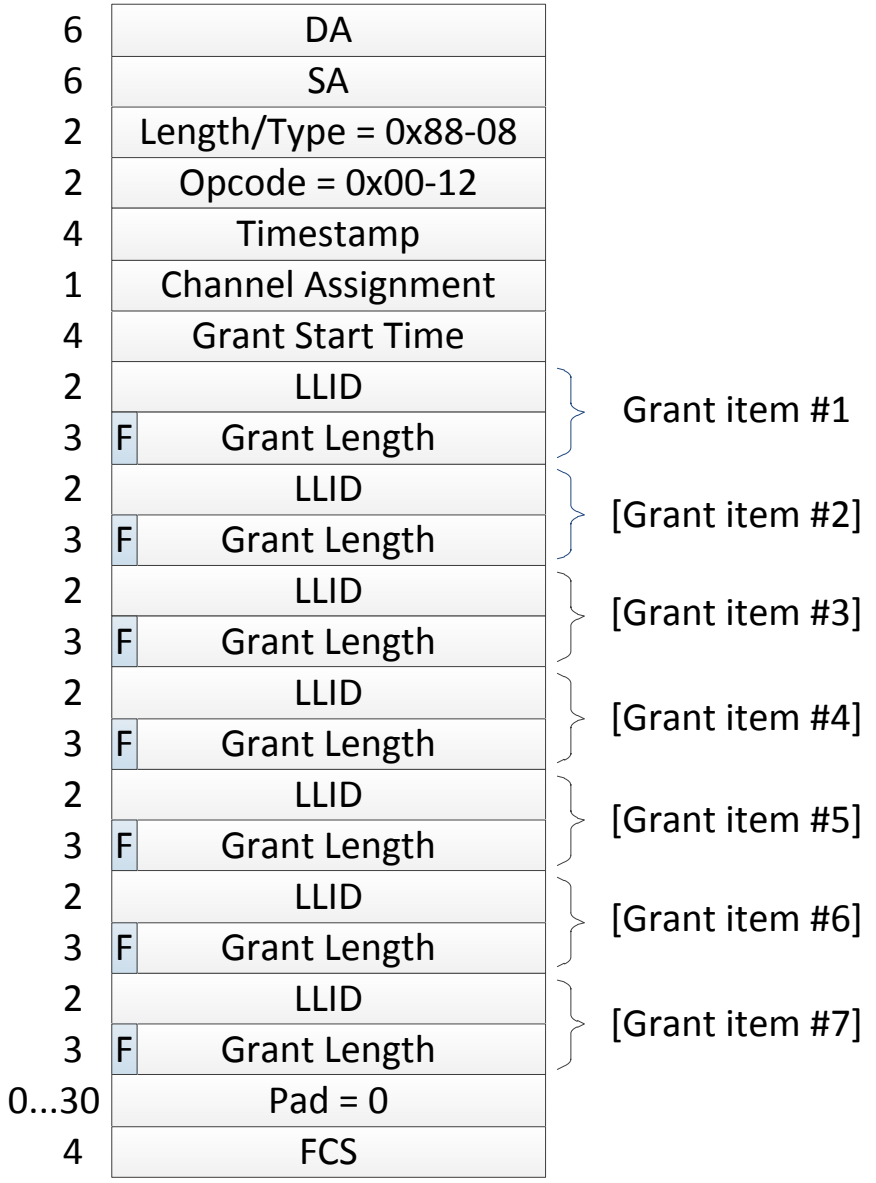
Updated GATE format

- What is the maximum grant size we need to support?

	TQ/EQ	MB	ms
1G-EPON	2^{16}	0.13	1
10G-EPON	2^{16}	1.31	1
25G-EPON	2^{24}	134	43

- Is it necessary to allow such a large grant, compared to 1G and 10G EPON? Will it ever be used?

- **Proposal:**
  Take ways 1 bit to carry ***fragment/do_not_fragment*** flag per LLID



Fragmentation Mode Locking

- Care should be taken to avoid Fragmentation Mode Locking, where the same set of LLIDs continuously find Reassembly Buffer vacancy counter too low and thus unable to fragment, while another set of LLIDs continuously retain reassembly slots by leaving fragments in the buffer, thus continuously being able to fragment.

Thank You