Sync PDUs and the MAC Stack

Help needed from 802.3 to properly process IEEE 1588 / P802.1AS sync PDUs

Norman Finn

Cisco Systems

Timestamping

Typical timestamping protocol:

- L2 or L3 packet has timestamp fields.
- As packet leaves or enters a port, the port fills in the timestamps.
- Example: ITU-T Y.1731 EtherOAM Delay Measurement packets.
  - 3 packet types
  - 4 places to put a timestamp, 2 for input, 2 for output

Typical timestamping protocol:

- Good news:

  Measurements can be extremely accurate, since they are based on actual transmission/receipt time.

  One-way measurements can be defined.

- Bad news:

  Packet must be modified at a very low layer to get the actual transmission/receipt time.

  CRC (and perhaps other higher-layer checksum) must be recomputed.

  Protocol cannot be protected (or perhaps even recognized) by 802.1AE MAC security.

  Every new protocol of a similar nature requires a respin of the low-level port ASIC.

IEEE 1588, IEEE 802.1AS

- Can you get the good news without the bad news?:

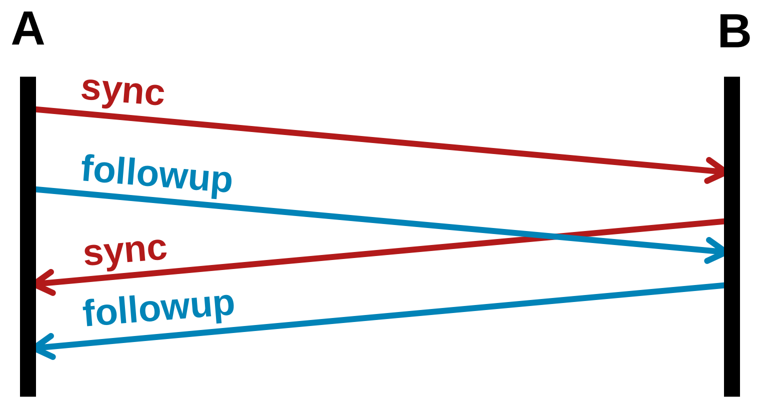
  YES!

- IEEE 1588 defines “sync” and “followup” packet types.

  Sync is recognized, but not modified, by the hardware.

  Timestamps on sync are recorded internally.

  Software recovers timestamps and transmits them in followup packet, that carries the timestamps in standard format.



IEEE 1588, IEEE 802.1AS

- Good news:

  Measurements can be extremely accurate, since they are based on actual transmission/receipt time.

  One-way measurements can be defined.

- Good news :

  Packet must be detected, but not modified at a very low layer to get the actual transmission/receipt time.

  No checksum is recomputed.

  Protocol can be fully protected by 802.1AE MAC security.

  New protocols of a similar nature need not require a respin of the low-level port ASIC.

IEEE 1588, IEEE 802.1AS

- Followup packet looks like typical (inserted) timestamp packet.
- The sync-followup protocol is very slightly less accurate than an inserted timestamp protocol if the two devices' clocks have random variations that can be significant in the time between the sync and the followup. (But, such variations would make the protocol less useful, anyway.)
- See IEEE 1588 for protocol details. http://grouper.ieee.org/groups/1588

Implementing sync packets: first cut

Naive approach to sync processing

Not the recommended approach

- L2 or L3 packet has timestamp fields.
- As packet leaves or enters a port, the port fills in the timestamps.
- Example: Y.1731 Delay Measurement packets.
  - 3 packet types
  - 4 places to put a timestamp, 2 for input, 2 for output

Sync processing: phase 0

(The naive approach)

- Add IEEE 1588 sync recognition to low-level ASIC.
- Record timestamp whenever packet is transmitted or received.
- Software recovers timestamp and generates followup.
- Problems:

  What about the next protocol to come along?

  What if multiple syncs pass through quickly?

  How do you match a timestamp to a particular packet?

  How do you recognize packets obscured by IEEE 802.1AE MAC security?

Sync processing: phase 1

(First cut at problem solving)

- What about the next protocol?

  Use programmable registers so that new sync types can be recognized as they are developed.

- Multiple syncs?

  Provide FIFOs to pass timestamps to the software for processing.

- How do you match sync packets to timestamps?

  Extract ID information from the packet to record along with the timestamp. (Counting fails because of 802.1AE, 802.3X, etc.)

- What about MACsec?

  Sacrifice accuracy by putting MACsec below timestamping; or

  Reduce security by running MACsec in integrity protection mode, without obscuring any data; or

  Don’t secure the sync packets.

Sync processing: phase 1

(First cut at problem solving)

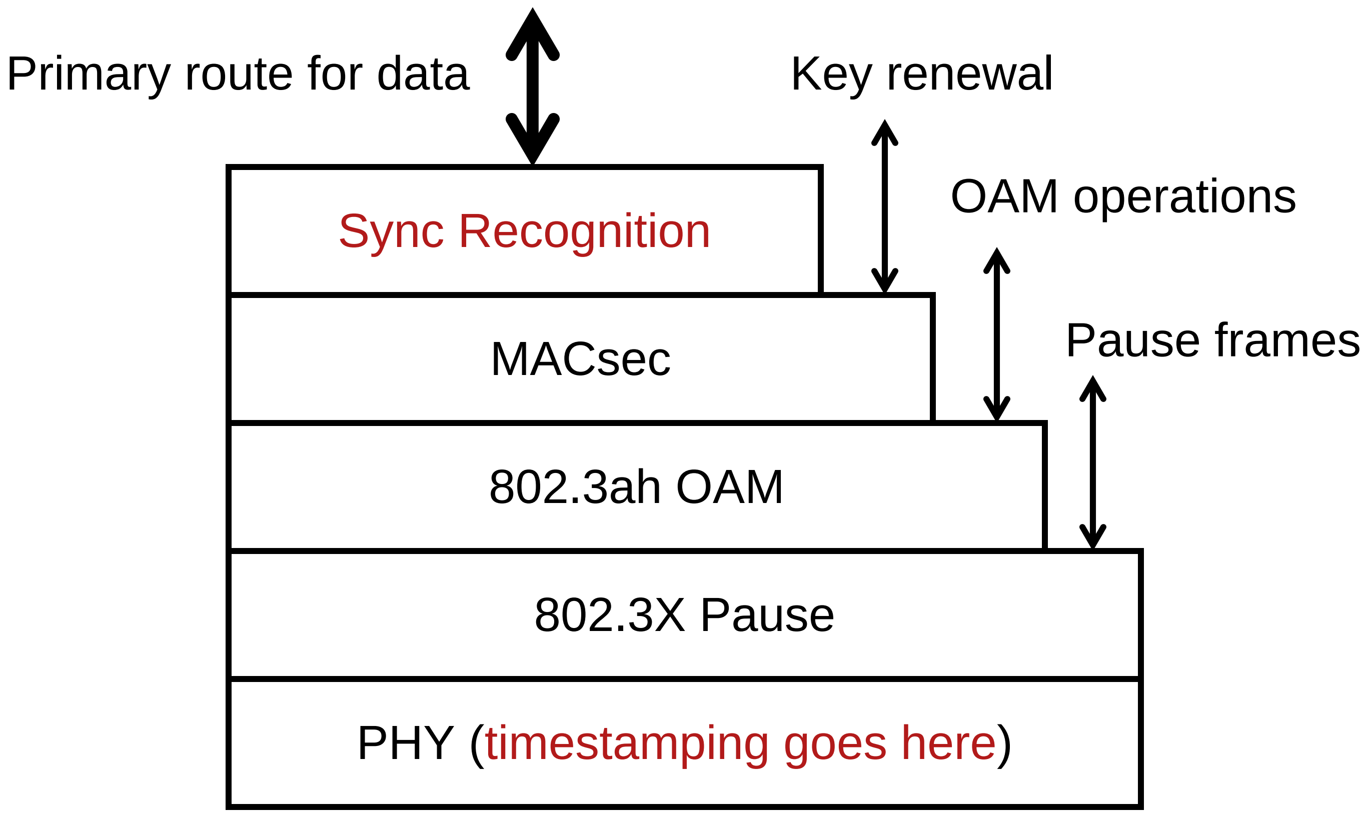
- And, we still have two issues:
- The low-level (near the PHY) ASIC must parse the packet, skipping over some number of more-or-less complex tags (e.g. 802.1Q, IP, etc.) to see if this is or is not a sync.
- There are still problems with MACsec scrambling the fields that identify the frame as a sync.

Implementing sync packets: a solution

Sync processing: Better Solution

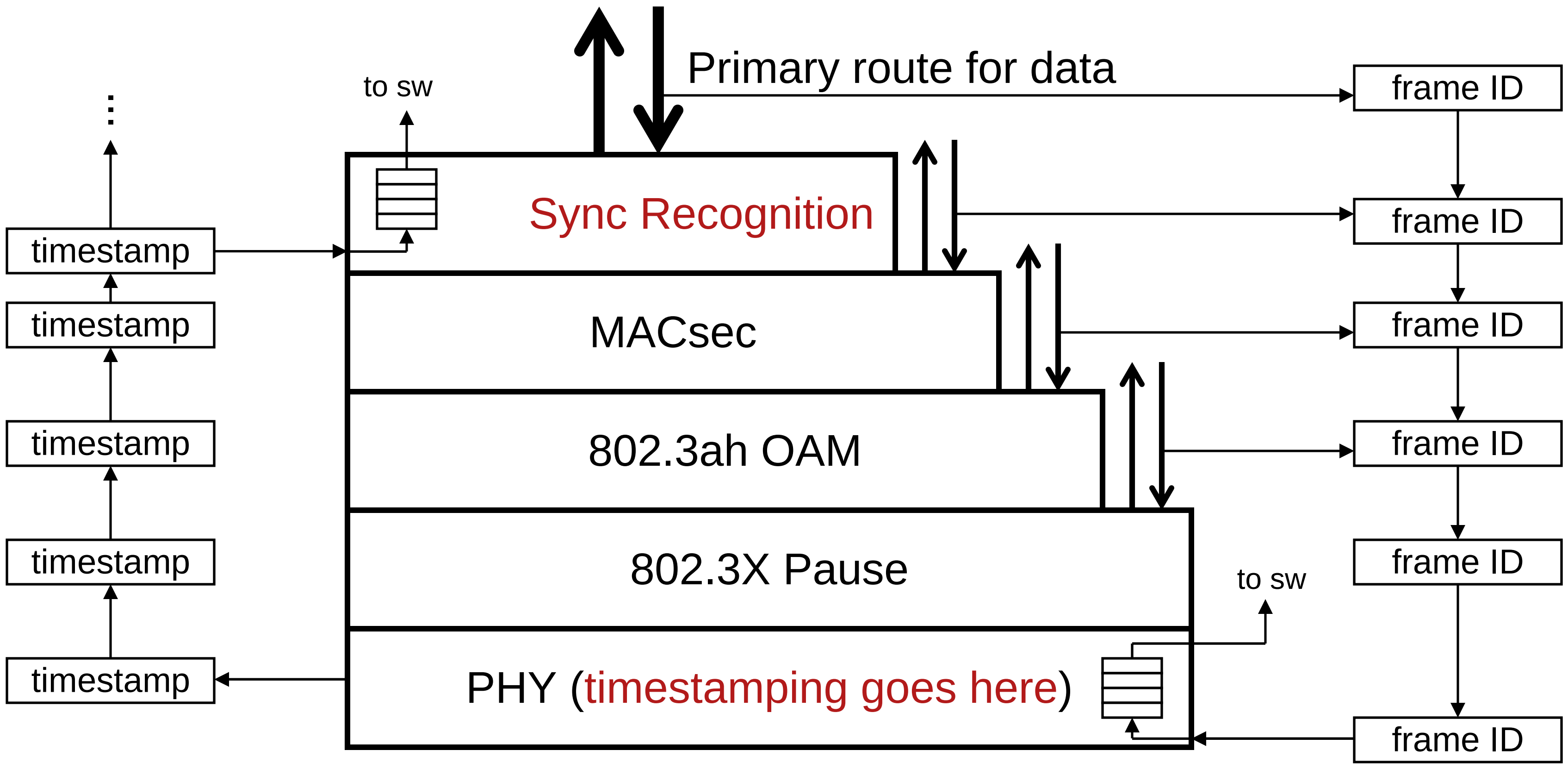
- If sync recognition is above MACsec, we can use MACsec to secure timing right along with other data.
- If timestamping is at the lowest levels, we can get the maximum accuracy.
- So, we separate the sync recognition from the timestamping; then, we get the best of both worlds.
- The cost is extra information that must be passed along with the packet through the MAC stack.

Sync processing: Better Solution



- It’s not always easy to match the data that passes through sync recognition with the timestamps because other paths insert/remove frames into/from the stack.

Sync processing: Better Solution



- Every received frame carries a timestamp up the stack.
- Every transmitted frame carries an ID down the stack.

Sync processing: Better Solution

- The frame ID in every transmitted frame carries:

  An indication of whether a timestamp is to be recorded.

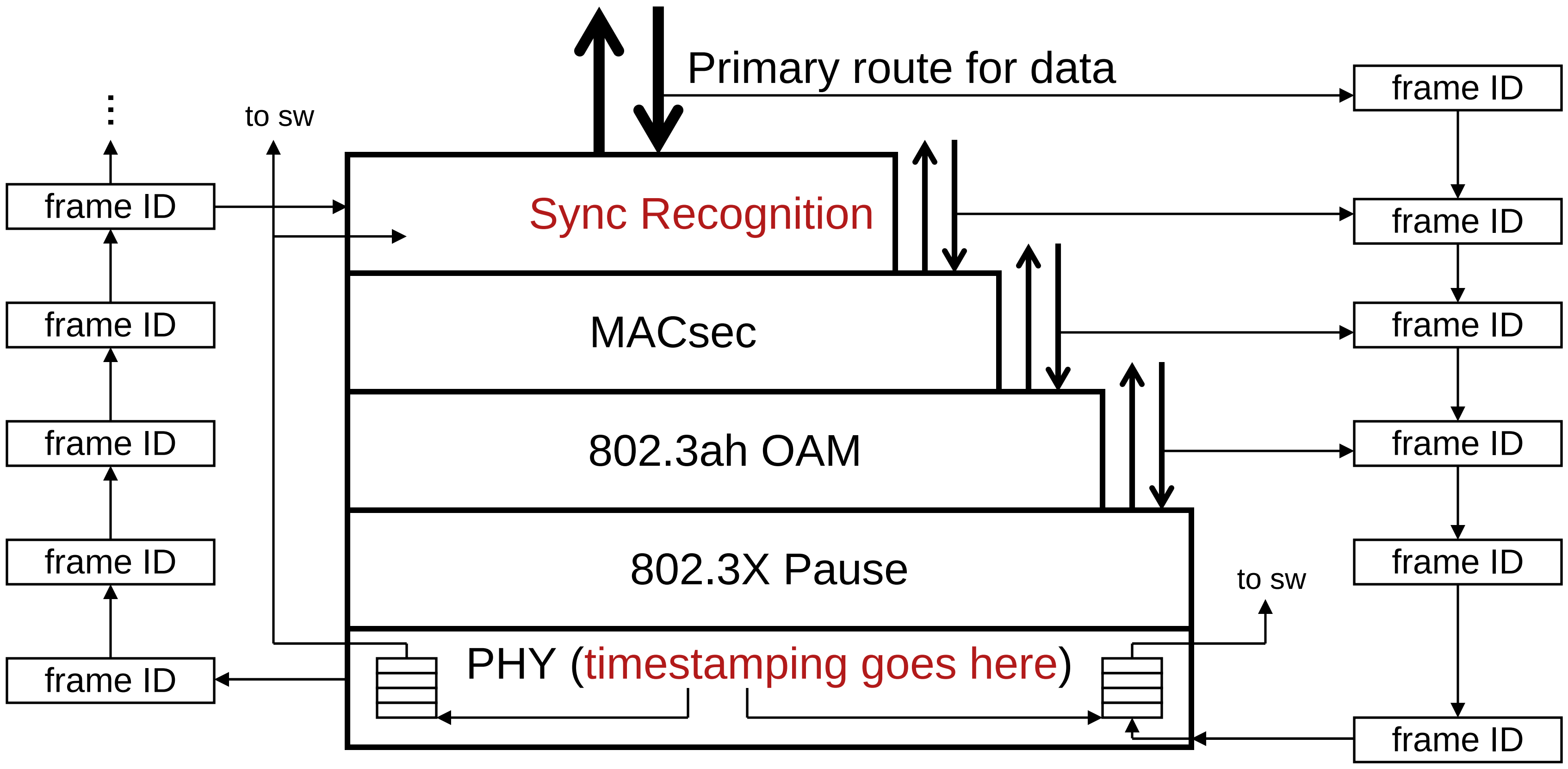
  A unique (to the timestamping mechanism) frame identification.

- For every frame marked for timestamping, the timestamping mechanism records the frame ID and the timestamp in a FIFO for recovery by the software.
- If there is only one source for syncs, then the frame ID can be simply a short sync serial number.
- If there are multiple sources (e.g. if .3ah OAM frames are to be timed) then the frame ID needs a source ID along with the serial number, to keep the frame IDs unique.

Sync processing: Better Solution

- A timestamp accompanies every received frame up the stack.
- If a layer does not care about timestamps, it can throw the timestamp away.
- At least one layer – the sync recognition layer – does care about timestamps.
- The sync recognition layer can place some kind of PDU identification (extracted from the sync frame) and a timestamp into a FIFO for recovery by higher layers.
- Alternatively, some kind of short Frame ID, instead of the timestamp, itself, can be passed up the stack. Then, the timestamp can be in a FIFO in the PHY.

Sync processing: Frame ID only



- Every received frame carries an ID up the stack.
- Every transmitted frame carries an ID down the stack.

Sync processing: Better Solution

- Frame Identifier, for both input and output, can consist of:
  - The ID of the (perhaps programmable) sync recognition register.
  - A serial number associated with that sync recognition register.
- That way, no part of the MAC needs to know how to extract information from the sync frame.
- The frame identifier can be short; it must be large enough to cover the frames in the stack, in intermediate layers, and perhaps, queues.

Implementing sync packets: the downside

Sync processing: Better Solution

- The frame ID is not needed by any function outside the stack.
- But, any function that is between the sync recognition module and the timestamp module must pass this information transparently through, and generate dummy frame IDs for frames it introduces into the flow.
- This is where the exact choice of method used to pass the frame ID is critical – ideally, a means can be found that will allow the frame ID to pass through the MACsec units already (or nearly) developed.

Why should 802.3 care?

Why should 802.3 care?

- At present, this author knows of 3 existing and in-process standards for protocols that insert timestamps into frames and require recalculation of checksums (not to mention a number of proprietary protocols):

  ITU-T Y.1731

  IEEE 1588

  IETF “TWAMP” project

- There are bound to be more, and very soon!
- MAC ASIC designers can expect a constant stream of requests for new insert-the-timestamp protocols.
- This is an untenable situation, especially for 10G and higher speeds.

Why should 802.3 care?

- If 802.3 defines a scheme for processing sync frames similar to the one herein described, then the designer of a system can provide a set of registers for identifying sync frames, and program them for various protocols that use a synch/followup technique.
- Remember that MAC sec makes it extremely difficult to insert timestamps, even with a highly programmable port ASIC.
- If 802.3 defines a sync recognition technique, protocol designers will be encouraged to use design around the sync/follow paradigm, instead of defining new insert-the-timestamp protocols.
- The world will then be a better place.

Summary

802.3 Sync processing

- All of the benefits of accurate timestamping can be had.

  Clock synchronization to nanosecond accuracy.

  One-way timing measurements.

  Accurate delay and jitter measurements.

- None of the drawbacks of accurate timestamping need be suffered.

  PHY level need not parse frames at all.

  No data insertion, removal, or CRC recalculation required at all.

  Hardware sync parsing can be done in a general way, easily extensible to future protocols that use the sync/followup technique.

  Timing measurements can be protected with MACsec, IP sec, or any other security protocol.

  ASIC do not have to be respun every time a new protocol is invented.

802.3 Sync processing

- But, there is some cost.

  All sub-layers between sync recognition and timestamping (e.g. MACsec) must pass a “frame ID” transparently.

  802.3 must define how the Frame ID and/or timestamp are passed up and down its internal sublayer stack.

  802.3 must define how the Frame ID and/or timestamp are passed through its interface to the higher layers.