

# Computer Communication Networks CSI 416/516

## Lecture 9: Peer to Peer Networking

Stephen F. Bush

GE Global Research

October 27, 2009

## Lecture Outline

- 1 Introduction
- 2 ACK/NACK
- 3 ARQ Protocols
  - Stop-and-Wait ARQ
  - Go-Back-N ARQ
  - In-Class Exercise
- 4 Selective Repeat ARQ
  - Selective-Repeat ARQ Example
- 5 ARQ vs. Flow Control
  - Flow Control Network Taxonomy
  - Hop-By-Hop vs. End-to-End
  - DECbit flow control
- 6 TCP Flow Control
  - TCP RED

## Computer Communication Networks CSI 416/516

**Instructor:** Dr. Stephen F. Bush, GE Global Research  
**Phone:** 387-6827  
**Email:** bushsf@research.ge.com  
**Office Hours:** Tue 10:05pm HU-132 and by appointment  
**Course Website:** <http://www.cs.albany.edu/~bushsf>

# Syllabus

Week	Topics	Readings	Notes
1	Intro to Networking Layered Architectures	L1 B1 N1 L2 B2.1-2.2 N4	
2	Layering Sockets	L2 B2.3	HW 1 Assigned
3	Information Theory Compression & Correction	N6 L3.9	HW 1 due HW 2 Assigned
4	Networks The Physical Layer	L7 L3, L12.1-12.3, N2	
5	Multiplexing and Switching The Telephone Network	L7.3 L4	
6	Active Networks The Active Network Framework	B3	<b>Proposals Due</b>
7	Mid Term Exam Covers Weeks 1-6		

- N Nanonetworks, Bush, S. F., Artech House, 2010.
- B Active Networks and Active Network Management: A Proactive Management Framework by Stephen F. Bush and Amit Kulkarni, Kluwer Academic/Plenum Publishers, New York, Boston, Dordrecht, London, Moscow, 2001, 196 pp. Hardbound, ISBN 0-306-46560-4.
- L Communications Networks: Fundamental Concepts and Key Architectures, Leon-Garcia and Widjaja, McGraw Hill, 2003.

# Syllabus

Week	Topics	Readings	Notes
8	Queuing Theory	L App A	HW2 Due
9	Network Simulation NS-2	N7	HW3 Assigned
10	Peer to Peer ARQ and Flow Control	L5, N3	
11	Multiple Access Ethernet	L6.1-6.2 L6.6-6.11	HW3 Due HW4 Assigned
12	Scheduling QoS	L6.3-6.4	
13	Routing Ad Hoc Networking	L7	HW4 Due (optional)
14	Sensor Networks	N2 N3	HW5 Assigned
15	Student Presentation Student Presentations	L App B B4-6	
16	Network Management		HW5 Due
	Final Exam and Vacation!		

- N Nanonetworks, Bush, S. F., Artech House, 2010.
- B Active Networks and Active Network Management: A Proactive Management Framework by Stephen F. Bush and Amit Kulkarni, Kluwer Academic/Plenum Publishers, New York, Boston, Dordrecht, London, Moscow, 2001, 196 pp. Hardbound, ISBN 0-306-46560-4.
- L Communications Networks: Fundamental Concepts and Key Architectures, Leon-Garcia and Widjaja, McGraw Hill, 2003.

# Scribe Schedule

Week	Scribe 1	Scribe 2	Scribe 3
1 (Intro)	Caldara, Logan Nathan	Cagan, Ferhat	
2 (Layering)	Crisafulli, Anthony Nicholas	Erbatur, Serdar	
3 (Info Theory)	Krishnaraj Ravindranathan		
4 (Phy Layer)	Kudlack, Edward A		
5 (Switching)	Zhang, Xing		
6 (Active Networks)	Krishnaraj Ravindranathan		
7 (Midterm)			
8 (Queuing Theory)	Caldara, Logan Nathan		
9 (ns-2)	Cagan, Ferhat		
10 (ARQ)	Crisafulli, Anthony Nicholas		
11 (Multiple Access)	Erbatur, Serdar		
12 (Sched+Routing)	Zhang, Xing		
13 (Proj Pres)	Subramaniam, Sathiyaseelan		
14 (Routing)	Caldara, Logan Nathan		

## Exam schedules

### Class schedule

- Tue Nov 17 – no class; use the time to work on your final projects

Nov 3	ARQ protocols
Nov 10	Multiple Access
Nov 17	No Class
Nov 24	Scheduling
Dec 1	Routing
Dec 8	Classes have ended
Dec 15	Final Exam

- Unless notified otherwise, all exams will be held in the same room and time room as regular classes
- Final: Tuesday 15-Dec 8:00pm – 10:00pm

## Homework Problems...

Topic	Assignment	Due Date
Nano Networks	Ch. 4 Ex. 25 & 28	Nov 3

Final project is most important component of your grade

Component	Weight	Total Grade
Final Project	30%	30%
Midterm/final	20% each	70%
Assignments (HW)	10%	80%
1 S/W Project	10%	90%
Scribe Notes	5%	95%
i>Clicker Responses	5%	100%

## Class Note Grades

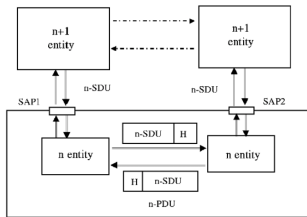
- Class notes are now on the class website
- Please review the website to see examples of good notes
- Grading for class notes:

Condition of my notes	My grade
I failed to submit my notes on time	0%
I cannot process with L <sup>A</sup> T <sub>E</sub> X	0%
My notes exhibit poor content/grammar	50%
I was lazy and copied from the lecture slides	50%
My notes were selected to post on website	100+%

## Some Review

Recall that:

- Peers are interacting entities at the same level of abstraction
- Peers communicate via protocol data unit (PDU)
- The service model of a protocol describes its interface (SDU) to higher layers

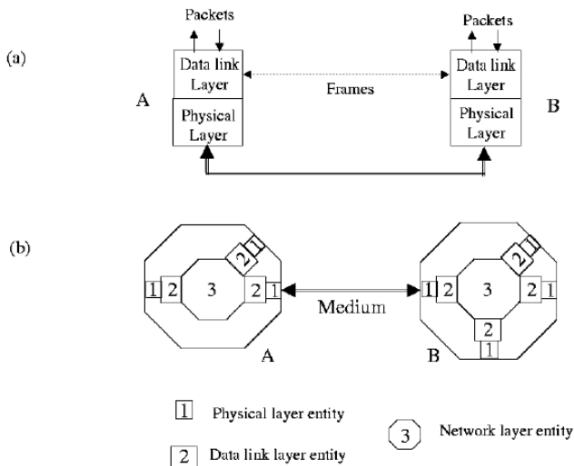


## Flavors of services

We can categorize services according to:

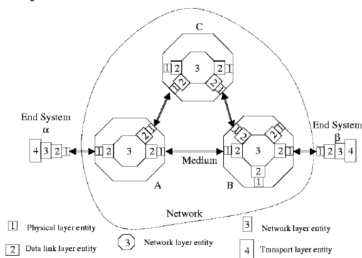
- Guarantees made
  - Confirm delivery of packets
  - Ensure packets arrive in order
  - Limit delay or jitter (QoS)
- The setting where they operate:
  - Link level protocols
  - End-to-end protocols

# Flavors of services



## End-to-end systems and network structure

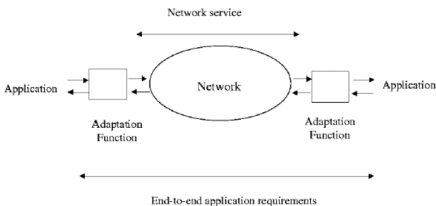
- The networks supporting end-to-end services are often dynamic in:
  - Connectivity
  - Congestion
- Layering permits designers to cope with these complexities



# Adaptation functions

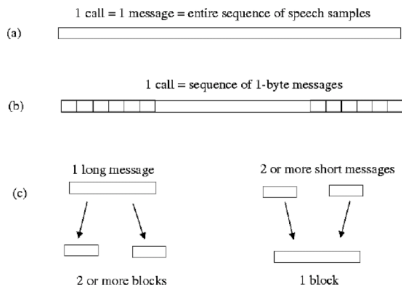
End-to-end systems use adaptation functions to deal with:

- Arbitrary message size
- Reliability and Sequencing – ACK/NAK
- Pacing and Flow Control – Sequence Numbers and Time Stamps
- Timing – MUX/DeMUX
- Addressing
- Security – privacy, integrity and authentication



## Handling arbitrary message sizes

- There are two cases:
  - Big PDUs and small SDUs – segment and reassemble
  - Many small PDUs and small SDUs – block and unblock
  - Basically the SDU is the data that a certain layer will pass to the layer below
    - It differs from the PDU in that the PDU specifies the data that will be sent to the peer protocol layer at the receiving end

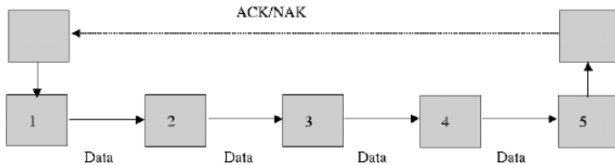


# ACK/NAK

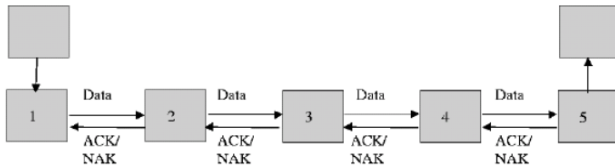
- To notify a sender of whether sent data was delivered successfully a receiver can send a:
  - Acknowledgment (ACK) – Indicates correct receipt
  - Negative Acknowledgment (NAK) – Complain about delivery errors
- Note: ACK/NAK strategies require time-outs. For systems using reliable links, end-to-end (vs. hop-by-hop) style is preferred

# ACK/NAK

End-to-end



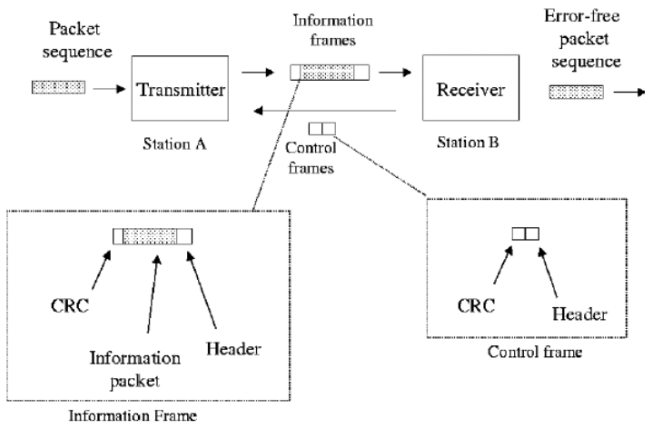
Hop-by-hop



## ARQ Protocols

- Automatic Repeat Request (ARQ) protocols ensure reliable delivery of data (or notify sender of inability to deliver)
- Frame – Groups of bits transmitted together, consists of:
  - Header – Describes how to treat the rest of the frame
  - Information Packet (or Payload) – user data (optional)
  - Error Detection Code – frequently a CRC
- Information frames (I-Frame) – Contain data
- Control frames – Regulate communication

# ARQ Protocols



## Types of Control Frames

Control Frames can be:

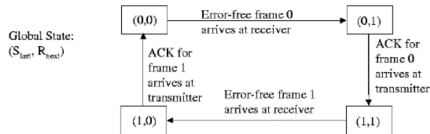
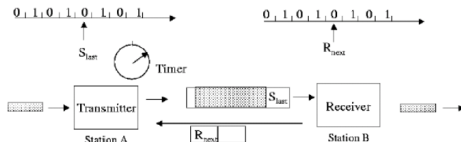
- From the receiver:
  - ACK – Got the data
  - NAK – Did not get the data
- From the sender:
  - Enquiry Frames (ENQ) – Request a status report from receiver

## Stop-and-Wait ARQ

- Stop-and-Wait ARQs work by:
  - Receiver Side gets an I-Frame and
    - CRC valid (no error detected) – reply ACK
    - CRC invalid – reply NAK
- Sender Side:
  - Transmits an I-Frame
  - Immediately starts an I-Frame timer
- Does one of the following:
  - Gets an ACK before timeout – Sends the next frame
  - Gets a NAK before timeout – Retransmits frame
  - Times out – Retransmits frame

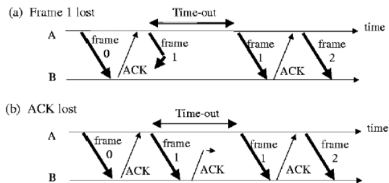
## Stop-and-Wait ARQ challenges

- ACKs and I-Frames need to avoid:
  - ACK frame mismatch – ACKs and I-Frames should contain sequence numbers
  - Incorrect timers – I-frame timers must not be too fast (too slow is bad too!)



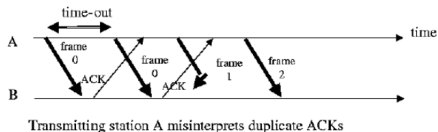
## Efficiency of Stop-And-Wait ARQ

- Stop-And-Wait ARQ's efficiency can be improved by:
  - Checkpointing – When timing out after sending a long I-frame, instead of retransmitting, send an ENQ first (a relatively minor modification)
  - Pipelining – When propagation delay is large relative to frame transmission time (Requires a new protocol – discussed next)



In parts (a) and (b) transmitting station A acts the same way, but part (b) receiving station B accepts frame 1 twice.

## Efficiency of Stop-And-Wait ARQ

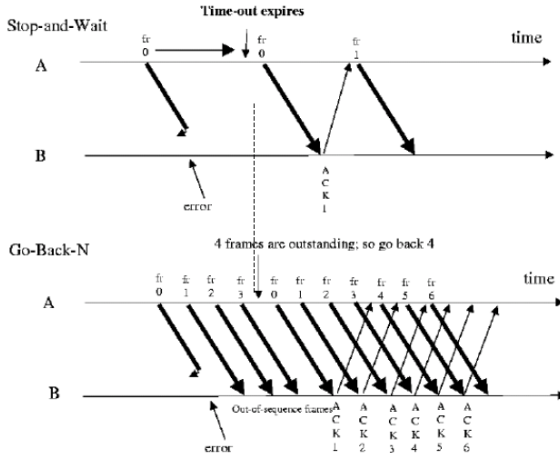


(when ACKs are unnumbered)

## Go-Back-N ARQ

- Go-Back-N ARQ refers to **pipelined** ARQ with pipeline depth  $N$ :
  - The sender:
    - Remembers the previous  $N$  I-Frames
    - Upon receiving a NAK (or timing out) on I-frame  $i$ , restart transmission at I-Frame  $i$
  - The receiver:
    - Discards out-of-sequence and duplicate packets
    - Sends ACK/NAK much the same as Stop-And-Wait ARQ

# Go-Back-N ARQ vs. Stop and Wait ARQ



## In-class exercise

### Exercise

Suppose that a 10 Mbps channel has 1250 byte packets sent with a 50 msec round trip delay from beginning of transmission to receipt of the reply. You can treat header, CRC, and control frame bandwidth consumption as negligible.

- 1 What is the effective bandwidth (in packets per second) of the Stop-And-Wait ARQ?
- 2 How many packets could have been transmitted during the round trip delay if we didn't wait for an acknowledgment?

## Solution

- The effective bandwidth is the amount of bandwidth the user sees:

$$\begin{aligned}
 \text{Effective bandwidth} &= \frac{\text{Packets sent}}{\text{Delay}} \\
 &= \frac{1 \text{ Packet}}{50 \times 10^{-3} \text{ sec}} \\
 &= 20 \text{ Packets per sec}
 \end{aligned}$$

- The number of packets,  $P$ , we could have sent if we didn't wait for an ACK is:

$$\begin{aligned}
 P &= \text{Bandwidth} \times \text{delay} \\
 &= \frac{10 \times 10^6 \text{ bits}}{\text{sec}} \times \frac{1 \text{ packet}}{1250 \text{ bytes}} \times \frac{1 \text{ byte}}{8 \text{ bits}} \times 10^{-3} \text{ secs} \\
 &= 50 \text{ packets}
 \end{aligned}$$

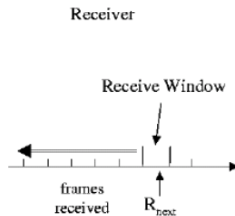
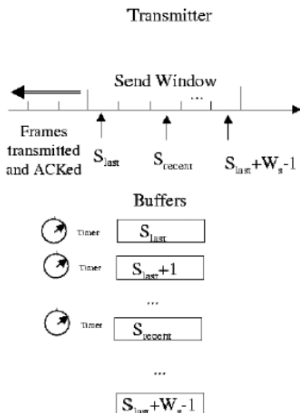
## How big should $N$ be?

The best window size  $N$  is:

- 1 Big enough to avoid data loss in the event of a NAK or timeout (must apply to packet within current window, not before it)
- 2 Small to avoid having to consume memory with unnecessary state saving

If we let the sender transmit at the maximum bandwidth, then we can compute the smallest values for  $N$  satisfying condition 1. This implies (as per our example) the optimal window size (pipeline depth) is given by the delay bandwidth product.

# Go-Back-N Diagram

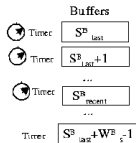
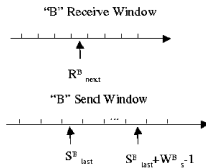
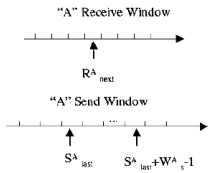
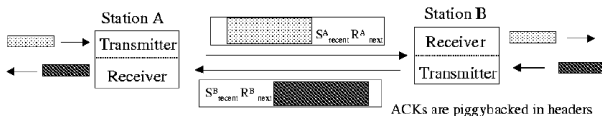


The receiver will only accept a frame that is error-free and that has sequence number  $R_{renext}$

## But what about full duplex systems?

- Suppose both ends are senders, how does that impact Go-Back-N ARQ?
- The receiver can piggyback acknowledgments on I-frame headers, which requires:
  - Using an ACK timer to trigger an ACK in the event that no I-frame is scheduled for sending
  - Ignoring erroneous frames (i.e. those with a bad CRC). However subsequent out-of-sequence, but error-free, frames can only be discarded *after* the receiver examines the ACK from the sender

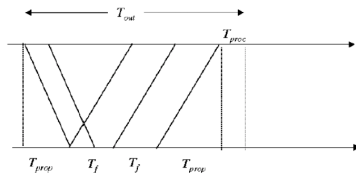
# But what about full duplex systems?



## Choosing a time out value

In bi-directional Go-Back-N protocols we can measure:

- ① Round trip propagation delay as  $2T_{prop}$
- ② The maximum I-Frame transmission time  $T_f^{max}$
- ③ The time to process a frame  $T_{proc}$



Since the worst case is when both sides send the largest possible frames the time out interval  $T_{out}$  should be at least:  $T_{out} = 2T_{prop} + T_{proc} + 2T_f^{max}$

## Go-Back-N ARQ efficiency issues

In Go-Back-N ARQ:

- 1 The Sender has to remember  $N$  I-frames
- 2 A single lost frame causes  $N$  frames to be retransmitted

But most frames are lost due to congestion and retransmission increases traffic

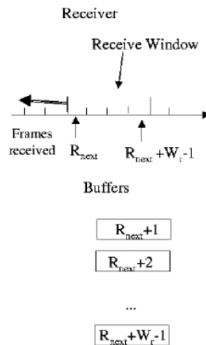
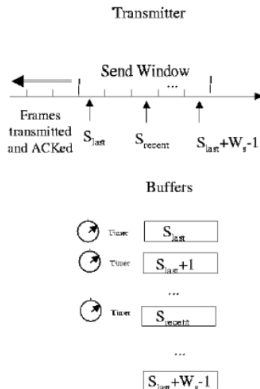
## Selective Repeat ARQ 1 of 2

- Selective Repeat ARQ improves on Go-Back-N ARQ in congested networks by reducing retransmission
- The Sender:
  - Keeps a timer for each frame sent within its send window
  - Upon getting a NAK or time-out retransmits **only the corrupted/lost frame**
  - Will not advance more than the send window  $W_S$  frames past the last acknowledged frame

## Selective Repeat ARQ 2 of 2

- But now the receiver must also be smarter!
- The receiver:
  - Remembers the sequence number of the next packet it expects to see  $R_{next}$
  - Remembers correctly formed packets which arrive within the receive window of  $W_R$  packets
  - In the event of a corrupted or lost packet, sends a (possibly piggybacked) NAK with that sequence number in response to every received packet
  - When the retransmitted packet arrives, advance  $R_{next}$  ahead the number of subsequent frames with **consecutive** sequence numbers stored in the receive window

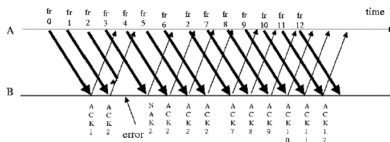
# Selective Repeat ARQ 2 of 2



[Click here for video](#)

## Selective-Repeat ARQ example

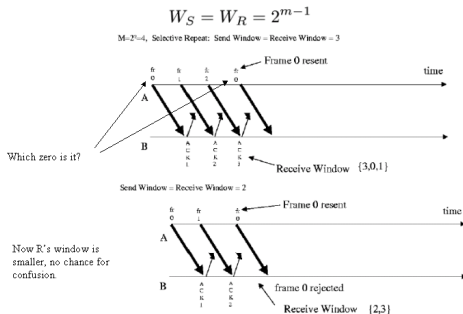
Here we see that Frame 2 was dropped and the receiver repeats requests for retransmission. Once received, the receiver and sender can advance their sliding windows.



[Click here for video](#)

## Window size in Selective Repeat ARQ

If the range of sequence numbers is  $2^m$ , then the maximum window size is:  $W_S = W_R = 2^{m-1}$



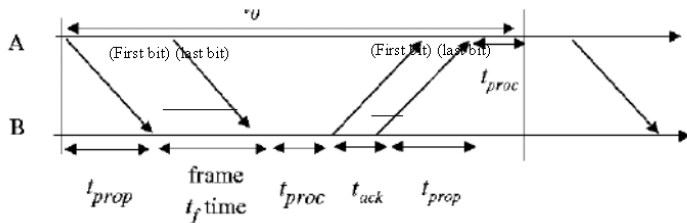
## Efficiency of Stop-and-Wait ARQ 1 of 3

First consider Stop-and-Wait ARQ without errors:

Symbol	Meaning
$\eta$	efficiency of the protocol with errors
$\eta_0$	efficiency of the protocol without errors
$n_a$	ACK frame size
$n_f$	I-Frame size
$n_o$	frame overhead
$p_f$	per packet error probability
$R$	link bandwidth (symmetric)
$R_0^{EFF}$	effective transmission rate
$t_0$	time to send a frame and get an ACK
$t_a$	ACK transmission time
$t_f$	I-Frame transmission time
$t_{prop}$	propagation delay
$t_{proc}$	time to process header or ACK

Table: Efficiency Symbols

# Efficiency of Stop-and-Wait ARQ 1 of 3



## Efficiency of Stop-and-wait ARQ 2 of 3

- By definition, and substitution we get:  
 (transmit time w/ACK)

$$t_0 = 2t_{prop} + 2t_{proc} + t_f + t_a \quad (1)$$

$$= 2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R} \quad (2)$$

(actual efficiency is less than raw capacity)

- $R_0^{EFF} = \frac{\text{bits of information}}{\text{time to deliver}}$   
 (information size/transmit time)

- $= \frac{n_f - n_o}{t_0}$

(Efficiency w/o errors)

$$\eta_0 = \frac{R_0^{EFF}}{R} = \frac{\frac{n_f - n_o}{t_0}}{R}$$

## Efficiency of Stop-and-Wait ARQ 3 of 3

Applying some algebra we get:

- (Expand values from previous slide)

$$\eta_0 = \frac{1}{R} \times \frac{n_f - n_o}{2t_{prop} + 2t_{proc} + \frac{n_f}{R} + \frac{n_a}{R}}$$

(Carry  $R$  thru denominator)

- $$= \frac{n_f - n_o}{2(t_{prop} + t_{proc})R + n_f + n_a}$$

(Multiply by one)

- $$= \frac{n_f}{n_f} \times \frac{n_f - n_o}{2(t_{prop} + t_{proc})R + n_f + n_a}$$

(Carry  $n_f$  through top and bottom)

- $$= \frac{1 - \frac{n_o}{n_f}}{1 + \frac{n_a}{n_f} + \frac{(t_{prop} + t_{proc})R}{n_f}}$$

Some notes about terms:

- The  $\frac{n_a}{n_f}$  term is the overhead to data-sent ratio

## More Notation

Symbol	Meaning
$E[total]$	mean time to transmit a frame
$P[n_t = i]$	probability of a success after $i$ errors
$W_S$	window size

## Efficiency of Go-Back-N ARQ

Take a weighted average of delays due to loss:

- Mean transmission time

$$E[\text{total}] = t_f \left[ 1 + W_S \sum_{i=1}^{\infty} (i-1)P[n_t = i] \right] \quad (\text{Go back } N \text{ (window size) when error})$$

- Probability of success after  $i$  failures

$$\begin{aligned} E[\text{total}] &= t_f \left[ 1 + W_S \sum_{i=1}^{\infty} (i-1)(1-p_f)p_f^{i-1} \right] \\ &= t_f \left[ 1 + W_S \frac{p_f}{1-p_f} \right] \\ &= t_f \left[ \frac{1 + (W_S - 1)p_f}{1-p_f} \right] \end{aligned}$$

- Plugging above into efficiency

$$\begin{aligned} R_0^{EFF} &= \frac{n_f - n_o}{E[\text{total}]} \\ \eta &= \frac{R_0^{EFF}}{R} = \frac{\frac{n_f - n_o}{E[\text{total}]}}{1 - \frac{n_o}{n_f}} \\ &= (1-p_f) \frac{R}{1 + (W_S - 1)p_f} \end{aligned}$$

- Note that  $(W_S - 1)p_f$  is the wasted time due to window size

## Efficiency of Selective-Repeat ARQ

- Similar to Go-back-N, but entire window does not need to be re-sent

- $E[total] = t_f \left[ 1 + \sum_{i=1}^{\infty} (i-1)(1-p_f)p_f^{i-1} \right]$

- $= t_f \left[ 1 + \frac{p_f}{1-p_f} \right]$

- $= t_f \left[ \frac{1}{1-p_f} \right]$

- $R_0^{EFF} = \frac{n_f - n_o}{E[total]}$

- $\eta = \frac{R_0^{EFF}}{R} = \frac{\frac{n_f - n_o}{E[total]}}{R}$

- $= (1-p_f) \left( 1 - \frac{n_o}{n_f} \right)$

## ARQ vs. flow control

- ARQ is designed to handle:
  - Occasional lost or corrupted packets/ACKS
  - Confirm delivery of data
- Flow Control
  - Negotiates Speed of Sender with Receiver
    - Sender too fast, buffer overflows
    - Sender too slow, bandwidth wasted
  - Problem: How to get the right send rate

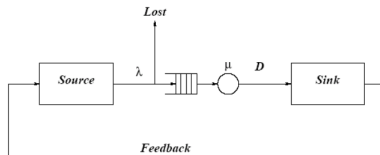
## Flow control considerations

- Flow control algorithms should be:
  - Fair
  - Simple
  - Scalable
  - Low Overhead
  - Stable
  - Can trade off one for the other
    - for example, stability for simplicity

Typically implemented at the transport level, sometimes at the data link layer

## Traffic flow model

- Traffic can be modeled as queuing for service where the entities are:
  - Sources – Creating packets
  - Buffers – Where packets are queued
  - Servers – Responsible for forwarding packets
  - Sinks – Destinations of packets



## Flow control network taxonomy

- The system can be:
  - Open loop
    - Source requests a rate
    - Network admits call
    - Source sends at this rate
  - Closed loop
    - Source monitors available service rate
    - Sends at this rate
    - Sometimes sends at the wrong rate (due to propagation delay)
  - Hybrid
    - Source requests some minimum rate
    - But will use more if available

## Open Loop

In open loop:

- Flow establishment
  - Call setup
    - Network prescribes parameters
    - User chooses parameters
    - Network admits or denies call
  - Source sends at advertised rate
    - Users transmit within parameters (self-regulation)
    - Network polices users
    - Relies on scheduling for QoS
- Challenges
  - Getting parameters to the source
  - Choosing a scheduling discipline at internal nodes
  - Call admission control

## Traffic Descriptors 1 of 2

Traffic descriptors:

- Forms an envelope describing worst case behavior
- Are used to:
  - Provide framework for traffic contract
  - Permit network policing of sources
  - Provide input to regulator
- Should provide:
  - Representivity – precisely describe the traffic
  - Verifiability – can be checked
  - Preservability – not changed within network
  - Usability – easy to design, implement and use

## Typical traffic descriptors 2 of 2

- Some examples:
  - Sequence of interarrival times (not usable)
  - Peak rate – max number of packets over some interval
    - Fixed size packets – use minimum inter-packet spacing
    - Variable size packets – maximum number of packets over all intervals of a particular duration
    - Sensitive to extremes and non-representative
    - Regulator uses timer, sending next packet after timeout
  - Average rate
    - Accumulated over a time window
    - Less susceptible to outliers
    - Window can jump or slide
  - Linear bounded arrival process (or leaky bucket) – we may hit this later

## Flow control protocol taxonomy

- First generation:
  - Ignores network state
  - Only matches receiver
- Second generation – responds to network state, categorized by:
  - State measurement – explicit or implicit
  - Control – window size or rate based
  - Point of control – endpoints or internal nodes

## Explicit vs. Implicit

- Explicit
  - Network reports its current rate
  - Better control (no guessing)
  - More overhead (need to obtain and report data)
- Implicit
  - Endpoints estimate by examining network
  - Less overhead
- What we want (but cannot have)
  - Effectiveness of explicit
  - Cost of implicit

## Flow control window vs. rate 1 of 2

- Need a timer for every connection
- Window – reminiscent of ARQ
  - Tracks largest packet outstanding (sent but not ACKed)
  - If endpoint has sent all packets in the send window, it must wait (slows down)
  - Window provides both error and control
  - Small receive window (too few buffers) implies a slow rate
  - Advantages
    - No need for fine-grained timer
    - Self-limiting

## Flow control window vs. rate 2 of 2

- Rate – controls time between sending packets
  - Adaptive rate directly controls the rate
  - Needs one timer per connection
  - Must avoid overhead or sending too much
  - Advantages
    - Better control (finer grained)
    - Separates flow control and error control

## Hop-By-Hop vs. End-to-End

- Hop-by-Hop
  - First Generation Flow Control at each link
    - Predecessor = Source
    - Successor = Sink
  - Advantages
    - Easier to implement (simplicity)
    - Distributes over network
    - Better Control
- End-to-End
  - Sender matches all “servers” on the path
  - Advantage – Cheaper

## Some flow control protocols

- ON-OFF – used in serial lines and LANs (data link layer)
  - Receiver sends ON/OFF signals
    - ON – ok to send at full speed
    - OFF – stop!
  - OK if round trip time (RTT) is small
  - What if OFF is lost? Oh no!
  - Bursty
- Stop-and-Wait – window size = 1
- Go-Back-N – static window size  $> 1$ , on loss resend whole window
- Selective-Repeat – static window size  $> 1$

## Finding a good flow control window size

- Recall that optimal window size is bandwidth round trip time (RTT) product
- But bottleneck rate is dynamic, so we can:
  - Pick a static window size and live with it
  - Adapt the window size
- Always try for the **current** optimal value

## DECbit flow control

- DECbit was developed by DEC, an early flow control protocol:
- Each packet has a bit in header
  - Bit = 0 – try a larger window size (default value)
  - Bit = 1 – try a smaller window size
- Intermediate nodes handle the bit according to queue length:
  - Queue too long – set bit to 1 (window too big)
  - Queue not too long – propagate the bit
- Sink copies the bit into ACK
- Steady state oscillates about optimal size

## DECbit routers and queuing

- Fair share – amount of bandwidth deserved (depends on scheduler)
- Sensitivity vs. stability – classic control problem
  - Over sensitive – out of control
  - Too stable – not responsive
- For each source DECbit measures:
  - Demand – how much each source sent
  - Mean queue length – over some duration (forget ancient history)
- Router actions:
  - Mean queue length  $> 1.0$  – set bits on sources demanding more than their fair share
  - Mean queue length  $> 2.0$  – set bits on all traffic (panic!)

## DECbit Source

- Track bits
- Wait for past actions to take effect (avoid over-responsiveness)
- Measure bits over past and present window size
  - More than 50% set, decrease window size
  - Otherwise increase window size
- Additive increase, multiplicative decrease

## How good is DECbit?

- Works with FIFO schedulers
- Needs per-connection state (demand)
- Can be built in software
- Limitations
  - Needs cooperative hosts
  - Has a conservative window increase policy

# TCP Flow Control

- Implicit
- Dynamic window
- **End-to-end**
- Like DECbit, but:
  - Cannot rely on router support (due to mixed version deployment)
  - Increase if no loss (timeout)
  - Decrease on timeout
  - Additive increase, multiplicative decrease
- Typical variants:
  - Router based – RED
  - Source based
    - Reno: halve the congestion window, enter a phase called fast recovery
    - Tahoe: reduce congestion window to 1, reset to slow-start state

[Click here for video](#)

## TCP RED

- Random early drop (RED) is suited for TCP, similar to DECbit
- Measure queue length
  - Queue too long – randomly select a packet in the queue and drop it
- Variant – marking instead of dropping (drop only on buffer overflow) returning mark information in ACK
- Average queue length computation – choose a weight  $0 \leq w \leq 1$  and:
  - New Avg =  $(1 - w) \times \text{Old Avg} + w \times \text{Length Now}$
  - Large  $w$  forgets prior estimates
  - Small  $w$  discounts current measurement

## TCP source-based flow control 1 of 2

- Window size initially 1
- Increase exponentially for a while, then linearly
  - Exponential – double each RTT (aka slow start)
  - Linearly – increase by 1 each RTT (aka congestion avoidance)
- When to increment window size
  - Exponential phase – for each ACK
  - Linear phase – when number of ACKs == window size

## TCP source-based flow control 2 of 2

- Slow start threshold (*ssthresh*) – a variable
  - Set when a packet is lost
  - Change from exponential to linear when window size exceeds *ssthresh*
- Variants based on window shrinkage policy:
  - Tahoe: drop window to 1 on NAK or timeout
  - Reno: on timeout, drop window to 1, on fast retransmit (NAK) halve window size

## TCP Vegas

- Expected throughput = Window Size/Propagation Delay
  - Numerator is known
  - Denominator – guess, try the smallest measured RTT
  - Actual throughput is known
  - Difference is how much to reduce or increase rate
- Algorithm
  - Send a special packet
  - On ACK, compute expected and actual throughput
  - Assume Expected – Actual  $\times$  RTT packets in bottleneck buffer
  - Adjust sending rate if this is too large
- Works better than TCP Reno

## References: For those interested in flow control

- Cela [1] has some good visualization tools and a nice introduction to a variety of control protocols.
- Keshav [2] provides a more in-depth treatment of control.
- Peterson and Davie [3] provide a good (but qualitative) treatment of DECbit, TCP RED, Tahoe and Vegas.
- Stevens [4] provides a detailed description of TCP/IP based congestion control (Chapter 21).

## References

- 1 F. Cela. A quick turn around tcp. On line at:  
<http://www.ce.chalmers.se/~fcela/tcp-tour.html>.  
Done at Chalmer's University, last updated 1999.
- 2 S. Keshav. An Engineering Approach to Computer Networking: ATM Networks, the Internet and the Telephone Network. Professional Computing Series. Addison Wesley, Reading, MA, 1997.
- 3 L. L. Peterson and B. S. Davie. Computer Networks: A Systems Approach. Morgan Kaufmann, San Francisco, CA, USA, 2 edition, 2000.
- 4 Stevens:W.R. TCP/IP Illustrated, Volume 1: The Protocols, volume 1. Prentice Hall, Upper Saddle River, NJ, 1 edition, 1994.