Channel Bonding Design for 100 Gb/s PON Based on FEC Codeword Alignment

Liang Zhang[1, 3], Yuanqiu Luo[1], Bo Gao[2], Xiang Liu[1], Frank effenberger[1], and Nirwan Ansari[3]

(1) Futurewei (Huawei) Technologies, 400 Crossing Blvd, Bridgewater, NJ, 08807, USA

(2) Huawei Technologies Wuhan Research Center, Optical valley software park, Wuhan, Hubei, China

(3) Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ, 07102, USA

Email: lz284@njit.edu; yuanqiu.luo@huawei.com

Motivation of this work

Channel bonding is needed for 100G PON:

- **High capacity and More flexible for data transmission**
- **Save capital expenditure (CAPEX) by using low data rate transmitters and receivers**

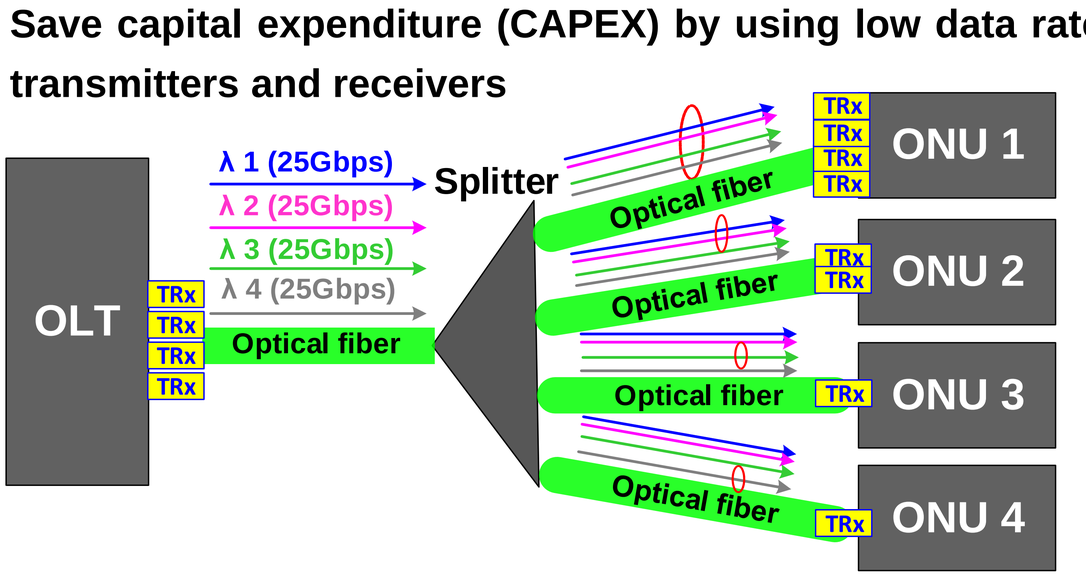

Fig. 1: An example of channel bonding for 100G PON system.

Where to add Channel Bonding

- **Add channel bonding function between MPCP and MAC layer to minimize changes to the 10G EPON.**

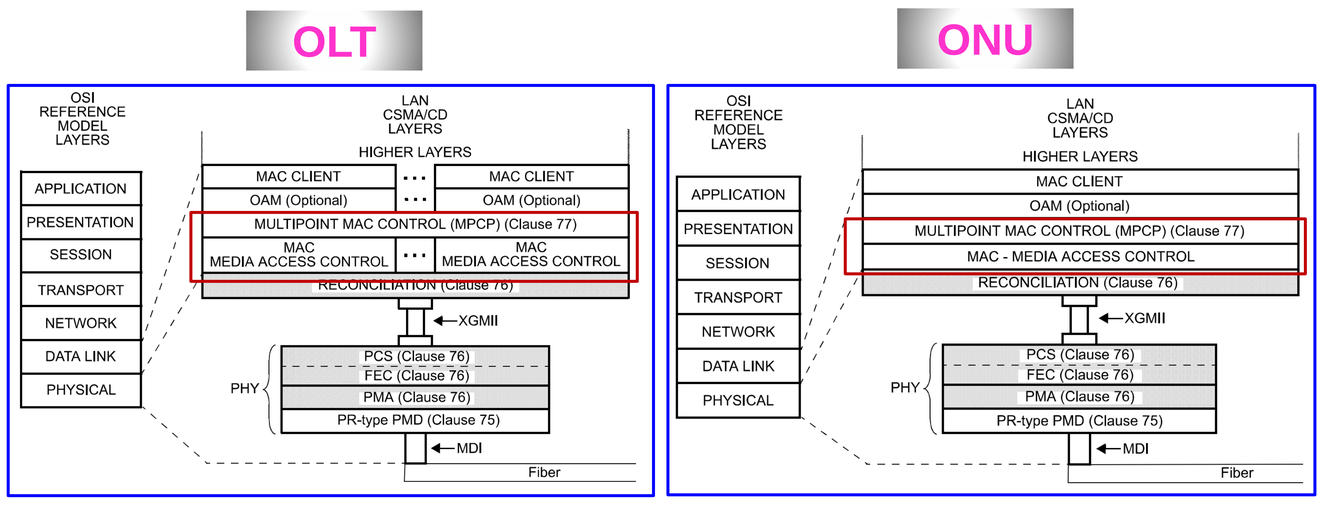

Fig. 2: Relationship of 10/10G-EPON P2MP RS, PCS, and PMA to the ISO/IEC OSI reference model and the IEEE 802.3 CSMA/CD LAN model

***From**: IEEE Standard for Ethernet 802.3, SECTION FIVE, 2012.*

System Structure of Channel Bonding

- **Channel bonding system structure by reusing IEEE 802.3 PON MAC**
- **Different ONU has different number of transceivers**
- **Each TRx uses one pair wavelength for transmission**
- **Channel bonding occurs when forming/parsing FEC codewords**

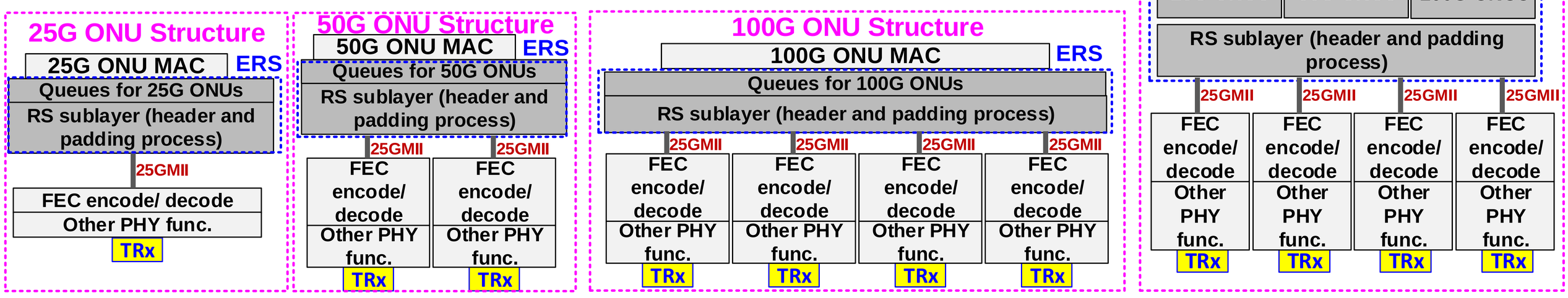

Fig. 3: System structure of channel bonding at FEC CW level.

Algorithms for Channel Bonding

We propose two heuristic algorithms for Channel Bonding:

- **ERS Queues are served by the scheduler for the ONUs**
- **Data are padded to an integer amount of FEC CWs and transmitted immediately (IP algorithm)**
- **A threshold κ for each queue is pre-set for the HP algorithm**
- **Data are transmitted when they are no less than an FEC CW, or Data (≤ 1 CW) are transmitted with padding after κ visit times (HP algorithm)**

Algorithm 1: Immediately Padding (*IP*)

Input : $\mathscr{P}_h$, $\mathscr{Q}_{h,k}$, $\mathscr{R}_{h,k}$, $\mathscr{M}$ and $\mathscr{N}$;
Output: $x_{h,k}^{i,j}$;

1 **while** $\sum_{k,i,j} x_{h,k}^{i,j} + |\mathscr{Z}_h| < |\mathscr{N}|$ **do**
2 **for** *channel* $j \in \mathscr{N}$ **do**
3 check status of corresponding queue $\mathscr{Q}_{h,k}$;
4 **if** $\mathscr{R}_{h,k} \neq \varnothing$ **then**
5 set $x_{h,k}^{i,j} = 1$, and remove one CW in $\mathscr{R}_{h,k}$;
6 **else if** $\mathscr{R}_{h,k} = \varnothing$ & $\mathscr{Q}_{h,k} \neq \varnothing$ **then**
7 add idle bytes to $\mathscr{Q}_{h,k}$ until $\mathscr{R}_{h,k} = 1$;
8 set $x_{h,k}^{i,j} = 1$, and remove one CW from $\mathscr{R}_{h,k}$;
9 check 100 Gb/s queues $\mathscr{Q}_{h,k}$, and repeat steps $4 \sim 8$;
10 **if** *no CW ready* **then**
11 add one padding CW to $\mathscr{Z}_h$;
12 schedule this padding CW to channel j;
13 transmit assigned CWs to all $\mathscr{N}$ channels;

Algorithm 2: Hold and Pack (*HP*)

Input : $\mathscr{P}_h$, $\mathscr{Q}_{h,k}$, $\mathscr{R}_{h,k}$, $\mathscr{M}$, $\mathscr{N}$ and κ;
Output: $x_{h,k}^{i,j}$;

1 **while** $\sum_{k,i,j} x_{h,k}^{i,j} + |\mathscr{Z}_h| < |\mathscr{N}|$ **do**
2 set the reading time $t_k = 0$ for each queue;
3 **for** *channel* $j \in \mathscr{N}$ **do**
4 check status of corresponding queue $\mathscr{Q}_{h,k}$;
5 **if** $\mathscr{R}_{h,k} \neq \varnothing$ **then**
6 set $x_{h,k}^{i,j} = 1$, and remove one CW from $\mathscr{R}_{h,k}$;
7 **else if** $\mathscr{R}_{h,k} = \varnothing$ & $\mathscr{Q}_{h,k} \neq \varnothing$ **then**
8 **if** $t_k = \kappa$ **then**
9 set $x_{h,k}^{i,j} = 1$ and add idle bytes till $\mathscr{R}_{h,k} = 1$;
10 set $t_k = 0$ and remove one CW from $\mathscr{R}_{h,k}$;
11 **else**
 $t_k = t_k + 1$;
12 check 100 Gb/s queues $\mathscr{Q}_{h,k}$, and repeat steps $5 \sim 11$;
13 **if** *no CW ready* **then**
14 add one padding CW to $\mathscr{Z}_h$;
15 schedule this padding CW to channel j;
16 transmit assigned CWs to all $\mathscr{N}$ channels;

Performance Evaluation

- **Normal distribution is used to generate packets. μ is set as 64, λ is set as 3000. Packet gap is randomly generated from 12 bytes to 200 bytes.**

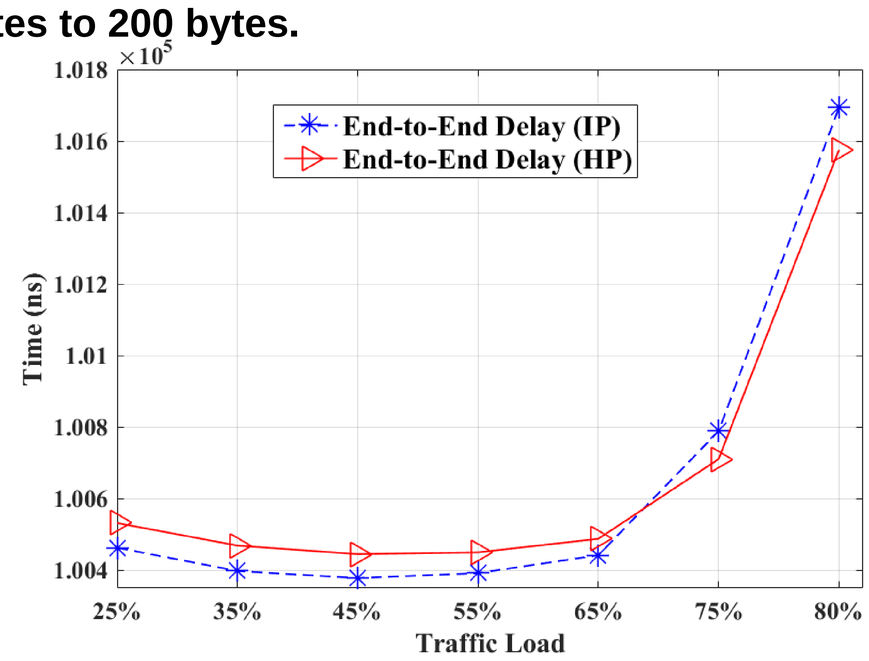

Fig. 4: End to end delay result.

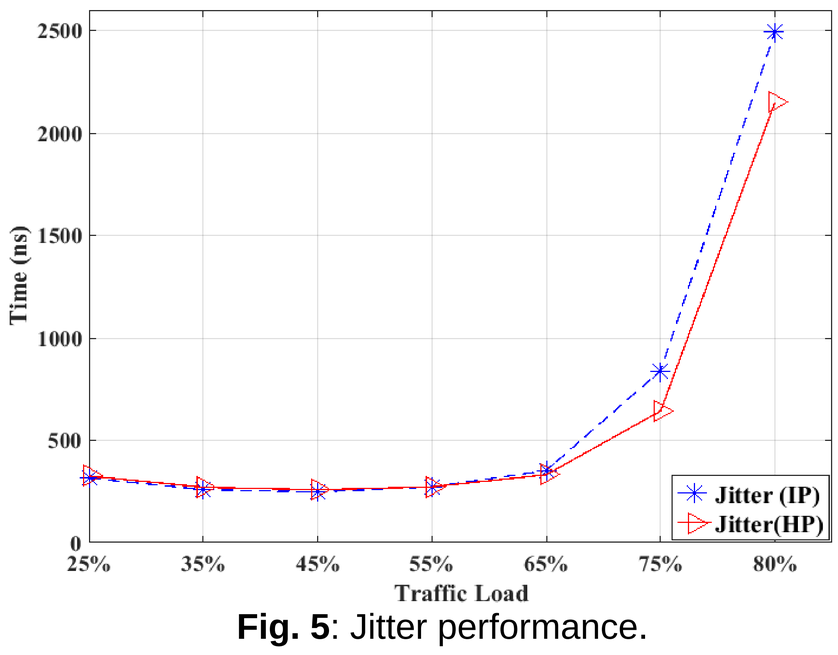

Fig. 5: Jitter performance.