Proposal for an Open Loop PHY Rate Control Mechanism

Shimon Muller
Ariel Hendel

Sun Microsystems Inc.
Computer Systems

May 23, 2000

Outline

- Introduction
  - Why is a Rate Control Mechanism Necessary for 802.3ae
- MAC<->PHY Rate Control Alternatives
- MAC Self-Pacing Proposal
  - Concept
  - Implementation Implications
  - Standard Implications
  - Issues
- Summary

Introduction --- Why Rate Control for 802.3ae?

- **At the November 1999 meeting, the HSSG adopted the following objectives for 802.3ae:**
  - **Support a speed of 10.0000 Gb/s at the MAC/PLS service interface**
  - **Define two families of PHYs:**
    - **A LAN PHY, operating at a data rate of 10.0000 Gb/s**
    - **A WAN PHY, operating at a data rate compatible with the payload rate of OC-192c/SDH VC-4-64c**
  - **Define a mechanism to adapt the MAC/PLS data rate to the data rate of the WAN PHY**

MAC<->PHY Rate Control Alternatives

- **Fine granularity rate control**
  - **Word-by-Word**
- **Packet granularity rate control**
  - **Carrier Sense based**
  - **Busy Idle**
  - **Self pacing in the MAC**

Rate Control Alternatives --- Word-by-Word

- **Concept**
  - **Adds a “hold” signal on the XGMII from the PHY to the MAC**
    - **Indicates to the MAC to stop transmission for one clock cycle**
    - **The MAC inserts “nulls” into the transmitted data stream**
- **Issues**
  - **Interrupts the flow of data through pipeline stages**
  - **Makes buffer pre-fetching difficult**
  - **Tricky timing**
  - **Doesn’t work with HARI**

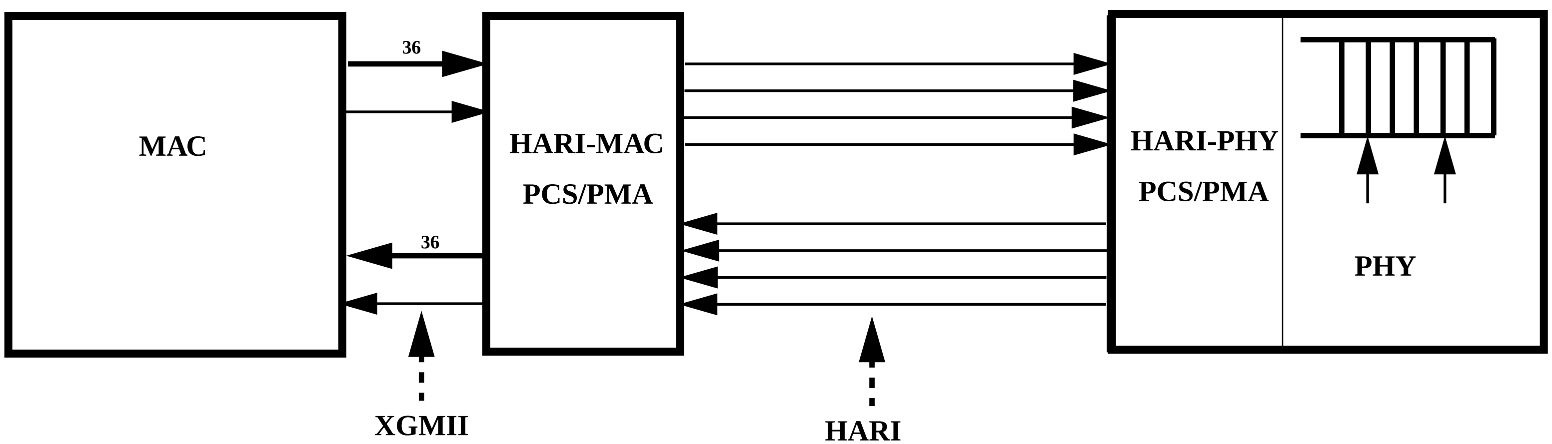
Unified LAN/WAN PHY Proposal --- H. Frazier, January 2000

  - **Fundamentally changes the operation of the current MAC algorithm**
    - **Packet processing is no longer continuous from start to end**
      - **Affects all the h/w processes in the MAC: State machine sequencing, CRC computation, Error detection, etc.**
    - ***The 10-GE MAC is no longer a scaled version of its lower speed siblings!***

Rate Control Alternatives --- Busy Idle

- **Concept**
  - **PHY sends "Busy Idle" to MAC during IPG**
    - **MAC pauses transmission at frame boundary**
  - **PHY sends "Normal Idle" to MAC during IPG**
    - **MAC resumes transmission**
  - **Need a ~256 byte FIFO in WAN PHY Tx path**

Unified LAN/WAN PHY Proposal --- H. Frazier, January 2000



Rate Control Alternatives --- Busy Idle (continued)

- Issues
  - Implementation: Too cumbersome for what it tries to achieve
    - Work required per packet transmission:
      - MAC transmits a frame
      - PHY monitors internal fifo occupancy
      - PHY fifo reaches high threshold
      - PHY waits until current frame reception is completed
      - HARI-PHY encodes Busy Idle on HARI
      - HARI-MAC decodes Busy Idle from HARI
      - HARI-MAC encodes Busy Idle on XGMII
      - MAC decodes Busy Idle from XGMII
      - MAC waits until current frame transmission is completed
      - MAC blocks next frame transmission
      - PHY fifo reaches low threshold
      - PHY waits until current frame reception is completed
      - HARI-PHY encodes Normal Idle on HARI
      - HARI-MAC decodes Normal Idle from HARI
      - HARI-MAC encodes Normal Idle on XGMII
      - MAC decodes Normal Idle from XGMII
      - MAC unblocks next frame transmission

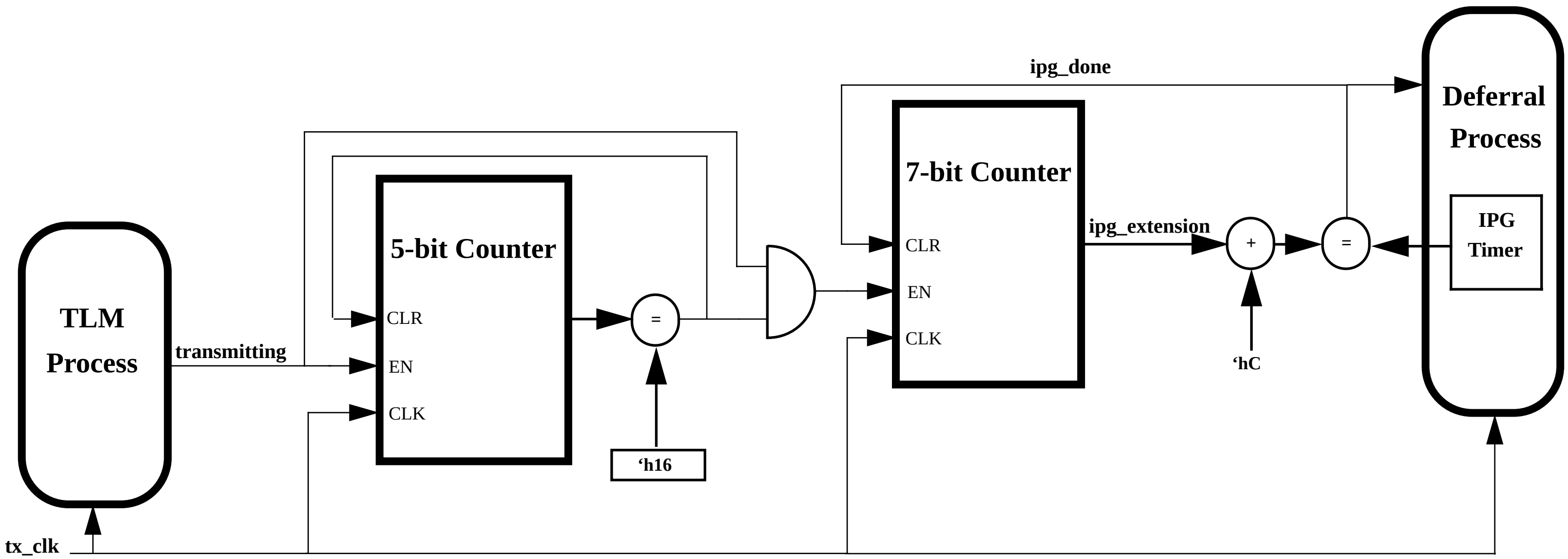
Rate Control Alternatives --- Busy Idle (continued)

- **Issues (Continued)**
  - **Standardization: Current MAC has no defined mechanism for halting frame transmission**
    - **1. Change the MAC**
      - **Change the Pascal code --- how?**
      - **Define a new service primitive between the RS and the MAC**
    - **2. Change the MAC Control sub-layer**
      - **Add an XON/XOFF flow control mechanism (in addition to Pause)**
      - **Define a new service primitive between the RS and the MAC Control (bypass the MAC)**

MAC Self-Pacing Proposal

- **Concept**
  - **The MAC "knows" that the PHY is slower and by how much**
  - **The MAC adapts its average data rate by extending the IPG after each frame transmission**
    - **This guarantees that the MAC never exceeds the average data rate in the PHY, with packet granularity**
  - **The IPG extension is "dynamic"**
    - **Depends on the size of the previously transmitted frame**
  - **The PHY is only required to sustain the transmission of one maximum size packet**
    - **Requires a rate adaptation fifo in the PHY of ~64 bytes (plus framer overhead)**

MAC Self-Pacing Proposal --- Implementation



Notes:

*** transmitting --- signal that frames the transmission of a frame in the MAC**

*** ipg_done --- signal that indicates the completion of IPG transmission**

MAC Self-Pacing Proposal --- Pascal Changes

- **New transmit state variables (4.2.7.2)**
  - ***const***
    ifsExtensionLimit = ...; {in octets; maximum allowable IPG extension}
  - ***var***
    paceMode: Boolean; {static variable}
    ifsExtensionCount: 1... {in bits; running counter, counts bits in a frame}
    ifsExtensionSize: 0... {in octets; running counter, counts IPG extension octets}
    ifsExtensionRatio: 1... {in bits; function of ifsExtensionLimit and maxFrameSize}

- **State variables initialization (4.2.7.5)**
  - *procedure* Initialize;
    begin
    ....
    <u>paceMode := ...;</u>
    while carrierSense *or* receiveDataValid *do* nothing
    end;

MAC Self-Pacing Proposal --- Pascal Changes (cont.)

- **Frame transmission (4.2.8)**
  - *procedure* StartTransmit;

```
begin
   currentTransmitBit := 1;
   lastTransmitBit := frameSize;
   transmitSucceeding := true;
   transmitting := true;
   lastHeaderBit := headerSize;
   ifsExtensionCount := headerSize
end;
```

MAC Self-Pacing Proposal --- Pascal Changes (cont.)

```
process BitTransmitter;
 begin
  cycle {outer loop}
   if transmitting then
    begin {inner loop}
     extendError := false;
     ifsExtensionSize := 0;
     PhysicalSignalEncap;
     while transmitting do
      begin
       if (currentTransmitBit > lastTransmitBit) then TransmitBit(extensionBit)
       else if extendError then TransmitBit(extensionErrorBit)
       else
        begin
         TransmitBit(outgoingFrame[currentTransmitBit]);
         ifsExtensionCount := ifsExtensionCount + 1;
         if ((ifsExtensionCount mod 8) = 0) then
          if ((ifsExtensionCount mod ifsExtensionRatio) = 0) then
                                        ifsExtensionSize := ifsExtensionSize + 1;
        end
       if newCollision then StartJam else NextBit
      end
     if (ifsExtensionCount > 0) then ifsExtensionSize := ifsExtensionSize + 1;
     if bursting then
      begin
       InterFrameSignal;
       if extendError then
        if transmitting then transmitting := false
        else IncLargeCounter(lateCollision);
        bursting := bursting and (frameWaiting or transmitting)
      end
    end {inner loop}
  end {outer loop}
 end; {BitTransmitter}
```

13

MAC Self-Pacing Proposal --- Pascal Changes (cont.)

- *process* Deference;

```
begin
  if halfDuplex then cycle {half duplex loop}
    ..........................................................
    ..........................................................
  end {half duplex loop}
  else cycle {full duplex loop}
    while not transmitting do nothing;
    deferring := true;
    while transmitting do nothing;
    StartRealTimeDelay;
    while RealTimeDelay(interFrameSpacing) do nothing;
    while (ifsExtensionSize > 0) and paceMode do
      begin
        Wait (8);
        ifsExtensionSize := ifsExtensionSize - 1
      end
    deferring := false;
  end {full duplex loop}
end; {Deference}
```

MAC Self-Pacing Proposal --- Issues

- Dynamic IPG Extension as proposed is imprecise
  - After each frame transmission, the extension value is rounded up to an integral number of octets
    - Can potentially add transmission overhead by sending out a longer IPG than otherwise required
- However:
  - The XGXS striping scheme already forces an even greater overhead on most frame sizes
    - Requires alignment of the first byte of a frame to lane 0
    - Worst case overhead is 4.62% (for 65-byte frames)
  - The XGXS striping overhead absorbs the "extension overhead" for >75% of frame sizes
    - Also, absorbs the overhead for all "popular" frame sizes (64, 1132, 1518, etc.)
- *If desired, a simple enhancement to the proposal will eliminate the overhead for the remaining <25% of frame sizes*

Summary

- Both Busy Idle and Self-Pacing achieve rate adaptation by extending the IPG between frames
  - Busy Idle provides a closed-loop rate control mechanism between the MAC and the PHY
  - Self-Pacing relies on the fact that the data rates of the MAC and the PHY are fixed and known at initialization time, which eliminates the need for a closed loop
- Advantages of the Open-loop Rate Control
  - Much simpler
  - Somewhat cheaper
  - Independent of the PHY and the interconnect between MAC and PHY
  - The self contained nature of this mechanism provides a more robust solution