Flow Control For Gigabit Ethernet

Moshe De-Leon
Digital Equipment Corporation (Digital)

IEEE 802.3z Task Force
9-July-1996

Scope

- The presentation discusses various issues related to flow control for Gigabit Ethernet
- XOFF(time) and the PHY based signaling suggested in the buffered repeater proposal are reviewed
- Two advanced (frame based) flow control schemes are then presented:
  - An absolute credit based scheme (as presented in January, modified for Gigabit Ethernet)
  - Simple (we hope -) rate based scheme
- We conclude with some suggestions regarding future work that is needed

– ... as well as some future/present work which is <u>not</u> needed

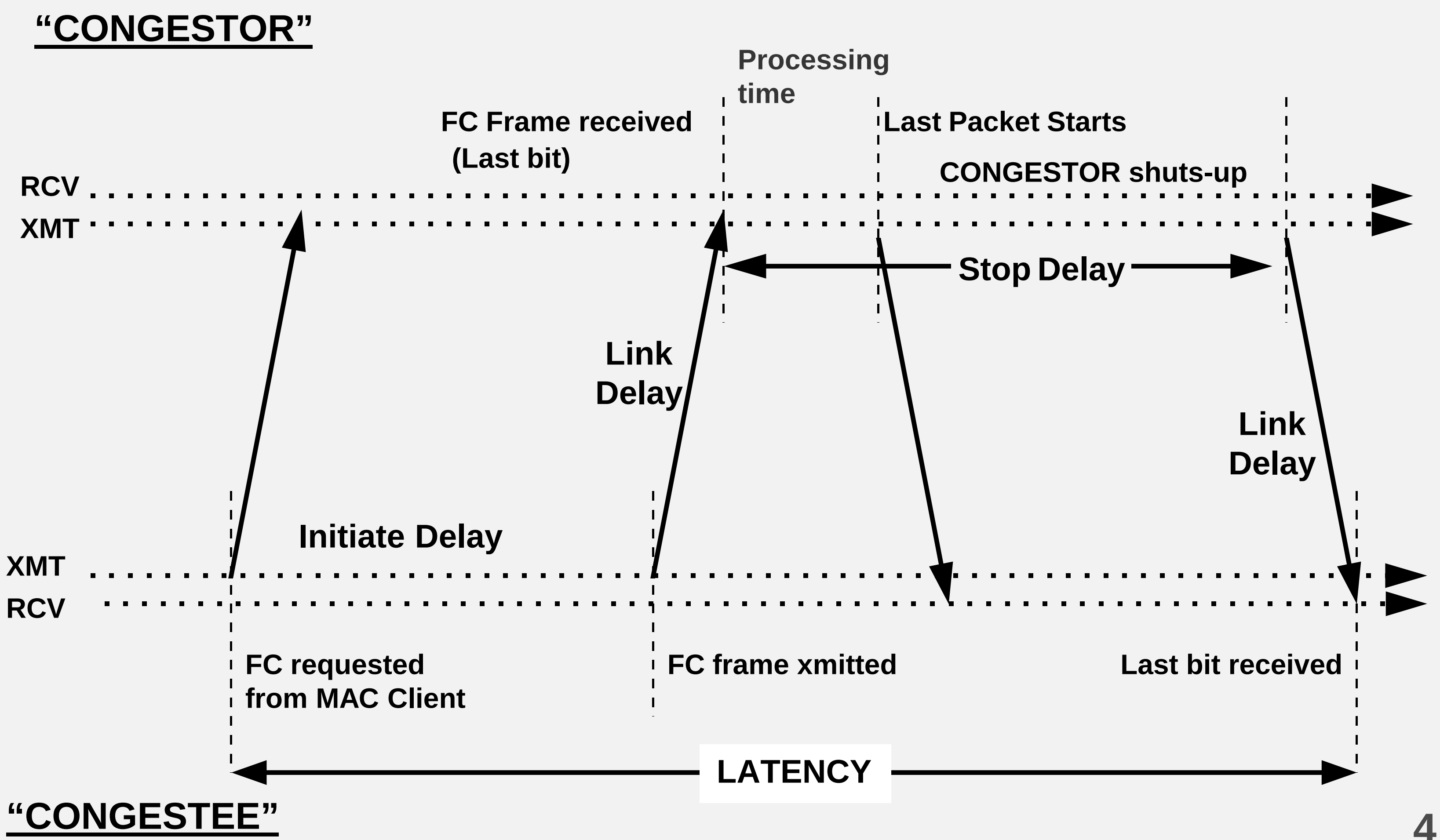
XOFF(t) Scheme

- Concept:
  - – An XOFF(t) keeps the link partner quiet for t*slot_time
  - – An XOFF(t2) received after an XOFF(t1) overwrites the previous value
    - Thus an XOFF(0) means XON
- This scheme ***IS*** the choice made by 802.3x for flow control in full duplex links
  - – There was an alternative PHY based scheme
  - – It was decided to adopt the frame based scheme
    - ... for many good reasons (should we repeat them here ?)
  - – The buffered repeater proposal stated that things are different for Gigabit Ethernet

Well ... are they ?

Latency Model (Frame Based)

Digital Equipment Corporation (Digital)



Latency Elements (Frame Based)

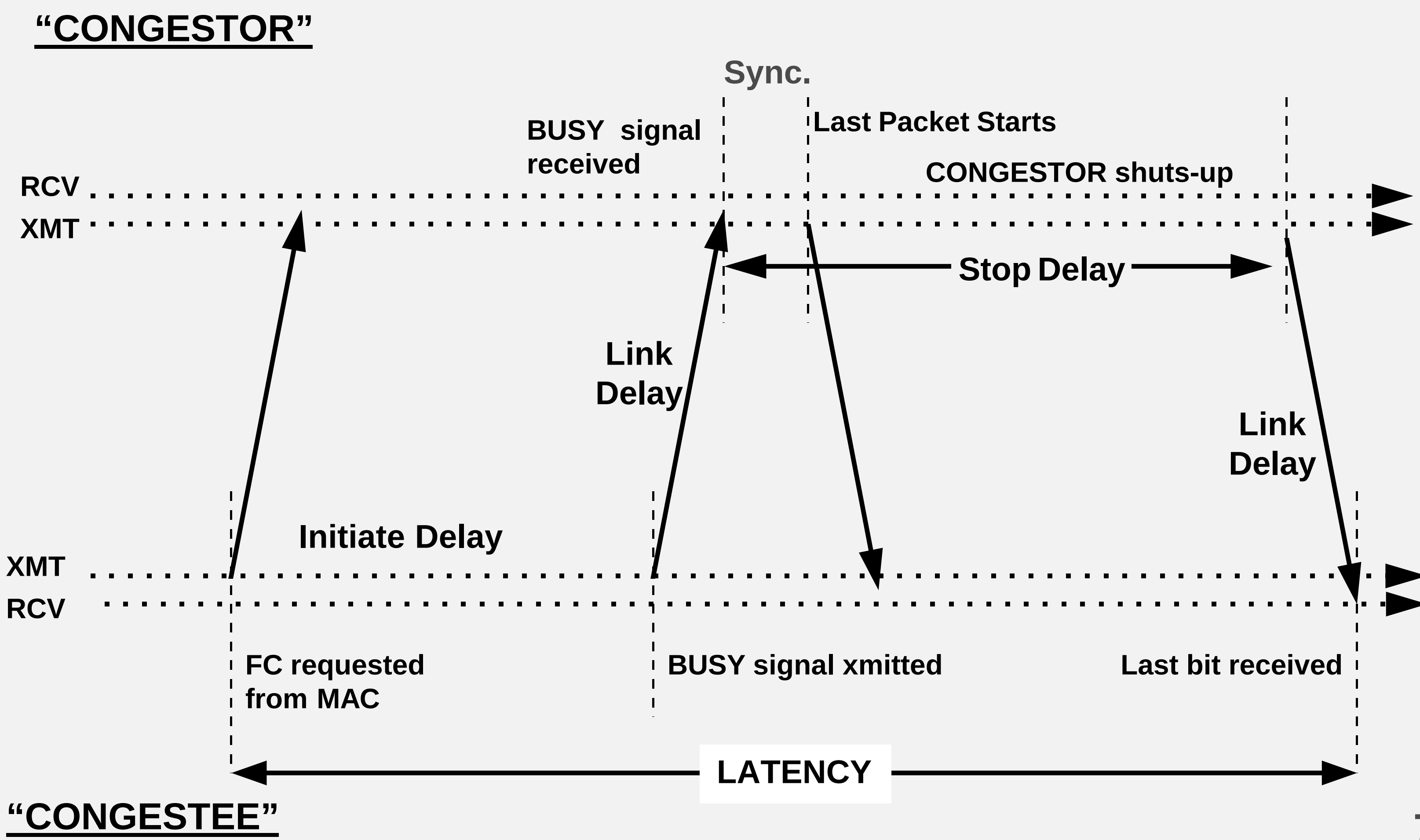
$$L_{FC} = \text{Initiate}_{Dly} + (2)\ \text{Link}_{Dly} + \text{Msg}_{Len} + \text{Stop}_{Dly}$$

- Initiate_{Dly} = Delay before FC message can be transmitted
- Link_{Dly} = Propagation Delay (1st bit) from MII to MII
- Msg_{Len} = Length of FC message (64 bytes + Preamble)
- Stop_{Dly} = Max number of bits transmitted after receiving last bit of FC frame

Latency Results (Frame Based)

	Bytes
◆ $Initiate_{Dly}$ = 1 max length packet + IPG =	1530
◆ $Link_{Dly}$ = XMT + Link (2Km) + RCV = 3 + 1250 + 3 =	1256
◆ Msg_{Len} = Min length packet =	72
◆ $Stop_{Dly}$ = 512 b - IPG + 1 max length packet =	1570
◆ $Link_{Dly}$ =	1256
◆ $Latency_{FC}$ =	5684

Latency Model (PHY Based)



Latency Elements (PHY Based)

$$L_{FC} = Initiate_{Dly} + (2)\ Link_{Dly} + Sync._{Dly} + Stop_{Dly}$$

- $Initiate_{Dly}$ = Delay before Busy signal can be transmitted
- $Link_{Dly}$ = Propagation Delay from MII to MII
- $Sync_{Dly}$ = MAC detecting CRS over the MII
- $Stop_{Dly}$ = Max number of bits transmitted after receiving Busy signal

Latency Results (PHY Based)

	Bytes
◆ $Initiate_{Dly}$ = 1 max length packet =	1518
◆ $Link_{Dly}$ = XMT + Link (2Km) + RCV = 3 + 1250 + 3 =	1256
◆ $Stop_{Dly}$ = 1 max length packet =	1518
◆ $Sync._{Dly}$ = similar to 100Base-T =	2
◆ $Link_{Dly}$ =	1256
◆ $Latency_{FC}$ =	5550

Summary: 5550 bytes delay for PHY Based Scheme
5684 bytes delay for Frame Based Scheme
Difference: 134 extra bytes of buffering

.... doesn't look too bad -)

Overhead Computations

- Let's take a look at the ***(unreasonable)*** worst case scenario (not our idea ... but):
  - 8K bytes buffering per port in the switch (oops ... we meant buffered repeater)
  - High watermark - low watermark = 1.5K bytes
  - As soon as we reach the low watermark the following pattern repeats ***endlessly:***
    - Congestor sends 1518 bytes packet and stops ***exactly*** then
    - Congestee ***(immediately)*** sends an XOFF (64bytes + ipg)
    - Congestee ***(immediately)*** empties an 1.518 bytes packet to reach low watermark
    - Congestee ***(immediately)*** sends an XON (64bytes + ipg)
- ===> Overhead will be (76+76)/(1518+1518) = 5%

Overhead Conclusion

- This *(unreasonably)* assumes a worst case condition for a long period of time
  - Congestor pattern is not likely to repeat too often
  - Congestee is extremely unlucky
  - All these *(immediately)*'s are not too likely to ever happen
- It ignores the *t* in the XOFF(*t*) command
- Given a more practical buffer size of 16K bytes per port (not too expensive for Gigabit ...):
  - High watermark - low water mark = 9.5K bytes
  - Overhead (yes, in this ... scenario) is less than 0.8%
- ===> In ALL *practical* scenarios - overhead is ***negligible***

The Issue of Robustness

Digital Equipment Corporation (Digital)

- Given a bit error rate of 10**(-12)
  - The chances of a corrupted XOFF(t) frame are:
    - 1 - ((1- (10**(-12)))**512) = 0.000000000512
    - One way to look at it - one flow control frame out of every 1953125000 flow control frames is corrupted
    - Assuming the worst case scenario of the overhead computation above, an XOFF(***t***) flow control frame is transmitted every 1518+1518+76 +76 bytes or every 25504 bits ... or 39210 frames per second
      - **==> we will have an XOFF dropped every 1953125000/39210 seconds = 49812 seconds = 830 minutes = every ~14 hours**
      - **This means ... 2 to 5 packets are dropped (until the "congestee" sends another XOFF)**
      - **Many more DATA frames will be lost simply due to ... bit error rate**

We "think" we can live with 2-5 packets being dropped every 14 hours in this unreasonable worst case scenario -)

Other Observations

- Assuming PHY is full duplex - why stop when receiving or for that matter - why look at CRS at all ?
  - ===> The suggested buffered repeater scheme introduces (unnecessary) 50% overhead
  - ... But if this is full duplex, then this is no longer a (buffered) repeater (I guess...)
- And there are "other" issues:
  - Why have ***three*** types of MAC ?
  - Why have PHY ***and*** Frame based flow control ? One of our (802.3x) goals was to have one flow control for all technologies and all speeds
  - Why specify the architecture and/or behavior of a ***switch*** ?
  - Why does 802.3 need to discuss ***products*** ?
  - Why make the PHY more complex (even slightly) ?
  - What does this ***"thing"*** buy us if "it does not replace traditional CSMA/CD ?"

On Idle/Busy PHY Signals

- It can be claimed about Idle/Busy:
  - "Let's just define these, maybe someone will use them some day"
- Problem:
  - As soon as we define them we have to specify their semantics
  - As soon as we specify their semantics we have a new flow control scheme

Do we really need two flow control schemes for Ethernet ?

When the benefit is so small (i.e. less than 134 bytes of buffering)

An Alternative Buffered Repeater Proposal

Digital Equipment Corporation (Digital)

- Buffered repeater is a ***black box*** whose architecture is NOT specified by 802.3

A low-cost, high-performance buffered repeater is:

Trivial to build using:

1. **802.3x frame based flow control**
2. **FDX MAC as defined by 802.3x**

No need to define a new type of MAC
No need for defining (and giving semantics) to idle/busy

Why spend so much time and energy and add complexity and definitions and semantics for something that *might* buy us 134 bytes of buffering in worst case conditions?

... In short, we do not need an alternative PHY based flow control for Gigabit Ethernet

Buffered Repeater Conclusion

====> There is full duplex Ethernet and there is CSMA/CD Ethernet:

1. Buffered repeater as proposed so far does not fit any of these branches:

a. For full duplex we simply do not need it (it adds nothing except additional complexity)

b. For half duplex (i.e. CSMA/CD) it is not relevant at all

2. Buffered repeater for the full duplex branch can be designed without any aditional work from 802.3z based on 802.3x work

====> Instead of taking more of 802.3z's valuable time to discuss switch architecture and Phy based flow control schemes which buy us (at most) 134 bytes of buffering -

We shall invest the time in looking at

1. More advanced flow control schemes (sub task force ?)

2. Solutions to the CSMA/CD problems (performance, capture effect, etc... another/same sub task force ?)

Credit Based Flow Control Concept

- ***Absolute*** credits are sent:
  - A credit(N Bytes) means: “You can send me N bytes”
- Transmit side is trivial:
  - After transmission is started it shall never be stopped
  - A new transmission can start if
    - I have credits to send at least 1500 bytes
    - I have credits to send N bytes & I have a frame with a (known) length <= N
  - When a frame carrying credits is received:
    - Simply write the new value in the credit counter
    - No need to synchronize
    - Value is *guaranteed* to be ***accurate*** when it is received

Credit Based Flow Control Concept (cont.)

- Receive side is very simple:
  - Assuming there are N free bytes
  - When there is a ***need*** (see below) or when there is ***nothing better to do*** (see below):
    - send a credit frame with (N - Const) bytes
  - Const = Sending time + RTD + Processing time (more details to follow)
- There is a ***need*** to send credit when:
  - N - constant > some threshold ***AND***
  - Link partner is short with credits ***OR*** we didn't send a credit frame for the last TBD time (order of seconds)
- There is nothing better to do means:
  - No transmit frames ***OR*** no available credits

Credit Based Flow Control - Constant Part Calculations

- Partner will get a new credit value sent now (time is ***t***) in time ***t*** + ***DT*** where ***DT*** is the sum of:
  - ***S***: The time it took the last byte of the frame to cross MDI (sending end)
  - ***HRTD***: The time it took the last byte of the frame to cross the MAC Control Interface (receiving end)
  - ***R***: Processing time (receiving end)
- ***S*** is ~ 64 bytes time, ***R*** is negligible and ***HRTD*** is either
  - Bounded in 2KM Gigabit links (1250 bytes ?)
  - Can be computed in future extensions
- Bytes in flight: bytes that were sent before time ***t*** but did not reach receiving end (In Gigabit links < 1250 bytes ?)
- ==> Value to send == # of free bytes - ***C*** (constant)
  - Where ***C*** == (***DT*** * line rate in bytes) + bytes in flight

Credit Based Flow Control - Intermediate Summary

- Basically this is it:
  - – No frame is sent before receiver is ready for it
    - The *no frame lost* is guaranteed
    - No need to act in real time to prevent frame loss
  - – No need for ***ANY*** re-synchronization due to frame lost
    - New credit frames turn old/lost frames obsolete
    - Sending a credit after a long (seconds) gap from the last credit sent is all that is needed to deal with lost credits
- It is important to note that the standard needs ***only*** to specify the syntax of the credit frame:
  - – One new opcode is needed
  - – ***Only*** the behavior of the transmitting end needs to be specified

Credit Based Flow Control - Pseudo Code

- The next part of the of presentation will provide the pseudo code
  - We believe it might help:
    - To understand this scheme better
    - To show how simple it is
    - To catch bugs and uncovered details
- Most of the code:
  - May be implemented by the MAC client
  - May be implemented by the MAC control sub-layer
- In order to facilitate this approach all we ***have*** to do is:
  - Specify the needed opcode
  - State that the transmitter is not allowed to send frames if it doesn't have credit to send them

Credit Based Flow Control - Constants

C: integer; {the constant, whose computation was presented before, to be subtracted from free_bytes when sending credits}

CRD_MAX_GAP: integer; {max time which is allowed between credit frames}

SND_CRD_MIN: integer; {minimal number of credits to send}

PRIO_THRSLD: integer; {when LP has less credit than this number, we set high the priority of returning credits}

Note: *ALL these constants might be implemented by the MAC client*

Credit Based Flow Control - Variables

byte_count: integer; {number of bytes we can send}

LP_byte_count: integer; {how many bytes link partner can still send us}

fctl_priority_flag: Boolean; {when set we need to send credit as soon as possible}

free_bytes: integer; {how many free bytes do we have being updated by the MAC client}

crd_time_sent: integer; {the last time we sent a credit frame}

Note: *ALL these variables except for **byte_count** might be implemented by the MAC client*

Credit Based Flow Control - Transmitter Process

Transmits credit or normal frames according to availability of credits and the priority of sending credits

```
Loop forever
{
   If (fctl_priority_flag && (free_bytes > (SND_CRD_MIN + C))) then
            transmit_fctl(free_bytes - C);
   else if (there is a frame to transmit with length < byte_count)
   {
            start a new transmission;
            while transmitting decrement byte_count every byte sent;
    }
    else if (free_bytes > (SND_CRD_MIN + C)) than
            transmit_fctl(free_bytes - C);
}
```

Note: only the part within the green box need to be implemented by the MAC Control sub-layer

Credit Based Flow Control - FCTL Processes & Procedures


```
Procedure Transmit_fctl(value);
{
   send a flow control frame with number of credits equal value;
   LP_byte_count := value;
   crd_time_sent := NOW;  fctl_priority_flag := FALSE;
}
Process Fctl_prio
Loop forever
{
  every byte received:
              LP_byte_count := LP_byte_count - 1;
  If ((LP_byte_count < PRIO_THRSLD) ||
     (NOW - crd_time_sent) > CRD_MAX_GAP)
  than fctl_priority_flag := TRUE;
}
```

Note: only the part within the green box need to be implemented by the MAC Control sub-layer

Credit Based Flow Control - Summary

- A simple (we hope) credit based scheme was presented
- The main features it provides:
  - No frame loss
  - No need for real time response
  - Suits high end and low end implementations (with respect to buffering)
  - Applicable to ***switch to switch*** and ***switch to end node*** links
- The main feature to emphasize:

Can work even with as little buffering as 4K bytes for 2KM Gigabit connections

Simple Rate Based Scheme - Introduction

- A word of caution:
  - A rate based scheme can turn into a ***nightmare*** with respect to both implementation and analysis
  - ... maybe this explains why a rate based scheme was adopted by ***"another"*** technology -)
    - ==== > the scheme we have in mind is very (too ?) simple
- The main advantages of rate based schemes:
  - Do not produce the "all or none" behavior
  - System behavior is smoother
  - Involve very little computation for ports
  - Are (somewhat?) less affected by the length of the links
- Main disadvantage: frames may be lost

Rate Based Scheme - Concept

- Basic idea - use IPG to control the rate
- Three commands in addition to the XOFF(*t*):
  - Set_ipg(*n*): ipg is **set** to *n*
  - Slow_down(*k*): ipg is **multiplied** by 2***k*
  - Speed_up(*l*): ipg is **divided** by 2***l* (down to current (*minimal)* ipg)
- Implementation is ... trivial:
  - Instead of using a constant IPG use a programmable value which may be affected by the flow control commands
- An acceptable simplification:
  - Use only the Set_ipg(*n*) command

Rate Based Scheme - Usage

- Not as trivial as the XOFF(t) or credit based schemes
  - However this is ***NOT*** the scope of 802.3
    - ===> All we have to do is provide the hooks and be convinced that they are useful
- It does ***NOT*** guarantee zero frame loss
- Simplest usage:
  - Define a series of high watermarks H1...Hn
    - Whenever we exceed a watermark we send Slow_down(1)
  - Define a series of low watermark L1 ... Lk
    - Whenever we go under a watermark we send Speed_up(1)
  - Whenever there is a major change we may use
    - XOFF command
    - Set_ipg command to set ipg to a suitable value immediately

Rate Based Scheme - Evaluation

- Advantages:
  - Does not produce the “all or none” behavior
  - Goes well with the XOFF(t) scheme
  - Simpler to implement than credit based scheme
  - Less (directly) affected by RTD
  - Goes well with flow control of “other technologies”
  - Provides more room for clever system design (i.e., is more flexible) than other schemes
  - Better suits Ethernet “philosophy”
- Disadvantages:
  - ***Does not guarantee zero frame loss***
  - Usage is less straightforward than in credit based or XOFF based schemes
  - More difficult to analyze

Summary

- Frame based XOFF flow control scheme was reviewed:
  - No (new) advantages to PHY based flow control
    - We really do not need the PHY vs. Frame based flow control debate *again* in 802.3z
      - **It might have a significant impact on schedule and will raise issues of compatibility with other pars**
      - **... And there isn't any advantage in Phy based signaling over the chosen frame based signaling in the first place**
- Buffered repeater concept was reviewed
  - *"Some"* concerns were raised
  - An alternative architecture was provided
    - One not involving ***any more work*** on 802.3z's part
- Two alternative flow control schemes for Gigabit Ethernet were reviewed
  - More work is needed