Physical Layer Link Aggregation

Matt Squire

HATTERAS NETWORKS

Summary of Current Draft

Packet Sequence Number (10b)	Total Fragments (5b)	Fragment Number (5b)

Section 61.1.2.2.2 PHY LOOP AGGREGATION Transmit function:

- Determine the number of loops (N)
- Partition Frame into N parts depending on link speeds
- Determine sequence number and fragment number for each part
- Set sequence number & fragment number in EFM Header
- Hold off on transmission til no back-pressure
- Transmit to PTM-TC layer
- PTM-TC layer responsible for CRC on sub-packet

Section 61.1.2.2.3 PHY LOOP AGGREGATION Receive function:

- Check validate CRC of sub-packet at PTM-TC
- If any fragment errored, discard packet and start over
- Take one fragment from each loop
- Grab sub-packet with that sequence number from all loops with it, waiting if nec.
- Figure out if entire frame received by keeping track of number of fragments
- When all fragments available reassemble in order of fragment number
- Pass frame to MAC after reassembly

Current Draft Analysis (1)

Good points:

- Receive doesn't have to know about transmit, not even the number of lines used
- Allows vendor specific algorithms for product differentiation

HATTERAS NETWORKS

Current Draft Analysis (2)

Bad points:

- Hard limit on the number of loops supported (protocol header)
- Requires division (divides packet size to segment)
- Hold and wait strategy (must hold transmission til no backpressure on *any* loop)
- Complexity of two sequence number management (per packet, per fragment)
  - More potential error conditions
  - Must determine when all fragments received
- Redundant CRC protection for payload (per sub-packet and per packet)
  - Lots of extra overhead!
- Requires CRC to be in PTM-SC to cover HDLC encapsulation

Problems with Current Draft

1. Complicated fragmentation and inefficient use of sequence numbers
2. Wasted CRC in fragmentation
3. High overhead

Complicated Fragmentation and Inefficient Sequence Numbering

Two isn't better than one

Complicated Sequence Numbering

Complex and inflexible fragmentation

- Actually using two sequence numbers, one for "which packet" and the other for "which fragment within packet"
  - Twice as many error conditions
  - Really treating two sequence numbers as one sequence number with "gaps"
  - Re-assembly requires two loops (get minimum sequence number, get minimum fragment with that sequence number)
- Hard limit on fragmentation capacity (5-bit fragment number)
- Requires backpressure checks (I.e. wait til all links ready)
  - More latency – wait for all lines to be ready
  - Less bandwidth - no data flowing while waiting

Complicated Sequence Numbering

Proposal: Use a single sequence number with an "end-of-packet" and "start-of-packet" marker

Replace:

Packet Sequence Number	Total Fragments	Fragment Number

with

Sequence Number	EoP (1b)	SoP (1b)

Wow, looks easy, eh? See how much better life got!

But how do we use it?

HATTERAS NETWORKS

Complicated Sequence Numbering

Proposed Loop Aggregation Transmit:

- Choose a loop (algorithm need not be specified)
- Choose number of bytes to xmit on that loop (algorithm need not be specified)
- Increment and set fragment sequence number in EFM Header
- Set EoP/SoP in EFM Header as appropriate
- Transmit to PTM-TC layer on selected loop

Section 61.1.2.2.2 PHY LOOP AGGREGATION Transmit function:

- **Determine the number of loops (N)**
- **Partition frame into N parts** depending on link speeds
- Determine **sequence number and fragment number** for each part
- Set **sequence number & fragment number** in EFM Header
- **Hold off on transmission til no back-pressure**
- Transmit to PTM-TC layer
- PTM-TC layer responsible for CRC on sub-packet

Easily vary # of loops in use (not always N), & disparate rates – best utilization
No waiting for backpressure (don't choose loop that's backed up!)
Simpler, less work – one sequence number, not two – no need to know #fragments

HATTERAS NETWORKS

Complicated Sequence Numbering

Proposed Loop Aggregation Receive:

- Determine next sequence number expected on any loop, waiting if necessary
- Grab that fragment
  - If EoP then pass buffer up to MAC
  - If unexpected SoP, flush buffer til next SoP
  - If buffer > maxFrameSize or errored fragment, then flush buffer til next SoP
  - Else throw fragment into current packet buffer

Section 61.1.2.2.3 PHY LOOP AGGREGATION Receive function:

- Check validate CRC of sub-packet at PTM-TC
- If any fragment errored, discard packet and start over
- **Take one fragment from each loop**
- Grab sub-packet with that sequence number from **all loops** with it, waiting if nec.
- **Figure out if entire frame received by keeping track of number of fragments**
- When all fragments available reassemble in order of fragment number
  - **Either grab fragments in order, or use more complex re-assembly**
- Pass frame to MAC after reassembly

Stop treating sequential assembly with two sequence numbers – simpler
Easy re-assembly – sequential buffer til EoP – no need to know #fragments

Benefits of simplified sequencing (1)

- Flexibility
  - Receive doesn't have to know about transmit, not even the number of lines used
  - Allows vendor specific transmit algorithms for product differentiation (more flexible loop use)
  - Supports greater number of loops - limited only by sequence wrap
  - Lower overhead – more loops does not mean tiny fragments!

HATTERAS NETWORKS

Benefits of simplified sequencing (2)

- Latency
  - Lower latency - no hold and wait for backpressure across all loops
- Bandwidth
  - No waiting implies more bits down pipes
- Simplicity
  - Less complexity with single sequence number
- Less overhead
  - Single sequence number more operationally efficient than two sequence numbers

HATTERAS NETWORKS

Wasted CRC Redundancy

Enough is enough

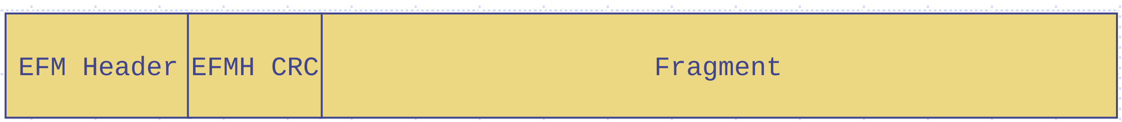
Current CRC Use

Current Draft CRC handling

- Based on HDLC PTM mode of xDSL systems
- Each fragment gets own 16-bit FCS
- This is in addition to packet 32-bit FCS on Ethernet frame!
- An additional 2xN octets of overhead per frame (N is the number of loops)
- Ugh! Can you say redundant? Ugh! Can you say redundant?

Only unprotected information is EFM header

- Add (smaller?) CRC to protect small EFM header
- No need to protect packet data again and again



HATTERAS NETWORKS

Benefits of CRC only on EFM Header

- Less overhead, less processing
- Doesn't alter Ethernet FCS behavior
  - Is a fragment CRC error an Ethernet packet CRC error? I.e. how do we count it?
- Applies protection only where needed

Higher Overhead

Right-sizing the header

HATTERAS NETWORKS

Current Overhead

Main components of EFM overhead are

- Two sequence numbers (15-bits)
- Fragment protection (16-bits)
- Number of fragments (5-bits)

Lots of per fragment overhead

Where do these numbers come from?

- Support up to 32-pair
- 16-bit CRC
- 15-bit sequence number

What sizes are required?

HATTERAS NETWORKS

High Overhead – Number of pairs

Why 32-pair?

- Seemed like a half-decent number in Raleigh
- It's a power of 2! Utilizes a binary number space
- Vast majority of support focused on 2-4 pair
- 16 is a big stretch for real deployments
- 32 is pulling a muscle, and not in a good way
- 32 pairs span across at least 2 bundles
  - How likely is that?

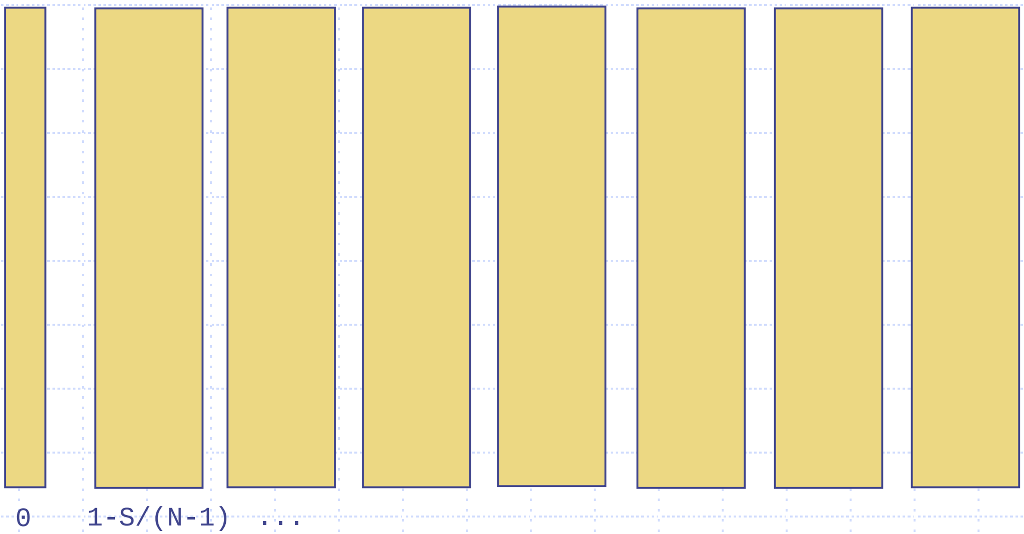
Proposal:

- Limit maximum number of supported pairs to 24 (one binder group)
- Note that proposed header does not limit via #fragments field, only limited by sequence number space

HATTERAS NETWORKS

High Overhead – Sequence Number

- Assume aggregating N loops
- Assume differential rate of loops <= R
- Assume fragments have max/min fragment size ratio M/m
    - Likely related to differential rates
- What is worst case? Send max fragment down slowest link, many min fragments down faster links



- To wrap sequence number, send S/(N-1) min size fragments down other links
- Each of these is received RM/m faster than max frag on slow link

High Overhead – Sequence Number

- Assume aggregating N loops
- Assume differential rate of loops <= R
- Assume fragments have worst case fragment differential of M/m
- Sequence number S must be so S>=R(M/m)(N-1)
- E.g. Say one supports 24 aggregated loops(N), min fragment size of 64 (m) max of 512 (M), loops rates at most 8:1(R) , then
  - S>=8*(512/64)*23~(2^11)
- Although the synchronization aspect has not been defined yet, it is likely that synchronization will require some kind of split horizon algorithm, resulting in another bit
- Conclusion: Need >= 12 bits for sequence number

High Overhead – CRC

CRC protects EFM header

- What are the effects of an error missed by CRC?
  - Possible incorrect re-assembly
    - If frame FCS fails and packet discarded
  - Possible re-ordering of packets
    - Fragment can be entire packet!
  - Possible blocking of receive queue with corrupt sequence number
    - Will likely cause flush and re-synchronization of sequence numbering
    - **Lots** of trouble

HATTERAS NETWORKS

High Overhead – CRC

- How big should CRC be?
  - DSL gives 10^{-7} (2^{-21}) bit error rate
    - Actually better, errors bursty
  - Probability of error in EFM header < 2^{-25} (assumes header/fragment <=1/16)
  - N-bit CRC fails to detect error 2^{-N} times
  - 8-bit CRC leads to undetected fragment probability < 2^{-33}

Conclusions on EFM Header

1. Need >= 12 bits for sequence number
2. Need >= 8 bits for CRC
3. Need 2 bits for EoP and SoP
4. Couple of bits left over

12-bit Sequence Number	EoP (1b)	SoP (1b)	Rsvd (2b)	8-bit CRC

Summary of Proposals

1. Simplify sequence number use and processing (single sequence number with start/end of packet markers)
2. Apply CRC only where needed (I.e. EFM header) not across data
3. Limit to 24-pair (binder) for calculation purposes
4. Think about correct sizes for header fields
    - Initial suggestions included
        - 12-bit Sequence number
        - 8-bit CRC

HATTERAS NETWORKS