Loop Aggregation Baseline

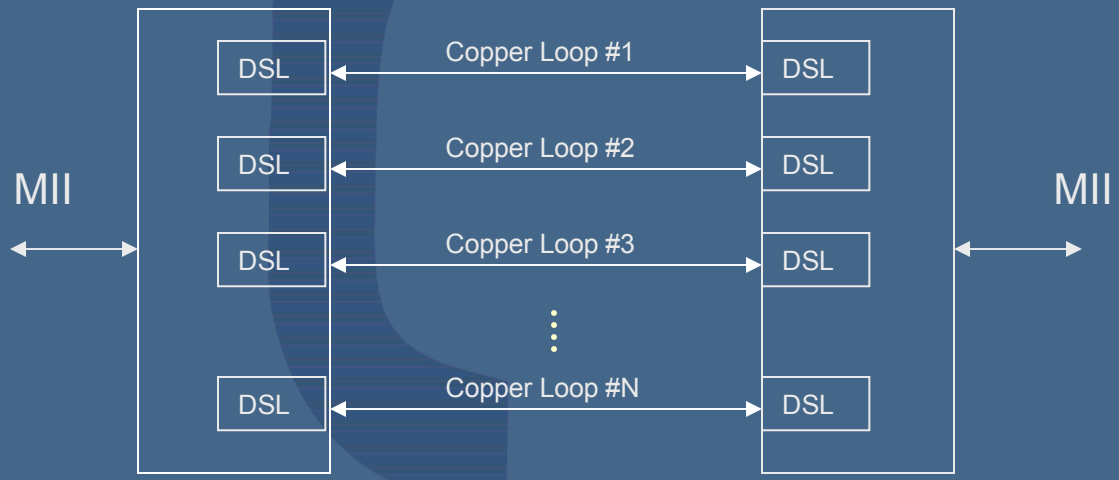
Klaus Fosmark
FirstMile Systems
klaus@firstmilesystems.com

With input from
Haim Yanko, Tioga,
and
Hugh Barrass, Cisco

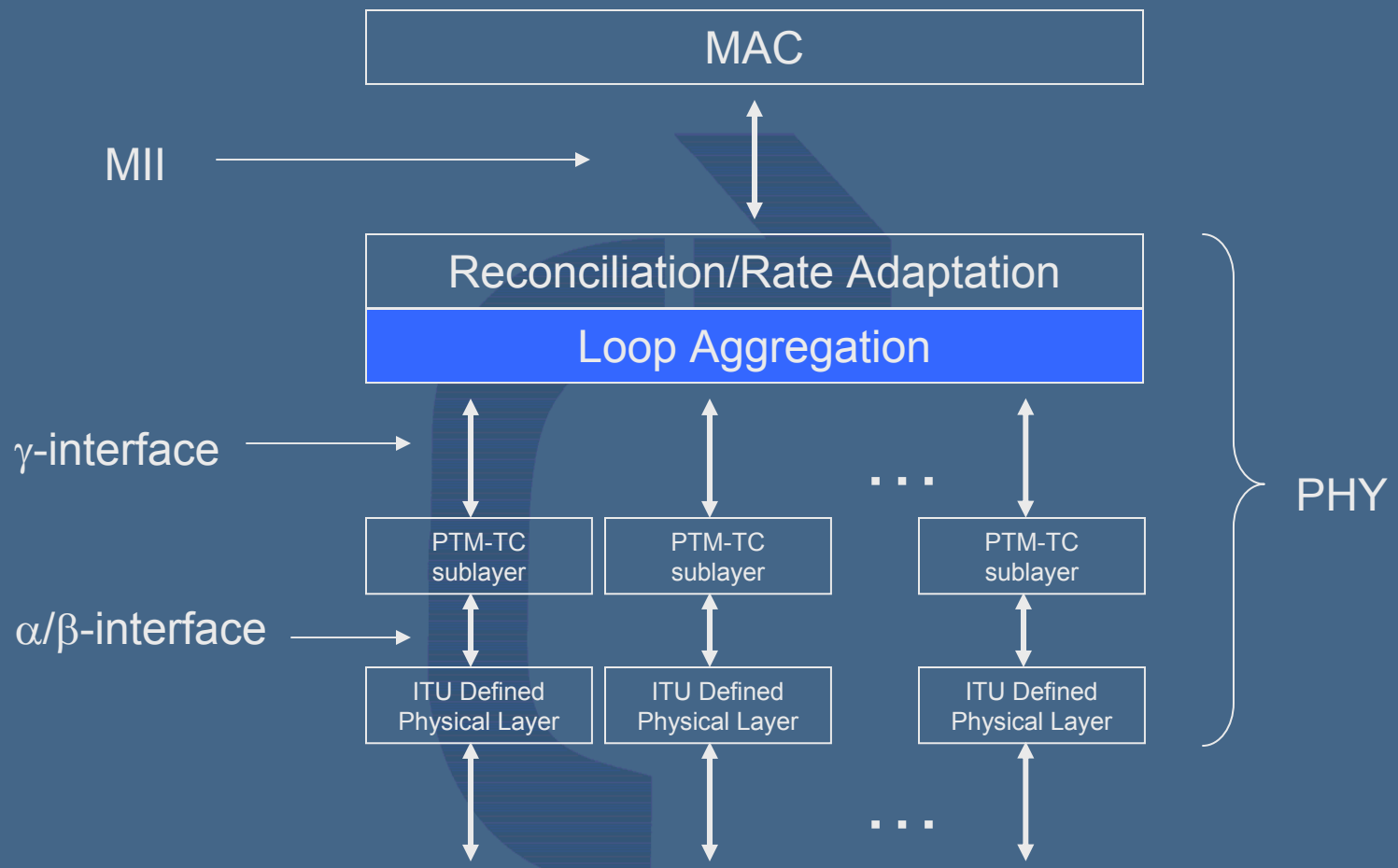
January, 2002

What is Loop Aggregation?

- Meets objective: “Include an optional specification for combined operation on multiple copper pairs”
- PHY Layer protocol for aggregation of up to N copper loops into one logical Ethernet link (N=8-32 TBD)
- Independent of PMD layer flavor of DSL
- Scalable and resilient to loop failures

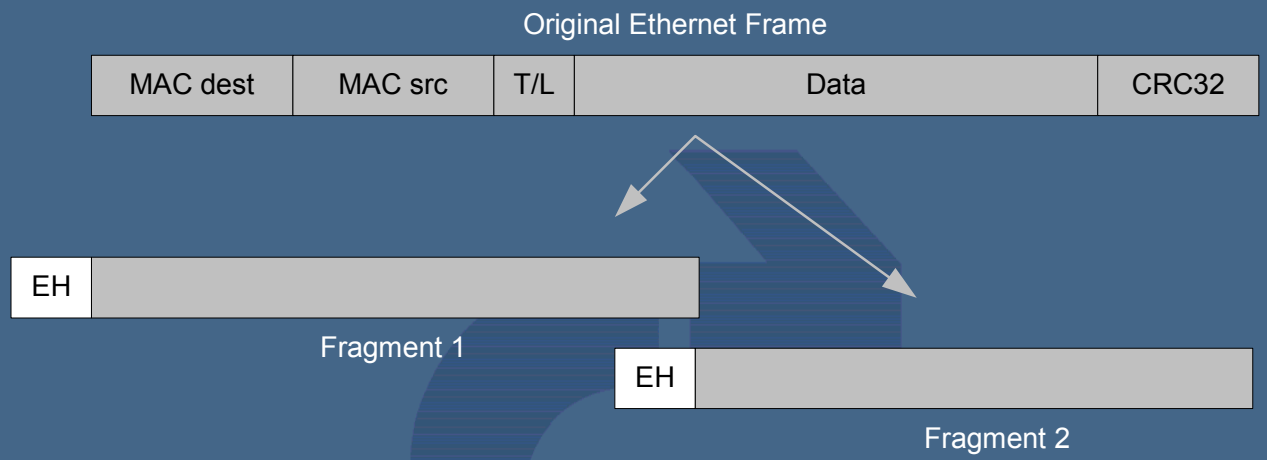


Protocol stack



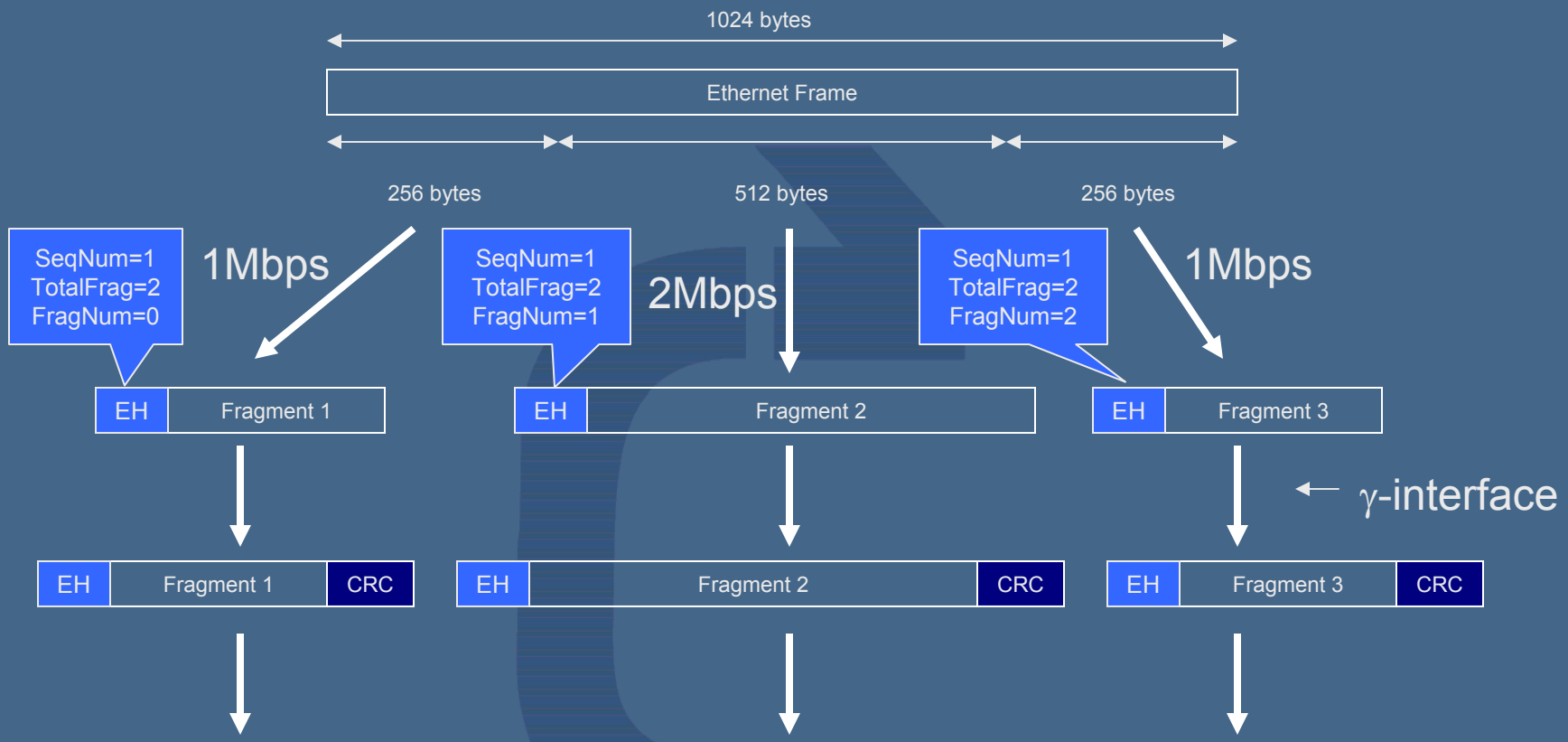
Fragmentation & Reassembly

EFM Protocol Encapsulation



- EFM Header (EH)
  - SeqNum - frame sequence number (10bit)
  - TotalFrag - # of other fragments that belongs to this Ethernet frame (3-5bit)
  - FragNum - fragment number (3-5bit)
- Underlying PTM-TC sublayer provides
  - HDLC framing
  - 0xFF 03 header (Can we use these?)
  - CRC checksum

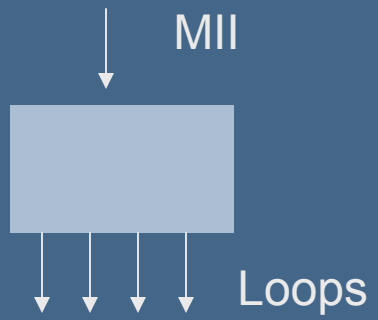
Example



- It does not matter which ports are connected to which, the protocol header implicitly determines how they are to be reassembled

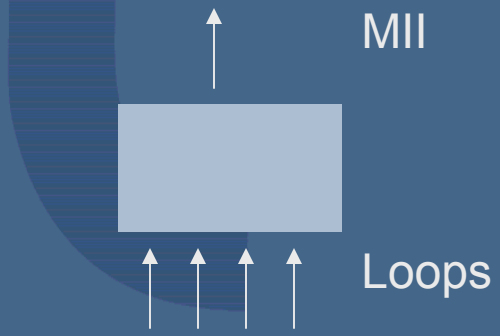
Fragmentation Procedure

- Ethernet frame from MAC layer
  - Determine N, the number of currently functional loops (if some loops are down, N will be smaller than the number of ports)
  - Slice up the frame into N fragments according to chosen multiplexing algorithm (see later)
  - Add EFM Header to all N fragments
  - Set SeqNum to SeqNum+1 from last frame sent
  - Set TotalFrag to one less that the number of loops (N-1)
  - Set FragNum to indicate fragment number of each fragment (it does not matter which fragment of the frame is sent on which loop)
  - Hold off transmission until no backpressure from any PTM-TCs, then send all N fragments in parallel across the N loops
  - In PTM-TC sublayer, CRCs are calculated and inserted on all N loops



Reassembly Procedure

- CRC is checked on each loop (PTM-TC)
  - if error, fragment is discarded
- Original Ethernet frame is reassembled
  - Using FragNum, TotalFrag, and SeqNum in the EFM Headers
  - Using chosen multiplexing algorithm (see later)
- If a fragment is received with SeqNum out of sequence the fragment is discarded

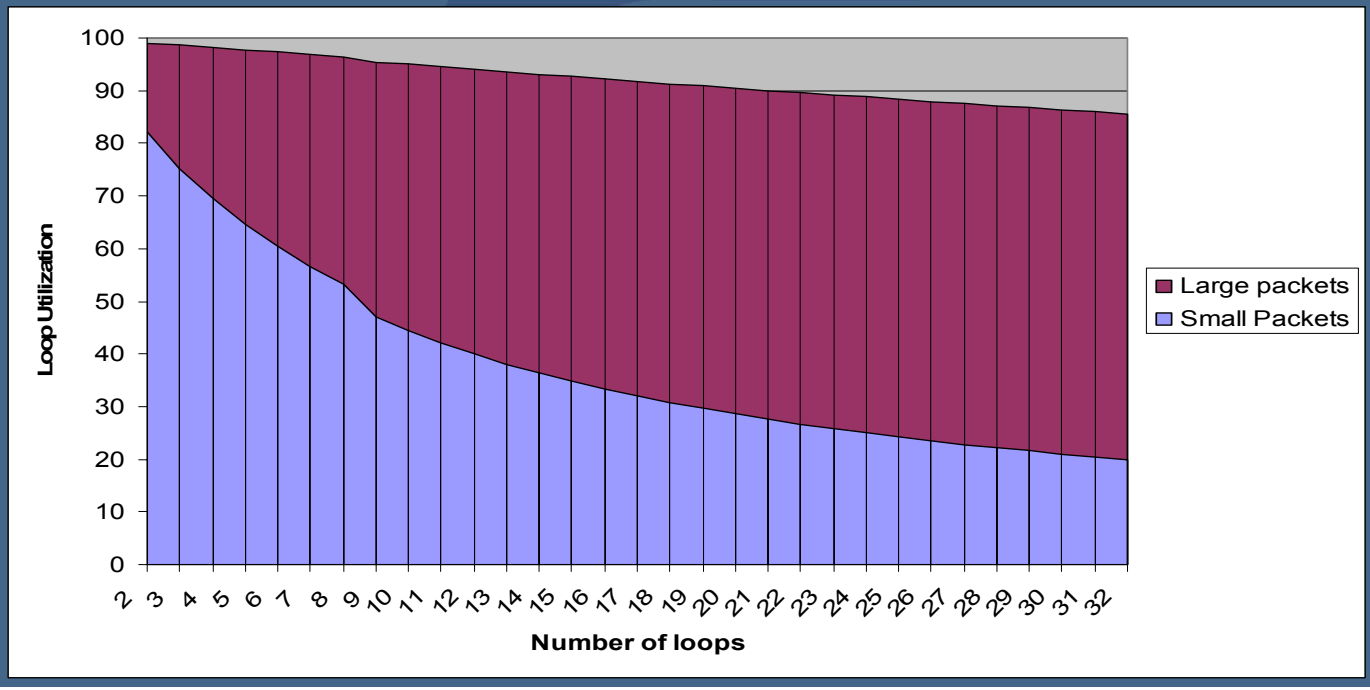


Resiliency

- A transmitter can in real time determine which of the connected loops are to be used (based on DSL link failures or bit error levels)
- The EFM header allows the fragmentation to only take place on a subset of the connected loops. The EFM header implicitly defines how many and which loops were used.
- The reassembly process can determine how many loops were used on a packet by packet basis

How Many Loops?

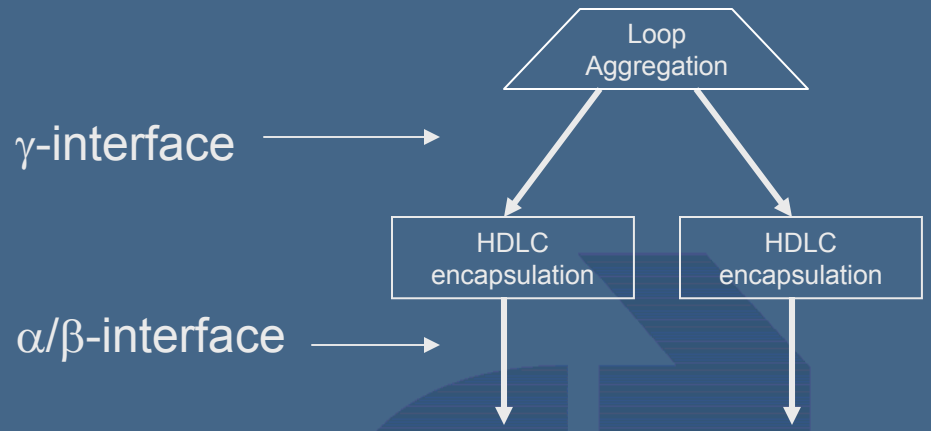
- Maximum number of loops that can be aggregated is implementation specific, but we need to pick a protocol limit!
- 8-32 loops, what does it cost?
  - More loops means smaller payloads per loop. I.e. more relative overhead.
  - More loops mean more bits needed in EFM Header:
  - N <= 8 loops means 2 bytes EFM Header
  - 8 < N <= 64 means 3 bytes EFM Header
  - N>32 should not be considered! (MDIO support)
  - No buffer cost (other than linear scale)



Open Issue

"Let's pick how many aggregated loops the 802.3ah Loop Aggregation Protocol shall support! Should it be 8, 24, or 32?"

The “HDLC Skew” issue



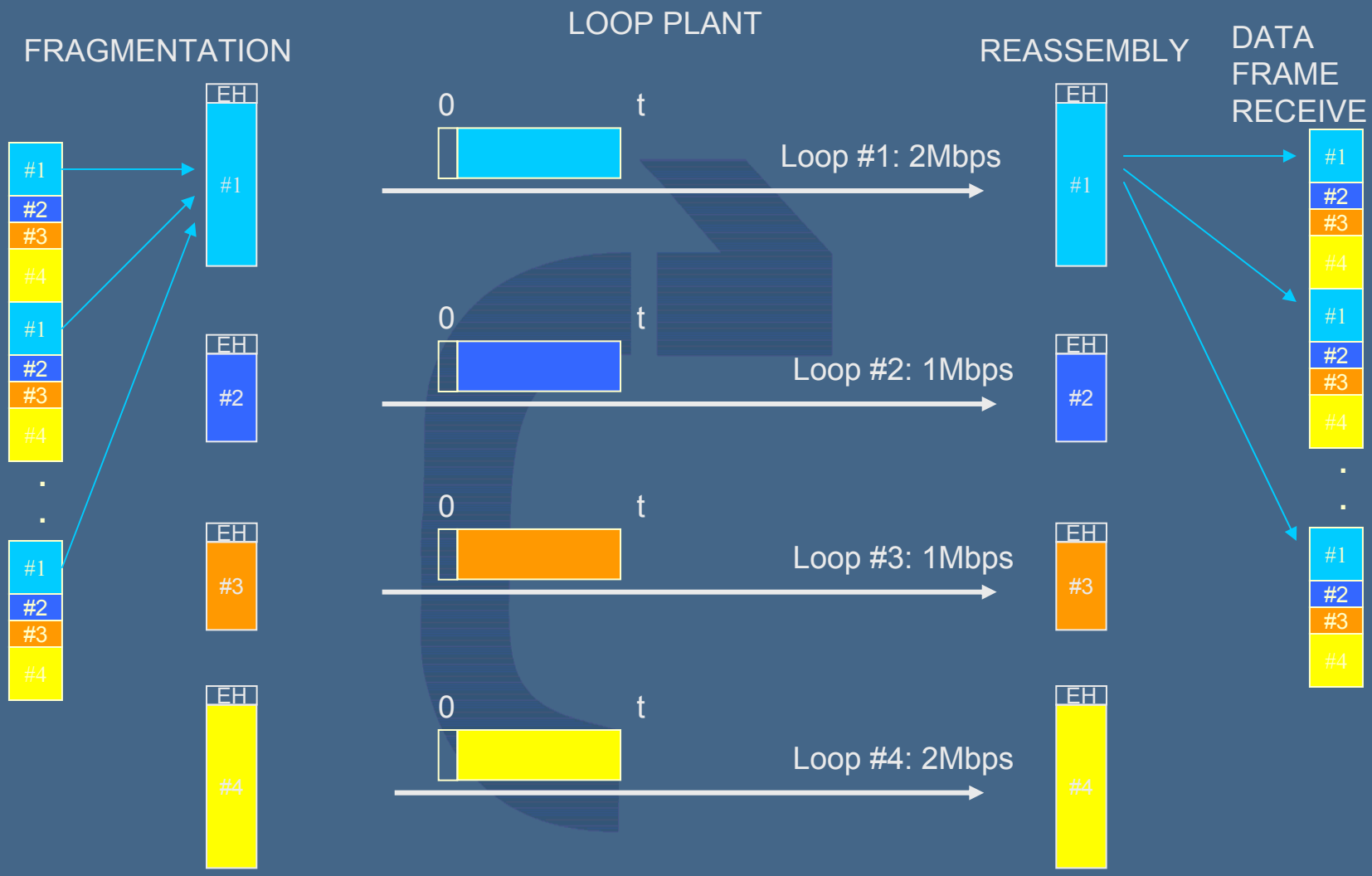
- ITU defined Packet Transfer Mode (PTM) defines use of byte synchronous HDLC encapsulation
- HDLC encapsulation makes the data stream longer than it was:
  - Data byte 0x7E is encoded as 0x7D-5E (two bytes)
  - Data byte 0x7D is encoded as 0x7D-5D (two bytes)
- Skew is dependant on content of packets
- Unless HDLC skew is compensated for, Loop Aggregation layer will not know the real transmission rates
- Can lead to lower loop utilization

Byte Mux or Packet Mux Fragmentation?

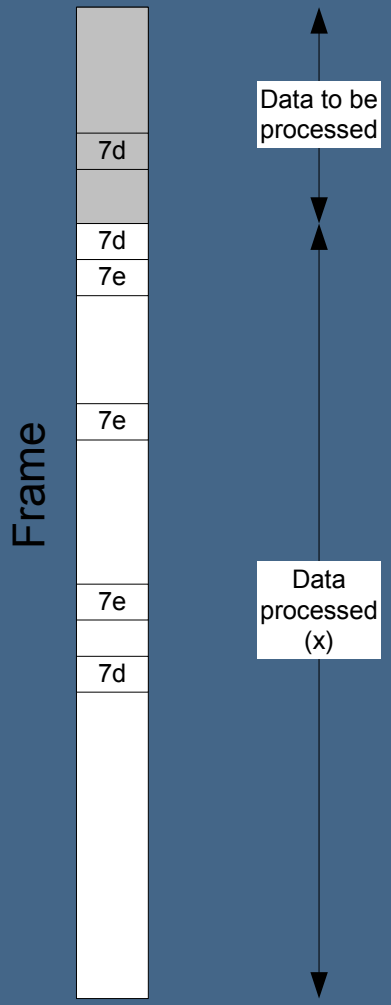
Two schemes for fragmentation presented:

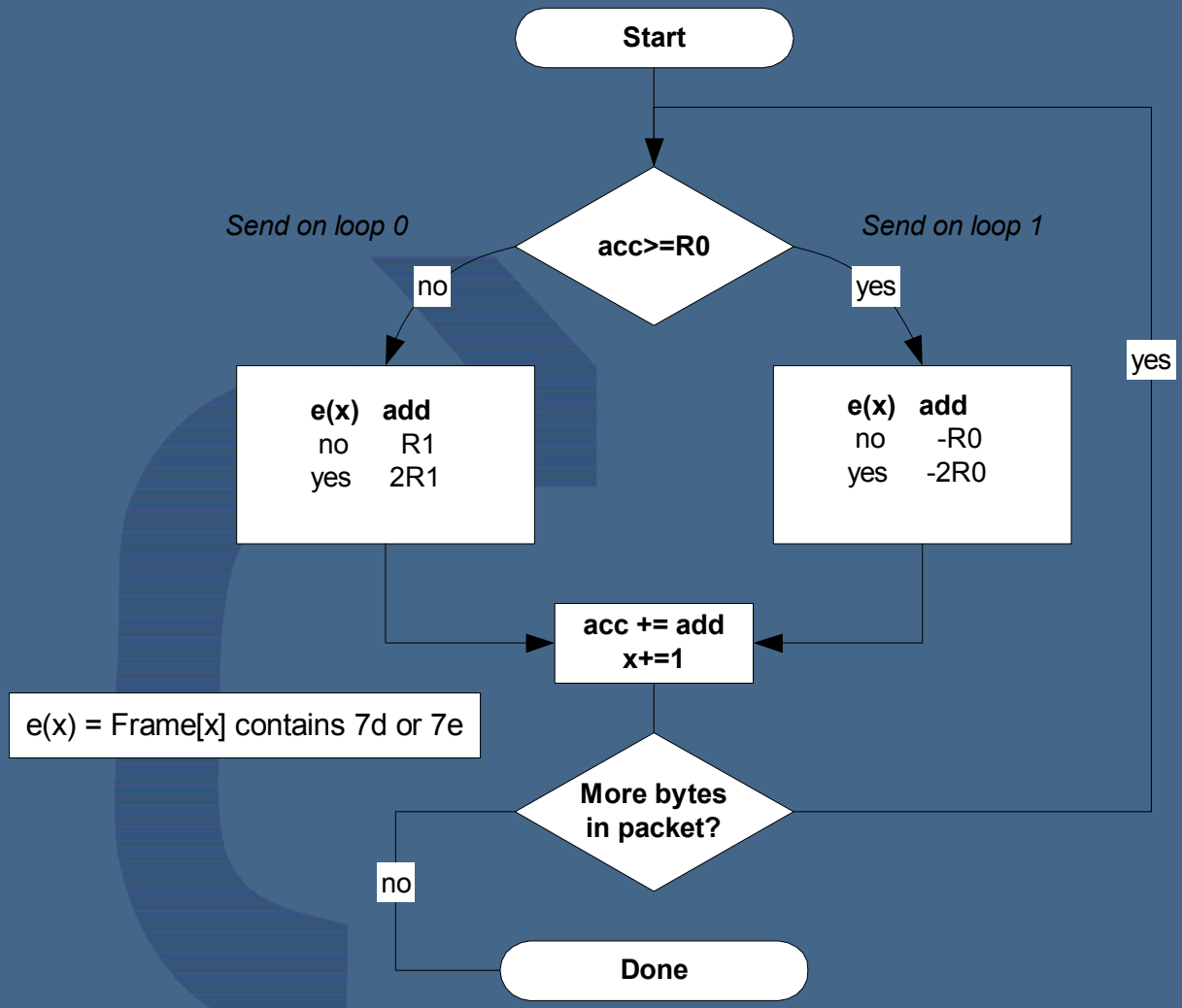
- Byte multiplex
- Packet multiplex

Byte Mux, Fragmentation and Reassembly



Byte Mux, Fragmentation Algorithm (2 loops)





- Incremental calculation (TX and RX end)
  - Algorithm details required to be in the standard

Byte Mux, Fragmentation Algorithm (N loops)

- Algorithm that works for N loops:
  - Initially, and each time a line rate changes:

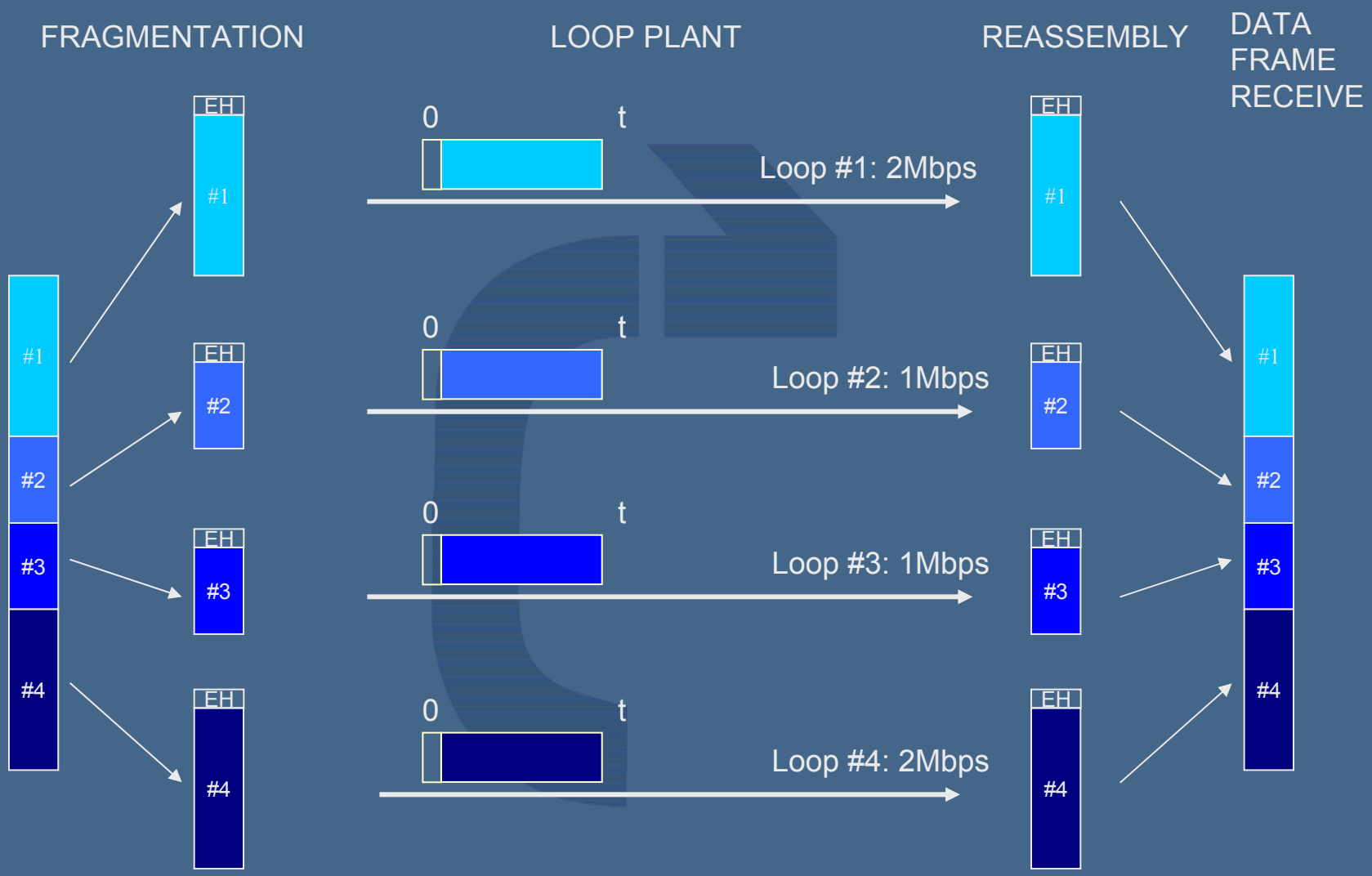
```
for i=1 to N do
   C[i] = G/R[i]
```

  - where G is the least common multiple (LCM) of R[1], R[2],…R[N].
  - For every packet:

```
Clear all A[i] ,  i = 1,2,,N
for each byte b in the frame do
{
  Find k where A[k] = min(A[i]) ,  i = 1,2,,N
  Send data byte to loop number k
  if b contains 0x7e or 0x7d
    f=2
  else
    f=1
  A[k] += f * C[k]
}
```

- Exact same implementation on both ends of loop

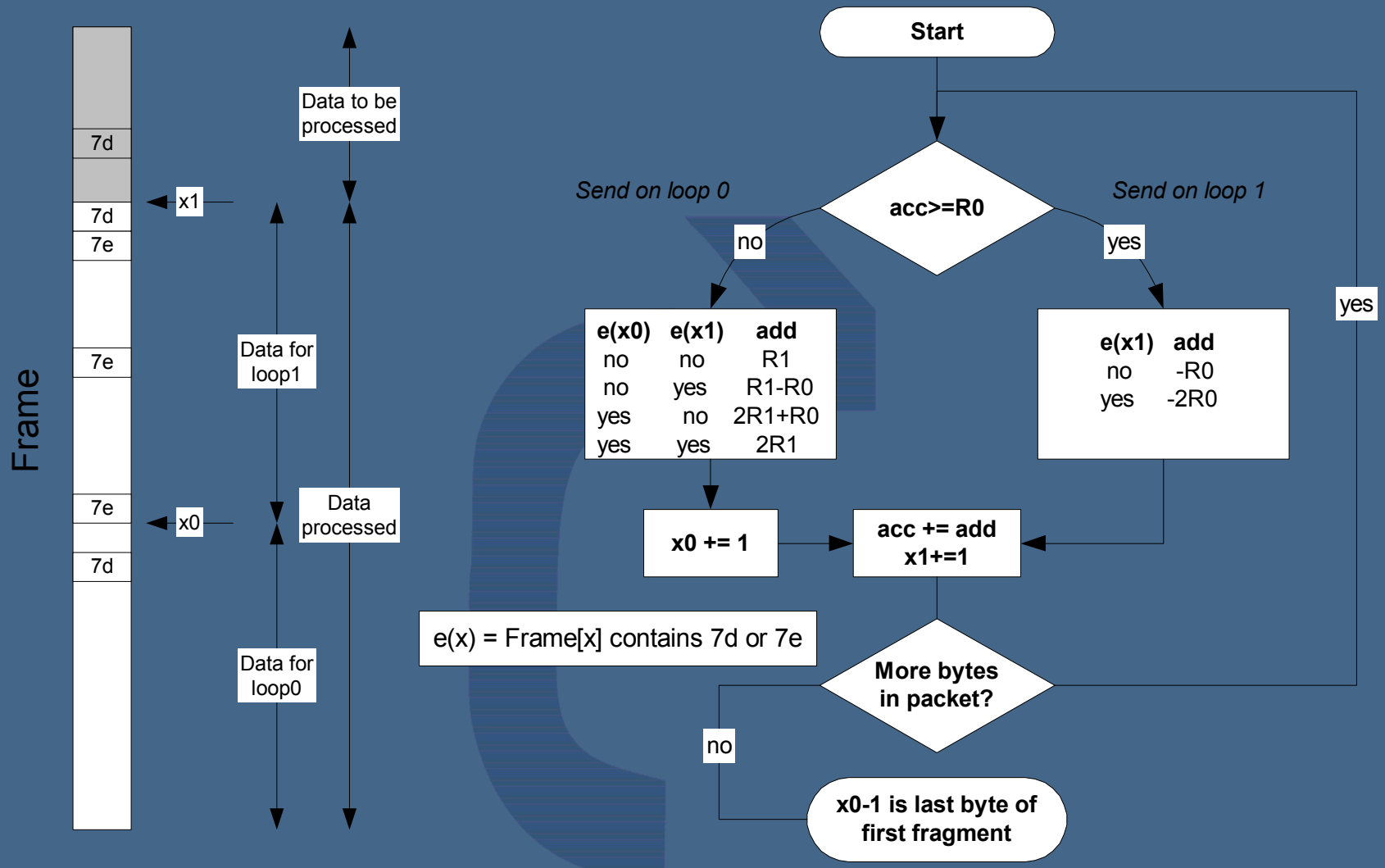
Packet Mux, Fragmentation & Reassembly



Packet Mux, Fragmentation Algorithm

- Fragmentation algorithm can be vendor specific, does not need to be defined in standard
- Fragmentation algorithm can optionally compensate for HDLC skew
- Fragmentation algorithm does not need to be known at receiver, it does not affect interoperability
- The following are examples of possible algorithms that do compensate for HDLC skew and are simple to implement

Packet Mux, Frag. Algorithm Example (2 loops)



- Incremental calculation (only TX end)
  - One pointer parameter (x0, x1,…) per loop

Packet mux, Frag. Algorithm Example (N loops)

- Algorithm that works for N loops:
  - Initially, and each time a line rate changes:

```
for i=1 to N do
   C[i] = G/R[i]
```

  - where G is the least common multiple (LCM) of R[1], R[2],…R[N].
  - For every packet:

```
Clear all A[i], x[i] ,  i = 1,2,,N
for each byte in the frame do
{
  Find k where A[k] = min(A[i]) ,  i = 1,2,,N
  for i=k to N do
  {
    if frame content in x[i] contains 0x7e or 0x7d
      f=2
    else
      f=1
    A[i]   += f * C[i]
    A[i+1] -= f * C[i+1], if i<N
    x[i]   += 1
  }
}
```

  - where the x's are the intersection pointers

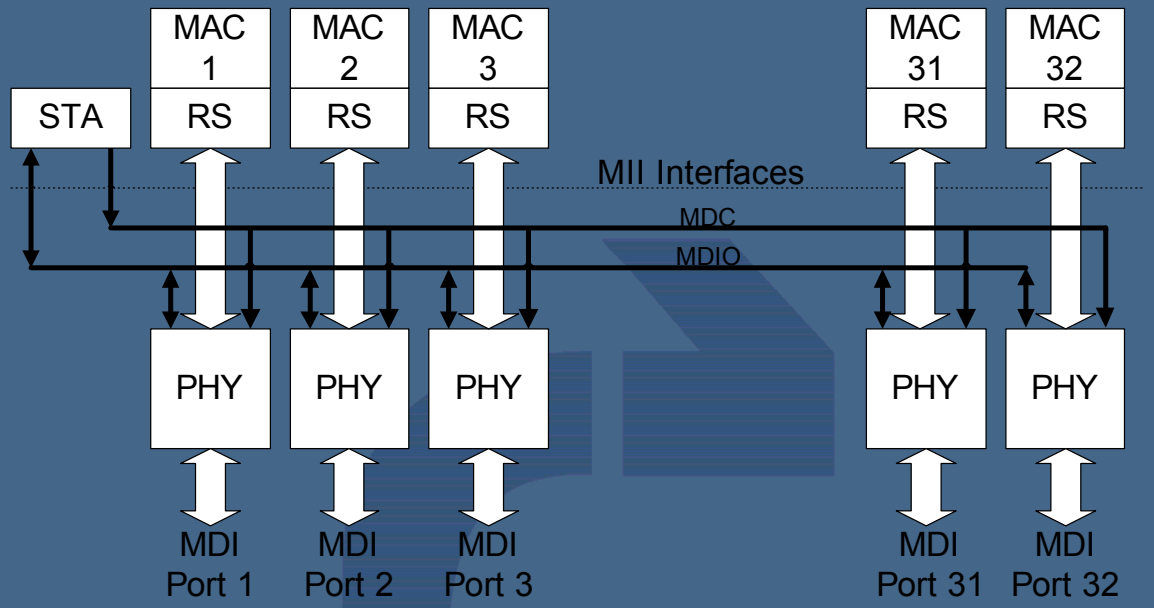
Comparison

	Byte Multiplex	Packet Multiplex
Packetization Delay	Data streaming is possible across MII	Ethernet frame has to cross MII before transmission can begin
Receiver independence	Receiver needs to know line rates	Receiver does not need to know anything about line rates
Real time synchronization	If a loop line rate changes, both receiver and transmitter need to discover that in sync with each other	No sync requirements
Fragmentation Math	Simple – but required in both transmitter and receiver	Without HDLC comp.: Can be very very simple, and only at transmitter With HDLC comp.: Slightly more complex in transmitter
Scaling	More options exist that allows a simple scaling of the math to support N loops. A specific solution much be defined and included in the standard	Several methods allows simple scaling of the math. Actual solution can be implementation specific.
Implementation flexibility	Exact transmitter and receiver scheme is defined. Register sizes, math accuracy and rounding has to be defined in standard	Allows competition. Allows simple implementations as well as better more complex, they all work.

Open Issue:

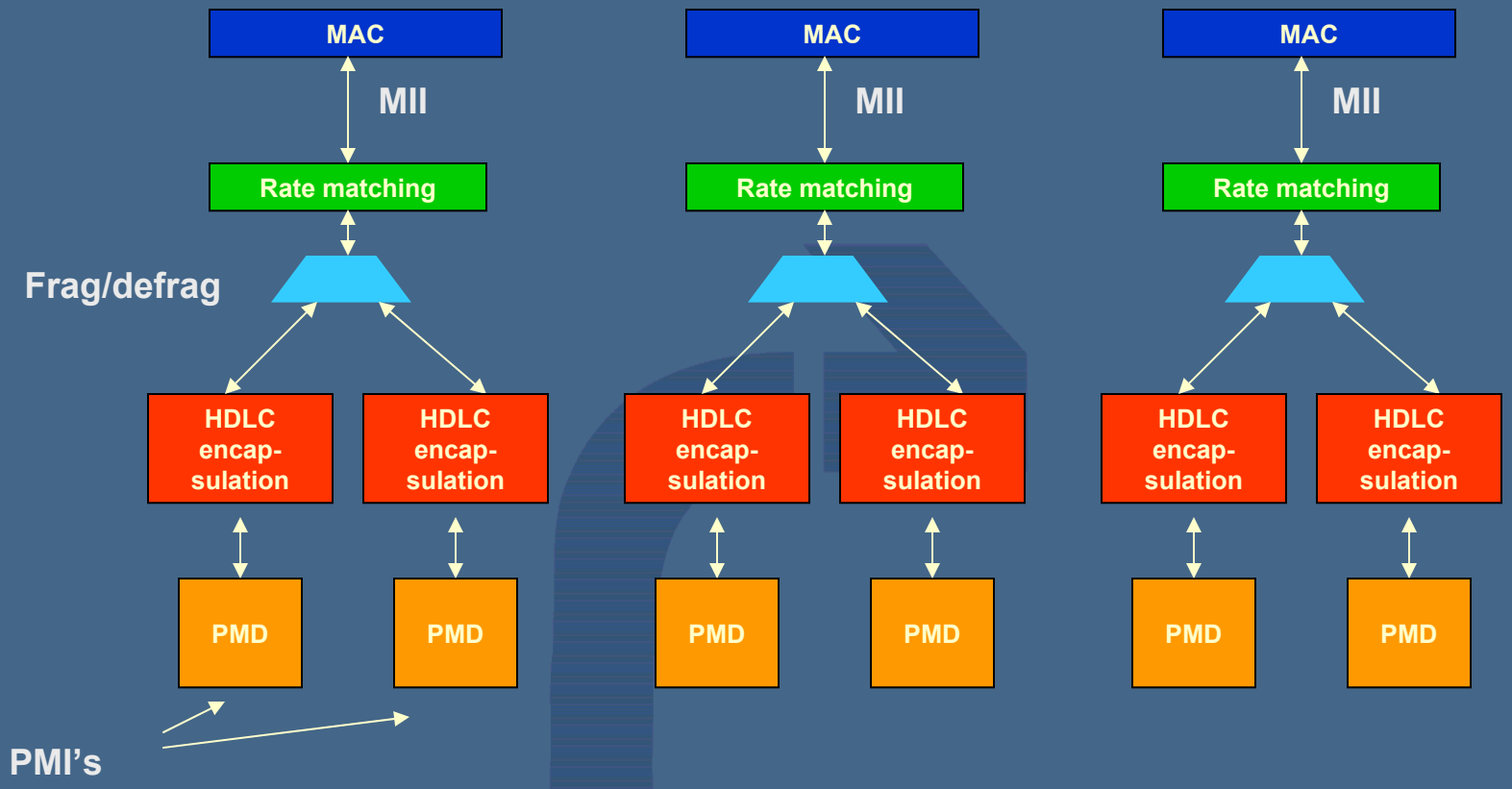
“Let’s pick one!”

PHY control



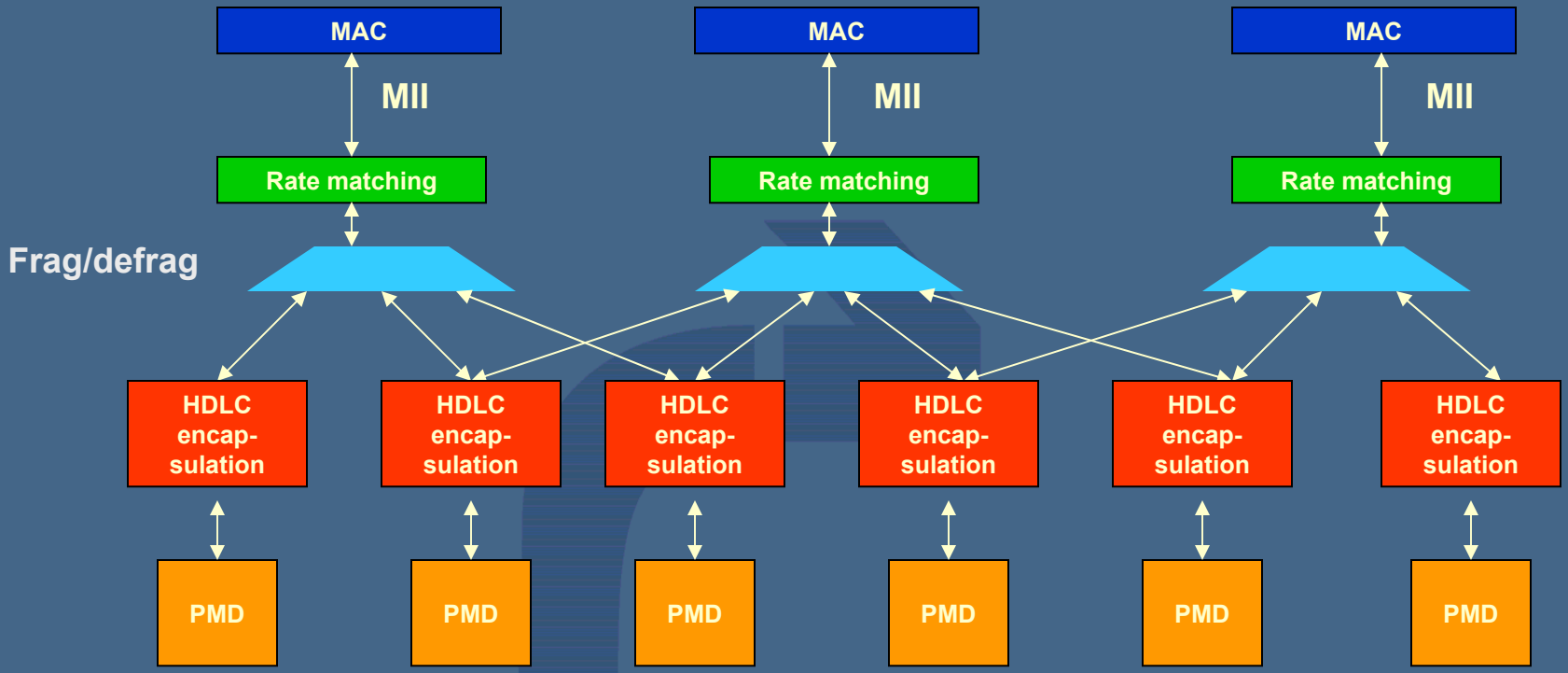
- MDIO/MDC is MII Management interface
- Allows layered addressing (used in XGMII) – Loop Aggregation is sublayer in PHY
- Need to define register set for Loop Aggregation sublayer
- Need to support wide range of system implementations…

Simplest system



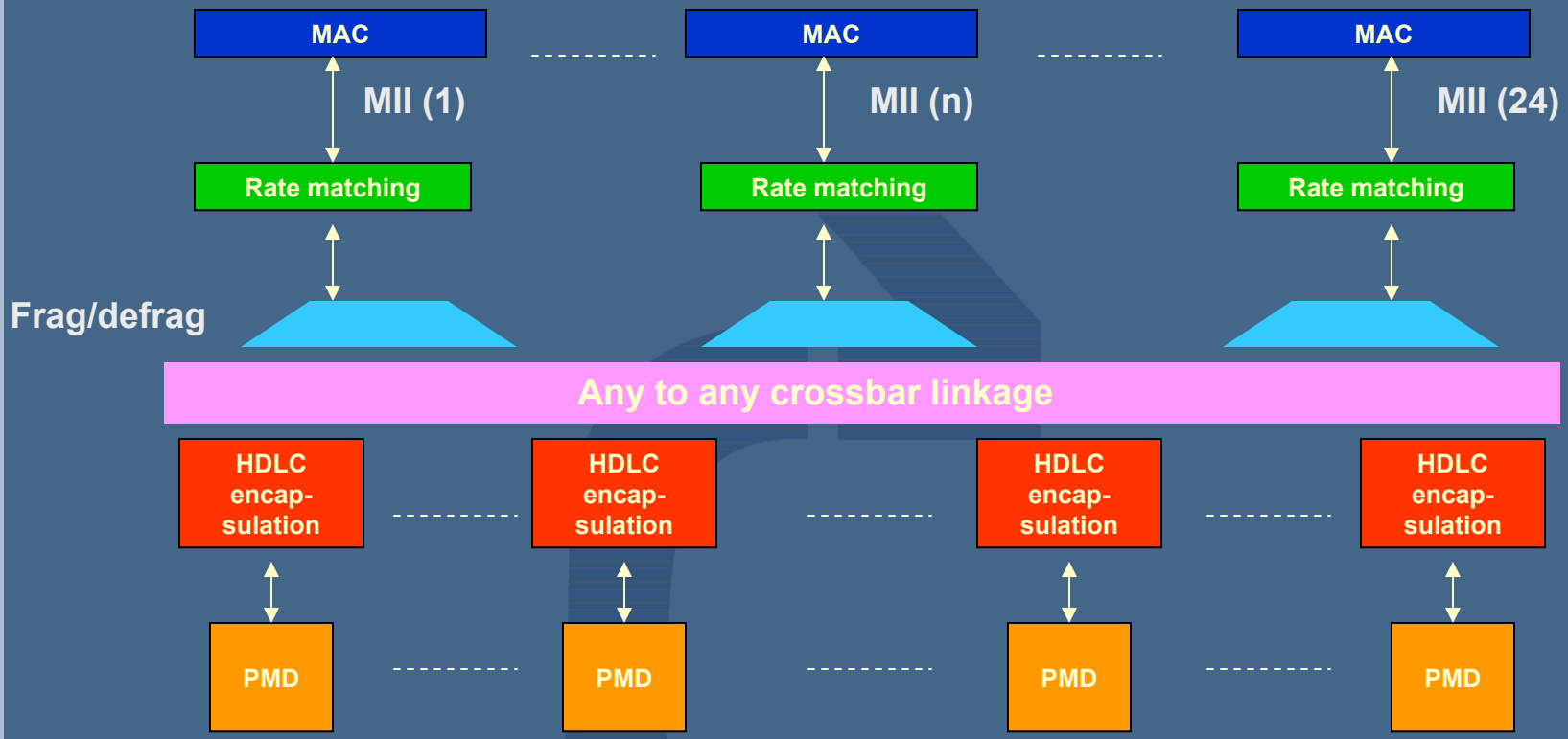
- Each MII is linked to 2 PMI's
- Either 1 or 2 loops aggregated
- Could be any integration from 2 PMI's upward

More complex system



- MII can link through 1, 2, 3 or 4 PMI's
- Allows more efficient use of loops
- Must be at least 6 PMI integration

Ultimate dream system



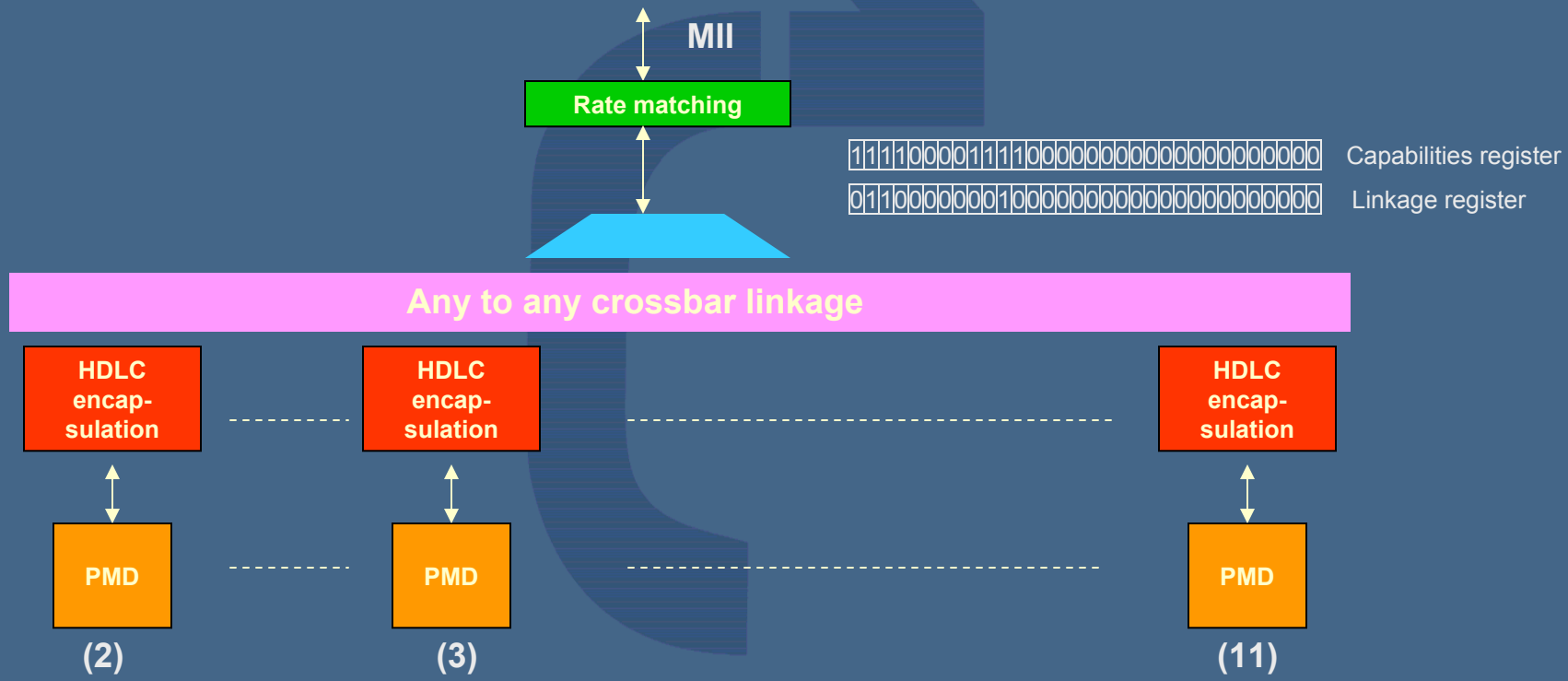
- Any connection between MII's and PMI's
- Implementation could use arbitrary number of MII's vs PMI's
- System vendor could choose optimal ratio

MMD register definition

- Presumably, 802.3ah will use XGMII style MII Management interface (indirect addressing, sublayer support)….
- “MII Manageable Device” register definition for the Loop Aggregation Sublayer
- There is one register set per MII (and there can be up to 32 MIIs):
  - 32 bit capabilities register (read-only)
    - Each bit corresponds to a PMD (up to 32)
    - If a bit is set, it indicates that PMD is available for Loop Aggregation on this MII
  - 32 bit linkage register (read-write)
    - Each bit correesponds to a PMD (up to 32)
    - If a bit is set, it indicates that PMD is currently part of Loop Aggregation for this MII (this is determined by system software)
- Smaller devices/systems do not need to implement 32 bit registers, only registers corresponding to the number of ports and number of MIIs in the system.

MMD Register Example

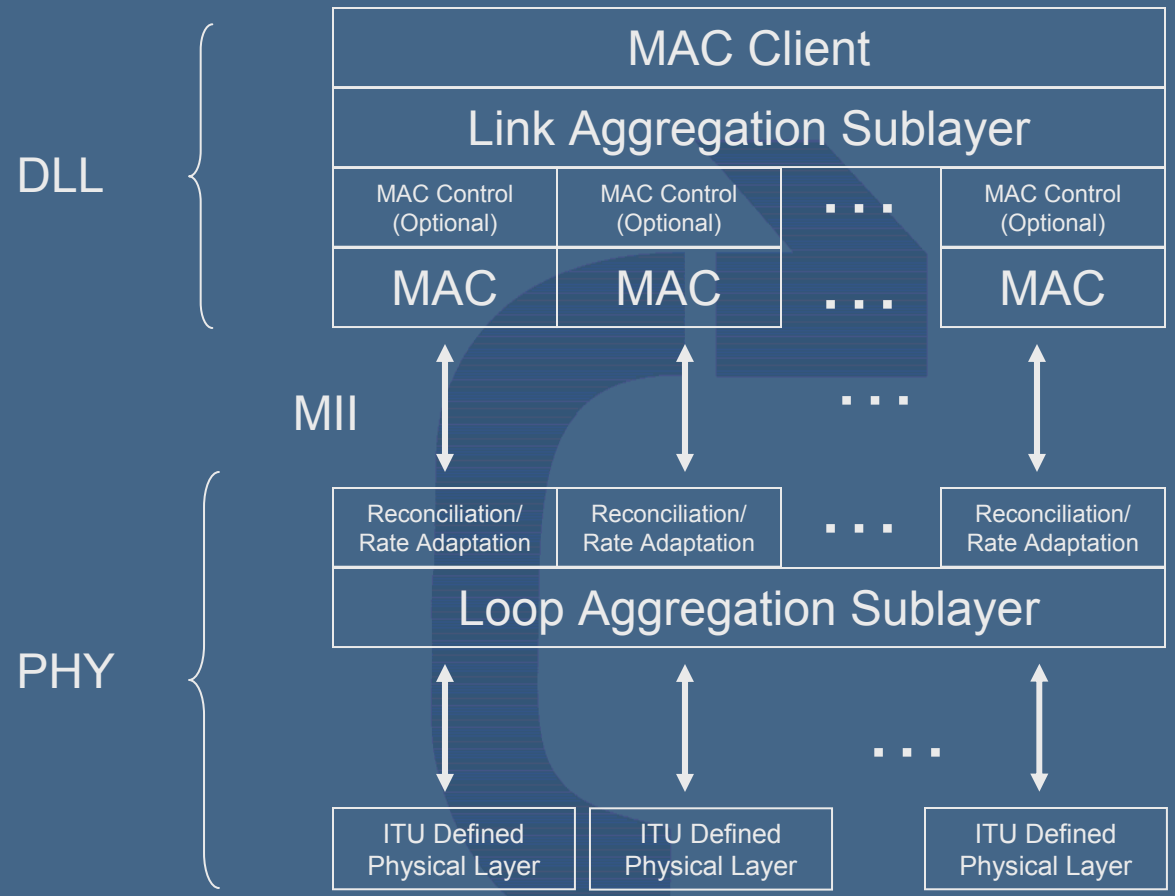
- In this example, device is capable of aggregating loops 1, 2, 3, 4, 9, 10, 11, and 12.
- Device is now configured to aggregate loops 2, 3, and 11.



Optional Auto Discovery Protocol

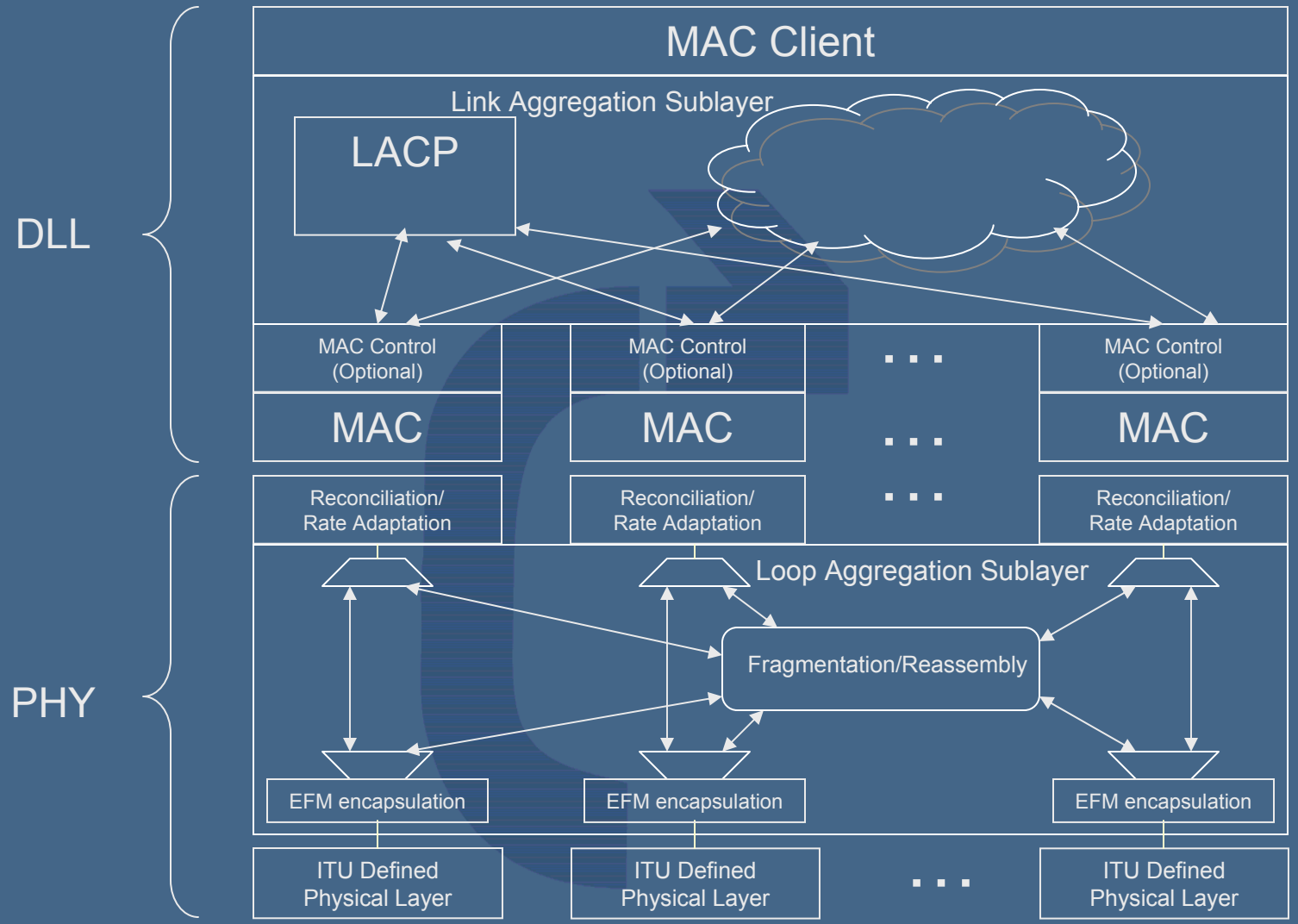
- Manual configuration is okay, but the ability to implement an automatic plug-and-play system is better…
- Allow optional use of higher layer control protocol to determine port configuration automatically.
- Reuse the Link Aggregation Control Protocol from IEEE 802.3ad (clause 43).
- Minor optional additional requirements in PHY
- Minor changes to clause 43 of 802.3.

Optional Auto Discovery Protocol Stack

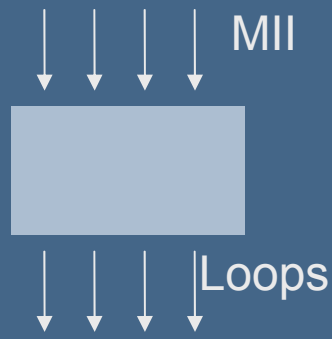


Optional Auto Discovery “Block Diagram”

Loop Aggregation Baseline



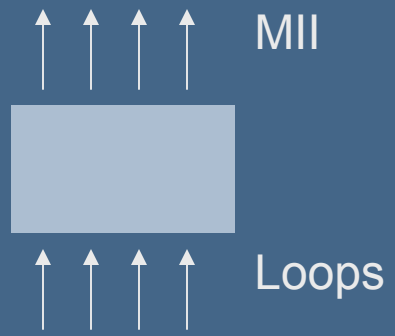
Optional Auto Discovery Frag. Procedure



- Identification of “Control” vs. “Data” Ethernet frames
  - MAC destination address of the Ethernet frame is checked
  - I.e. if MAC destination == Multicast Address used by LACP
- “Data” Ethernet frame from any MII from MAC layer:
  - Fragment frame as previously defined into N fragments
  - Fragment with FragNum=0 is destined for the loop corresponding to the MII where the frame came from
  - Send all N fragments in parallel across the N loops
- “Control” Ethernet frame from any MII from MAC layer (e.g. LACP and OAM):
  - Add EFM Header to frame
  - Set SeqNum to SeqNum+1 from last frame sent
  - Set TotalFrag to 0, set FragNum to 0
  - Hold off transmission until no backpressure from any PTM-TCs, then send the frame on loop corresponding to the MII where the frame came from
  - In PTM-TC sublayer, CRC is calculated and inserted on the loop

Optional Auto Discovery Reassembly Proc.

- CRC is checked on each loop,
  - if error, fragment is discarded
- Original Ethernet frame is reassembled
  - Using FragNum, TotalFrag, and SeqNum in the EFM Headers
- If a fragment is received with SeqNum out of sequence the fragment is discarded
- Ethernet frame is indicated on the MII port corresponding to the loop where the fragment with FragNum==0 was received



Optional Auto Discovery LACP Modifications

- To distinguish between Loop Aggregation and Link Aggregation, some minor additions are needed in Clause 43:
  - Since Loop Aggregation works on loops with different speeds, the Loop Aggregation Control implementation must use the same value for the Actor Key even though the line rates may be different.
  - Link Aggregation LAC PDUs and Marker PDUs use Slow_Protocols_Multicast MAC address with Subtype 0x01 and 0x02. Loop Aggregation should use same MAC address but define a new Subtype for LAC PDUs.
- Side effect: Much more than 100Mbit/s possible on aggregated loops, even though MIIs run 100Mbit/s each!

OAM Simplification

- To make our lives easier, let's decide that "OAM flows do not care about whether loops are part of an aggregation group or not".
- Requirement:
  - Need to support whatever OAM flows are defined on individual loops while the loop is still part of an aggregated group of loops.
  - OAM flow should not disrupt the aggregation function
- Solution will depend on how the OAM wars end… In-band/out-of-band/etc.

Summary

- Loop Aggregation baseline presented
- Scalable, resilient, DSL independent
- MDIO registers defined
- Optional Auto discovery method defined