YANG Data Model for FlexE Interface Management

draft-jiang-ccamp-flexe-yang-02

Y. Jiang, X. He, W. Cheng, J. Wang, Y. Han

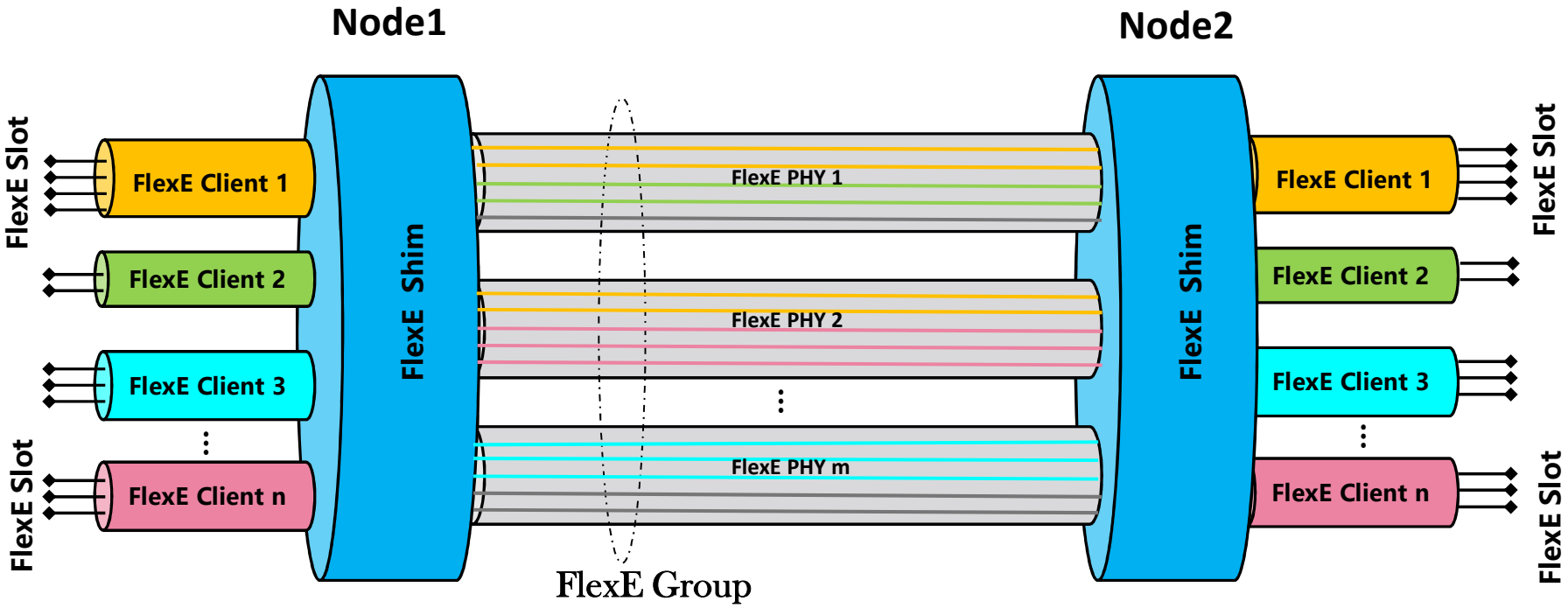
Presenter: Yuanlong Jiang

106th IETF – Singapore

Objective of FlexE Interface Mgmt

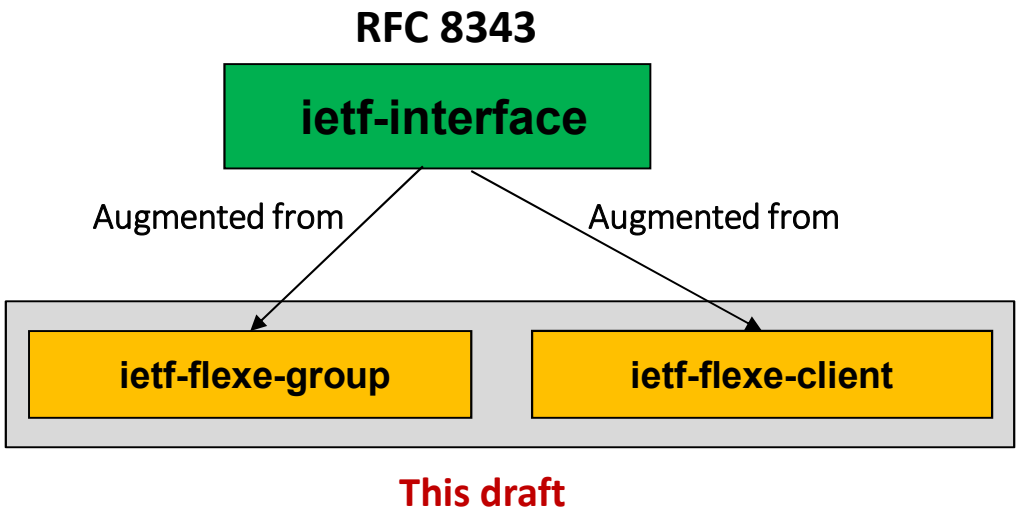
- Configuration of FlexE Group as a network interface, binding of multiple FlexE PHYs, configuration of FlexE Clients and their slots in a FlexE Group
- FlexE status retrieval including FlexE Group, FlexE PHYs and FlexE clients
- Support of interface management in SDN and NMS
- Support of FlexE 2.1 & 2.0, also backwards compatible with FlexE 1.1 &1.0

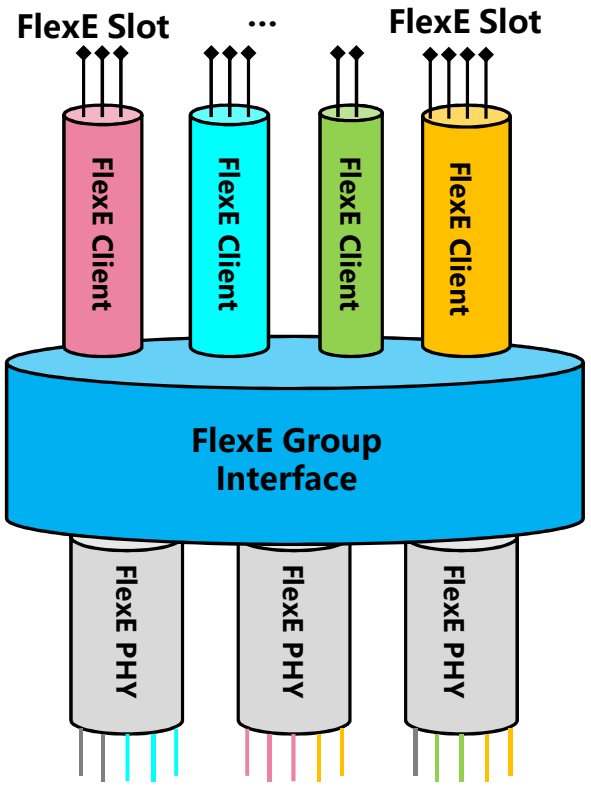
FlexE data plane in depth



FlexE slot is a basic construct unit for
FlexE client and PHY in a FlexE Group

Abstraction of FlexE management





- ✓ Both FlexE Group and FlexE Client are modeled as network interfaces
- ✓ FlexE Group is on the top layer, which contains multiple FlexE Clients and PHYs
- ✓ FlexE Clients can be added, deleted, configured, or resized with some FlexE slots

YANG Tree Diagrams (updates in Red)

```
module: ietf-flexe
 augment /if:interfaces/if:interface:
  +--rw flexe-group
    +--rw group-number?       uint32
    +--rw slot-granularity?   slot-granularity-enumeration
    +--rw flexe-phy-type?     flexe-phy-enumeration
    +--rw calendar-protocol-enable? Boolean
    +--rw flexe-phy-list* [flexe-phy-if]
    |  +--rw flexe-phy-if?       if:interface-ref
    |  +--rw phy-number          uint8
    |  +--ro phy-status?         uint8
    |  +--rw calendar-slot-list* [slot-id]
    |     +--rw slot-id                uint8
    |     +--rw flexe-slot-status?   slot-status-enumeration
    +--rw flexe-client-list* [client-id]
    |  +--rw client-id             uint16
    |  +--rw flexe-client-if?      if:interface-ref
    |  +--rw client-slot-list* [client-slot-id]
    |  |  +--rw client-slot-id          uint8
    |  |  +--rw mapped-phy-if?      if:interface-ref
    |  |  +--rw mapped-slot-id       uint8
    |  +--ro flexe-client-status?   uint8
    +--ro flexe-group-status?   uint8
```

support the FlexE calendar negotiation protocol

support mapping of slots to a PHY

```
module: ietf-interfaces-flexe-client
 augment /if:interfaces/if:interface:
  +--rw flexe-client
    +--ro mac-address
    +--rw group-number?         uint32
```

CCAMP Minutes from IETF 105

- People think CCAMP is the place to start the work on FlexE YANG
- Need to check the FlexE work done in other SDOs, such as ITU-T SG15
- Need to look into other IETF YANG models for similar work
- Needed more analysis on the pros and cons of the existing FlexE YANG I-Ds

FlexE progress in other SDOs

- In July, OIF published Flex Ethernet 2.1 Implementation Agreement (adds support of 50GBASE-R PHY)
- In September, during ITU-T Q14/SG15 interim meeting (Gothenburg), **regarding FlexE management information (MI) in G.8023, it was agreed**:
  - CCA &CCB are relevant only between the EMF and the atomic functions in a device;
  - CC, CR, & CA are controlled by EMF based on the client calendar configuration protocol or static configuration. They are not visible in the external interface
  - Tx and Ex are set to the same value
  - Some FlexE MI need not to be visible in the external management/control interface

Similar work on YANG modules

- **IANA defines a type of LAG interface, i.e., ieee8023adLag, see RFC 7224**
- **3 Predecessors of YANG interface with bonded Ethernet links:**
  - IETF: Ethernet Bonding Interface Module, see Appendix B of RFC8343
  - IEEE: module ieee802-dot1ax
  - OpenConfig: module openconfig-lacp
  - **They are not FlexE, but all of them are related to bonded Ethernet links, and modeled as a single interface**

Why augment FlexE from interface

Advantages:

- Inherit if:type, if:name and other attributes for conventional interface management
- Simplicity, just use ‘leafref’ to reference to a FlexE interface in applications
- Unified style of naming, addressing, accessing and exception processing procedures as any other interfaces
- Support of interface layering gracefully

Whether to model FlexE instance or not

- **Do we need to model FlexE instance?**
  - Instance number can be automatically inferred from a FlexE PHY number
    - For 50G and 100GBASE-R, Instance number=PHY number
    - For 200GBASE-R, Instance number=PHY number*2 + [0, 1]
    - For 400GBASE-R, Instance number=PHY number*4 + [0, 3]
  - Instance is internal, the only usage is to indicate unequipped instance in the case of 200 or 400G, but the later can also be calculated by slot usage given the unavailable slots are always located in the end of a PHY slot list (i.e., the highest numbered instance on a FlexE PHY gets unequipped firstly)
  - Furthermore, FlexE instance not exist in FlexE 1.1 or FlexE 1.0, thus will not be compatible if it is modelled
  - **Therefore, FlexE instance is suggested not to be modeled**

Whether to model CC, CR, and CA

- **Do we need to model CC, CR and CA?**
  - If calendar negotiation protocol is enabled: dynamic signaling of CC, CR and CA is done automatically in the data plane (in the FlexE Overhead), it is meaningless to configure or retrieve these ephemeral signaling states
  - If calendar negotiation protocol is disabled: CC, CR and CA are static
  - In both cases, NMS or SDN controller does not need to care
  - **Therefore, CC, CR and CA are suggested not to be modeled**

Whether to model both CCA & CCB

- **Do we need to model both CCA and CCB?**
  - CCA & CCB are used for hitless client reconfiguration, one for active and one for backup; it also means **writing to CCA & CCB at the same time will cause loss of traffic on all the FlexE clients**
  - If negotiation protocol is enabled: client configuration is always written to the backup calendar, and calendar switching to the backup can be hitless
  - If negotiation protocol is disabled: switching from the active to the backup is not needed, client configuration can be written to the active calendar directly
  - In both cases, SDN controller/NMS only needs to deal with a single calendar configuration, while the device knows whether CCA or CCB is actually used for all time
  - **Therefore, CCA & CCB are suggested not to be modeled**

Bidirectional symmetric vs. asymmetric

- **Do we need to configure both TX & RX parameters?**
  - FlexE IA only discusses the configuration of a unidirectional client
  - The signaling of CR and CA relies on a bidirectional overhead channel in the same FlexE Group
  - The FlexE links (including each of the bonding PHYs) are always bidirectional symmetric
  - FlexE clients are always reserved with the same number of slots (from which the bandwidth can be calculated) in both directions
  - Therefore, bidirectional symmetric parameters are more reasonable, **there is no need to model TX & RX parameters separately**

Where to model FlexE Slot list

- Assume FlexE Group binding of 4 PHYs, we have 3 options to model slot list (for the same slot):
  - Slot list within FlexE Group, Slot[k], where k is the index of the slot in the whole calendar
  - Slot list within FlexE PHY[m], Slot[m][i], where m is the index [0-3] of the PHY, and i is the index of the slot within the PHY
  - Slot list within FlexE instance[n], Slot[n][j], where n is the index of the instance, and j is the index of the slot within the instance
  - The relationship among them:
    - 50GBASE-R, k=10*n+j=10*m+i, where $k \in [0,39]$; n=m=floor(k/10); j=i=mod(k/10)
    - 100GBASE-R, k=20*n+j=20*m+i, where $k \in [0,79]$; n=m=floor(k/20); j=i=mod(k/20)
    - 200GBASE-R, k=20*n+j=40*m+i, where $k \in [0,159]$; n=floor(k/20), j=mod(k/20); m=floor(k/40), i=mod(k/40)
    - 400GBASE-R, k=20*n+j=80*m+i, where $k \in [0,319]$; n=floor(k/20), j=mod(k/20); m=floor(k/80), i=mod(k/80)
- Compared with other options, **model the slot list on each PHY is more compatible, and easier to find unused or unavailable slots in a particular PHY**

In summary-main differences

draft-jiang-ccamp-flexe-yang	draft-xiaobn-ccamp-flexe-yang-mod
FlexE Group modeled as a new interface, more conventional	FlexE-Groups is just a new container
FlexE Instance is not modeled	FlexE Instance is modeled, not backwards compatible
CC, CR and CA are not modeled	CC, CR and CA are modeled
A single calendar is modeled, independent of CCA/CCB	Both CCA & CCB can be configured, but actually it will defeat the purpose of hitless switching
Bidirectional symmetric, not TX & RX	Both TX & RX parameters are modeled
Slot list modeled on each FlexE PHY, unused slots are readily indicated in the calendar-slot-list	Used slots modeled on FlexE Instance when they are allocated to a client; unused slots can only be determined after traversing all flexe-clients
FlexE client is contained in FlexE group, it means FlexE client will only exist if its FlexE group exists	FlexE-clients is outside of FlexE-groups, thus FlexE client can be instantiated without a FlexE group, more error prone

In summary-other differences

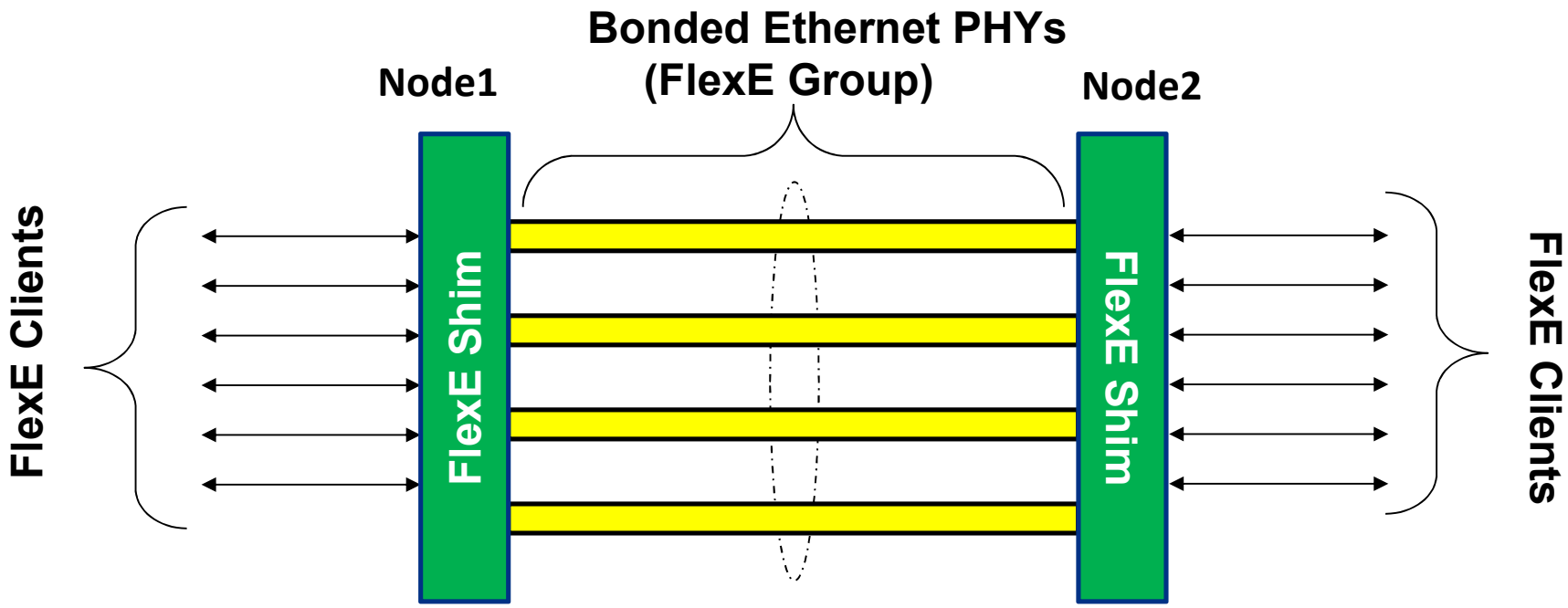
draft-jiang-ccamp-flexe-yang	draft-xiaobn-ccamp-flexe-yang-mod
4 tier YANG tree, flat & simple	7 tier YANG tree, deep & complex
Only slots are modelled, bandwidth can be calculated from slots*granularity	Both slots and bandwidth are modeled, but how to provide the bandwidth value consistently may pose a challenge
Only local parameters are included, no remote PHY needs to be configured	Both local and remote PHYs are included in the model, thus more complex
Support status retrieval of FlexE Group, PHYs and clients	No support of status retrieval
Client number is local to a FlexE Group, thus scalable in a large network (2^{36} clients in maximum together with 20 bits of FlexE Group)	As a single YANG key, client number must be globally unique independent of a FlexE Group, thus not scalable in a large network (only 2^{16} clients in maximum)

Next Step

- Update the draft according to WG feedbacks
- Call for WG adoption

Thank You

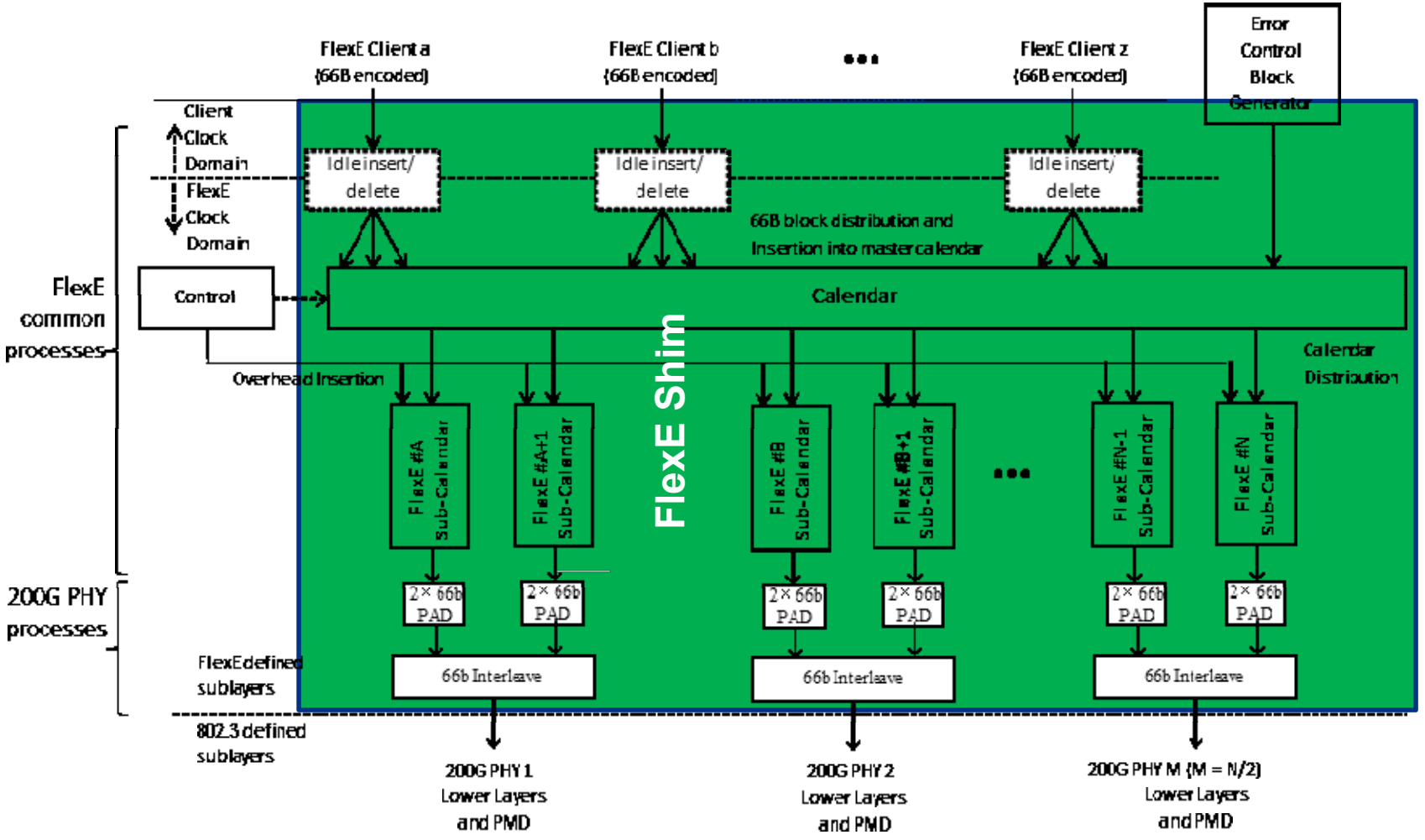
FlexE overview



FlexE support:

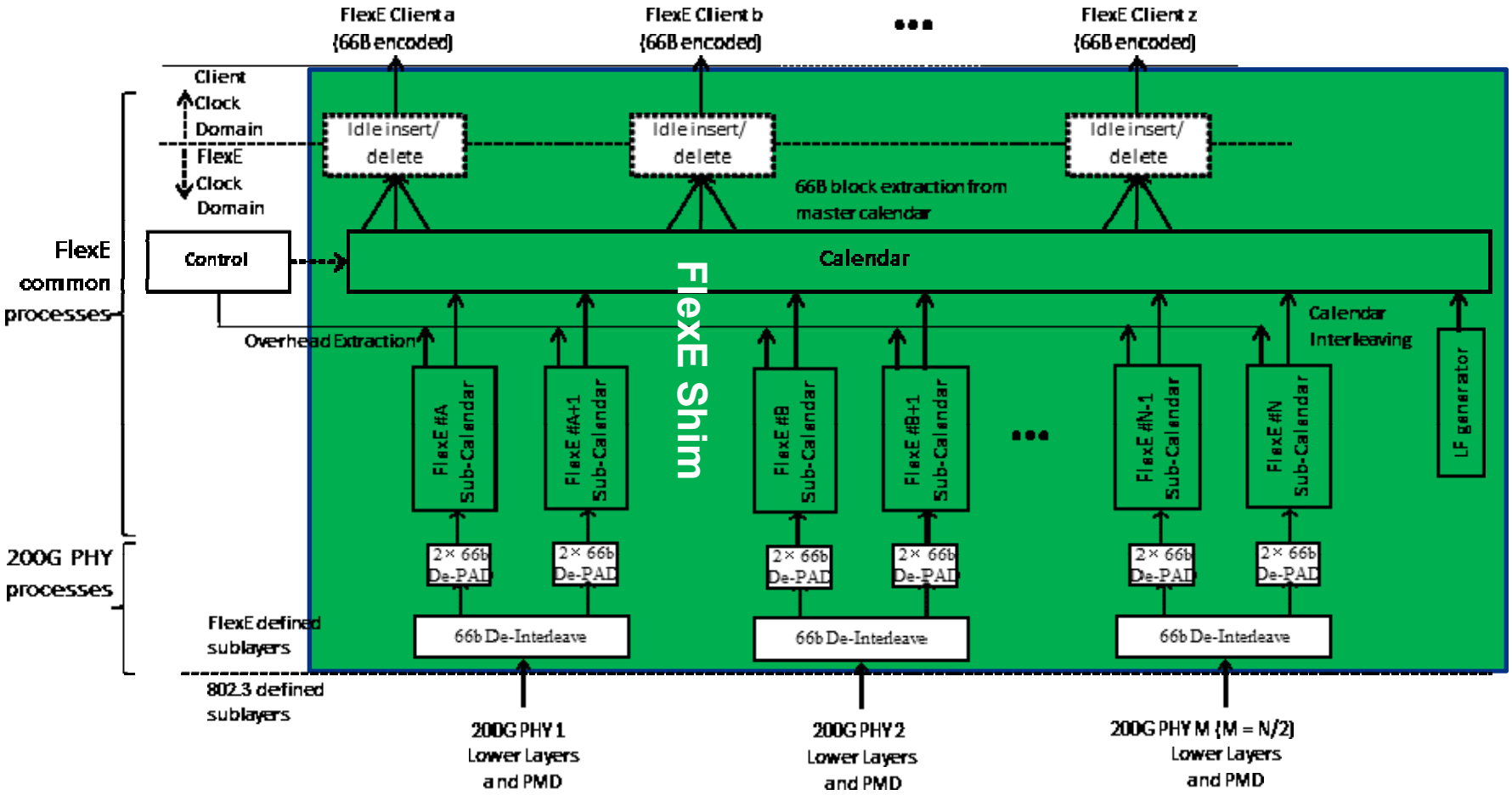
- **Bonding of ETH PHYs (1~n)**
- **Sub-rates of ETH PHY (minimum of 5G)**
- **Channelization within a PHY or a group of bonded PHYs (5G ~ m*5G)**

FlexE mux functions (200GBASE-R)



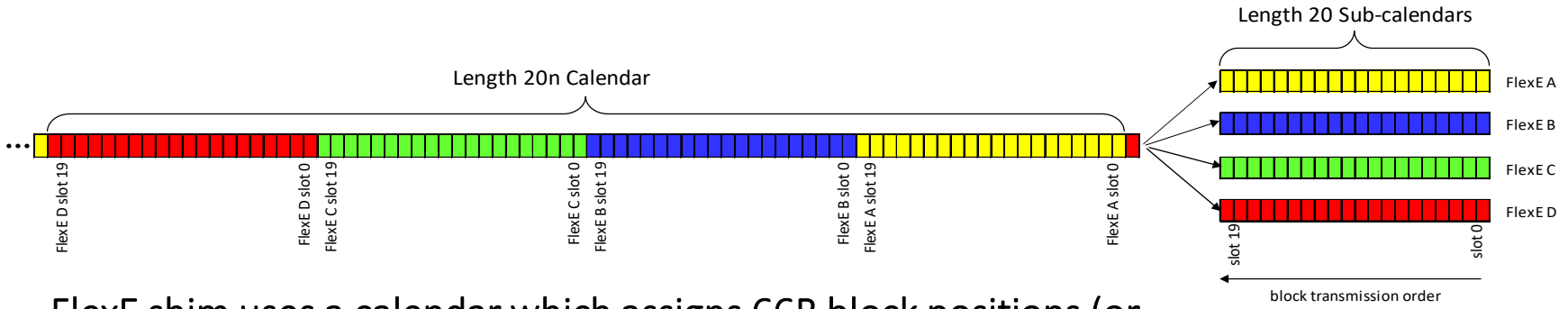
Node1

FlexE demux functions (200GBASE-R)



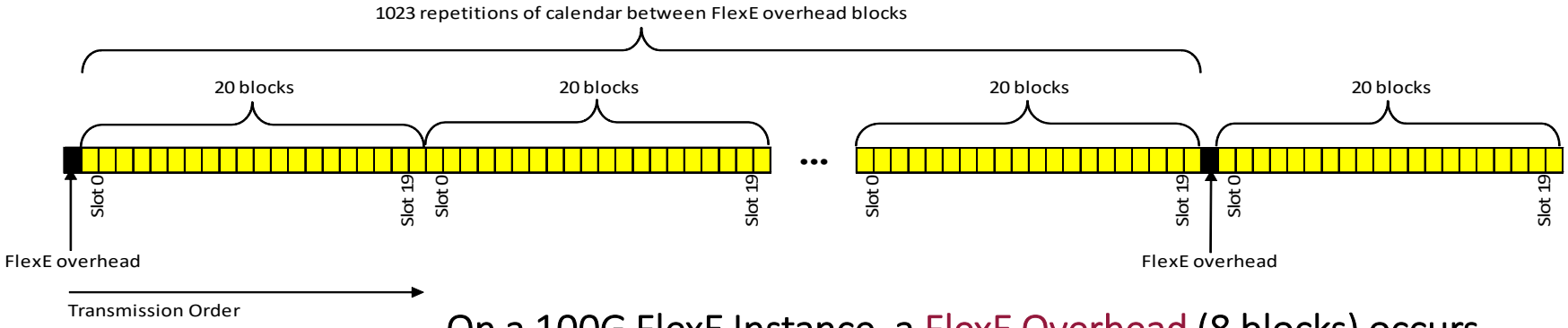
Node2

FlexE slots and overhead

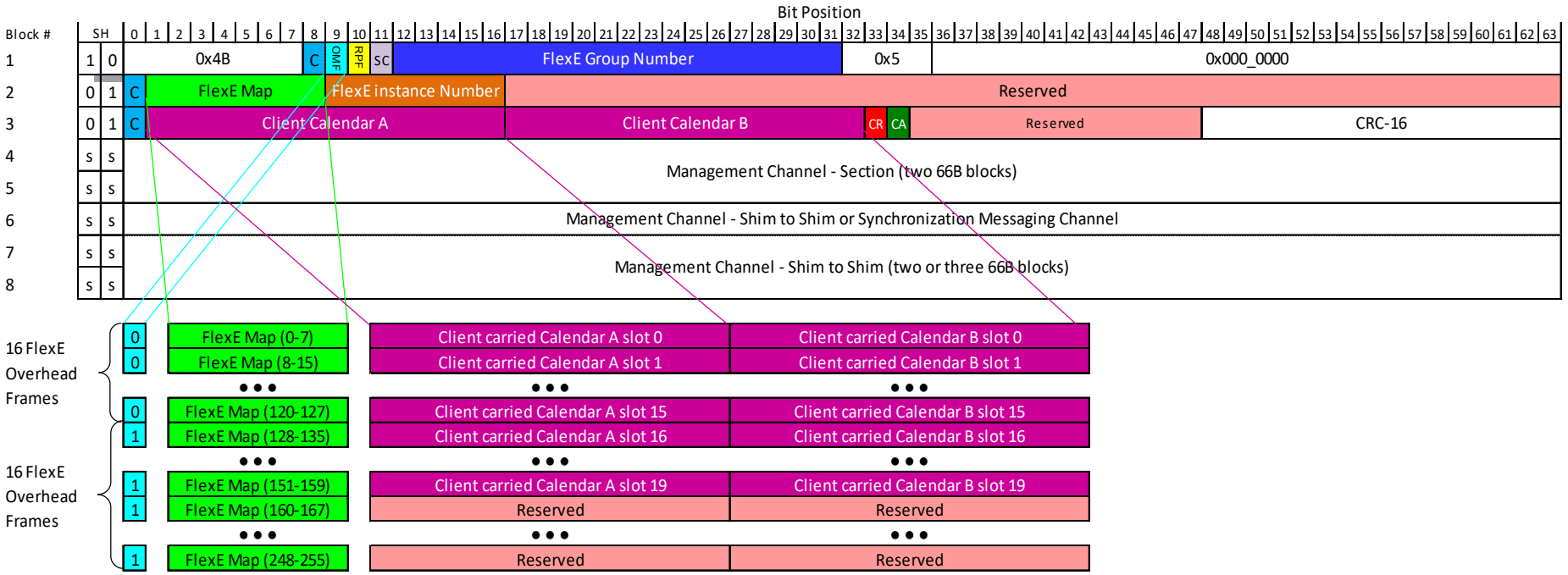


FlexE shim uses a calendar which assigns 66B block positions (or Calendar slots) to each of the FlexE Clients.

For a FlexE Group composed of n 100G FlexE Instances, the logical length of the calendar is 20n (slots).



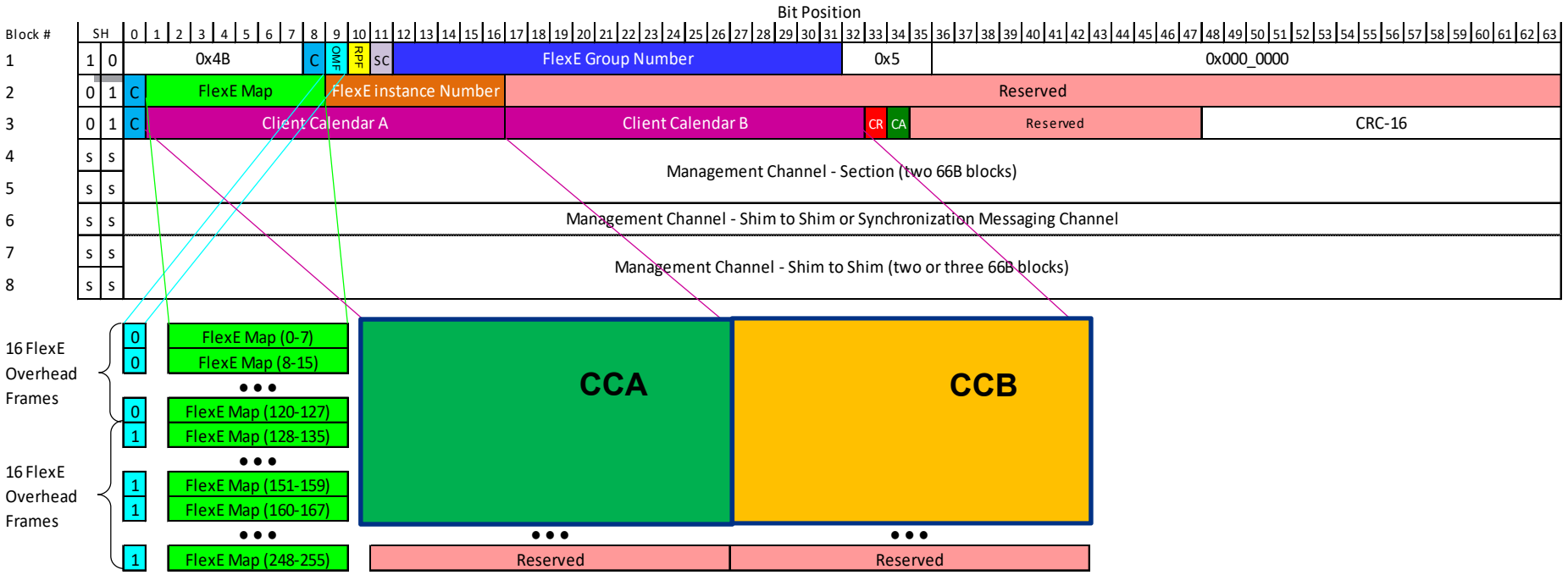
On a 100G FlexE Instance, a FlexE Overhead (8 blocks) occurs once per 104.77µs.



C = Calendar configuration in use (see 7.3.2)
OMF = Overhead Multiframe Indicator (see 7.3.1)
C = Calendar configuration in use (see 7.3.2)
RPF = Remote PHY Fault (see 7.3.8)
SC = Synchronization Configuration - see 7.3.5
CR = Calendar Switch Request (see 7.3.4)
CA = Calendar Switch Acknowledge (see 7.3.4)
s s = Valid sync header bits (01 or 1 0)

FlexE Map = Control of which 100G FlexE Instances are members of this group (see 7.3.3)
FlexE instance Number = Identity of this 100G FlexE instance within the group (see 7.3.3)
FlexE Group Number - See 7.3.6
Client (being programmed) - See 7.3.4
Reserved - See 7.3.7
Management Channels - See 7.3.5
CRC-16 - See 7.3.9
Synchronization Messaging Channel - See 7.3.5

FlexE Overhead Frame and Multiframe of each 100G FlexE Instance



C = Calendar configuration in use (see 7.3.2)
OMF = Overhead Multiframe Indicator (see 7.3.1)
C = Calendar configuration in use (see 7.3.2)
RPF = Remote PHY Fault (see 7.3.8)
SC = Synchronization Configuration - see 7.3.5
CR = Calendar Switch Request (see 7.3.4)
CA = Calendar Switch Acknowledge (see 7.3.4)
s s = Valid sync header bits (01 or 1 0)

FlexE Map = Control of which 100G FlexE Instances are members of this group (see 7.3.3)
FlexE instance Number = Identity of this 100G FlexE instance within the group (see 7.3.3)
FlexE Group Number - See 7.3.6
Client (being programmed) - See 7.3.4
Reserved - See 7.3.7
Management Channels - See 7.3.5
CRC-16 - See 7.3.9
Synchronization Messaging Channel - See 7.3.5

For a 100G FlexE Instance, a FlexE Multiframe consists of 32 Overhead Frames

FlexE Negotiation Protocol by Overhead (An example)