Internet Protocols
Fall 2006

Lecture 3

Andreas Terzis

terzis@cs.jhu.edu

Outline

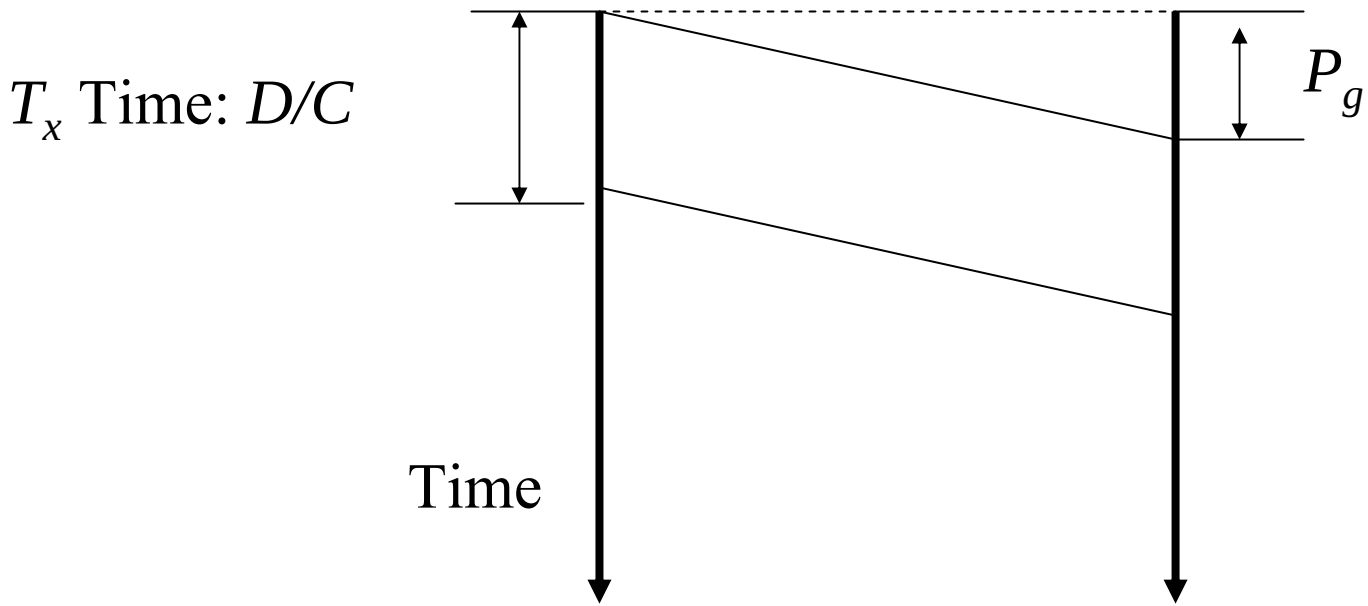
- Framing
- Error Detection and Correction
  - Parity and CRCs
- Reliable transfer
  - Automatic Repeat reQuest (ARQ) Protocols
    - Send-and-Wait
    - Go Back N
    - Selective Repeat
  - FEC

Physical Layer Abstraction

- Physical Layer Service: Deliver individual bits to the other side of a physical link
- Link Characteristics
  - Speed C (bits/sec)
  - Propagation Delay P_g
  - Loss rate p

Link characteristics

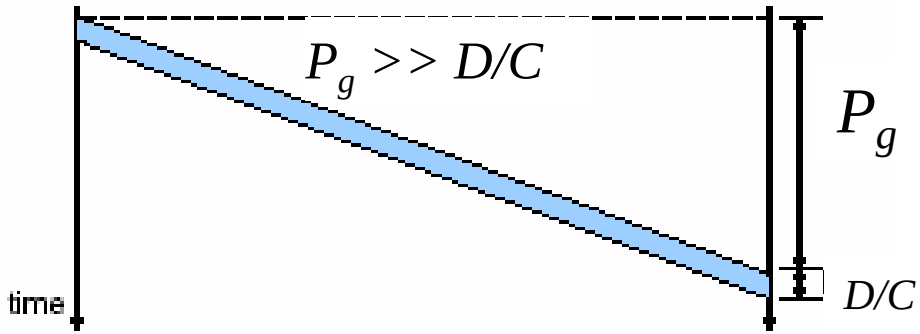
- Link Speed: C bps
- Propagation Delay: P_g sec
- frame Size: D bits



Examples

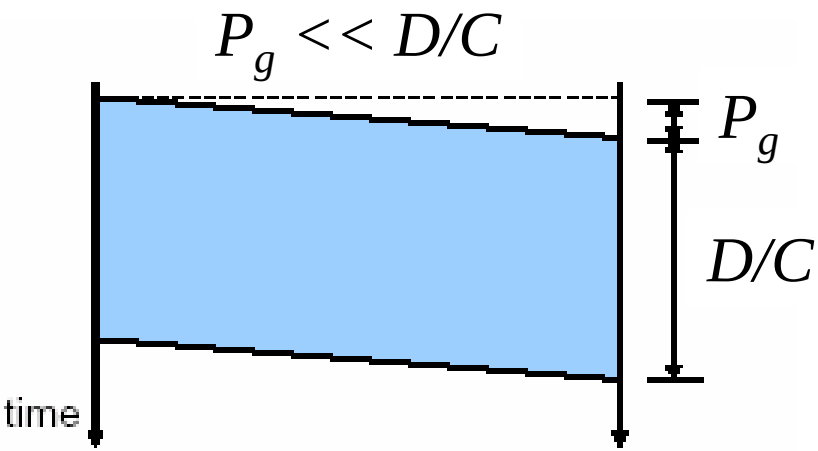
- D=1Kbyte
  C = 1Gbps
  100km fiber
  P_g=500 usec

  D/C = 8usec



  D=1Kbyte
  C=100 Mbps
  1Km fiber
  P_g=5usec

  D/C = 80usec



Data Link Layer Frame

- Terminology: A block of data
  - At link layer: normally called a data frame
  - At network layer: a packet
  - At transport level: TCP — segment; UDP — datagram
- A frame/packet has a header field
  - Optionally there may be a trailer too

  - Question: How do we detect the beginning/end of a frame?

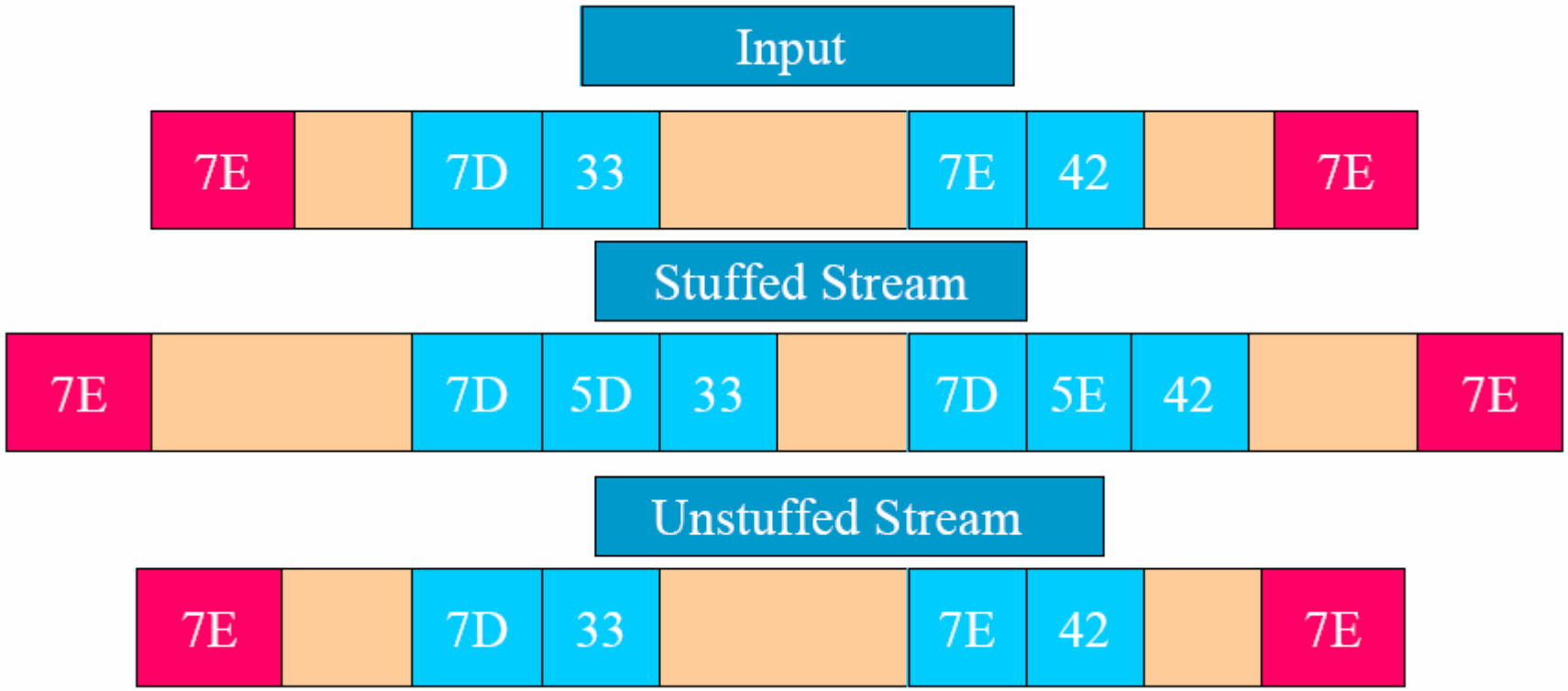
Methods to Identify Frames

- Byte Counts
- Starting/ending flags (bit stuffing)
- Using physical layer violations (i.e. invalid physical layer codes)
- Byte-Oriented Framing Protocol:
  - Delineate frame with a special bit sequence: 01111110 (0x7E)
  - Q: What if the bit sequence 0x7E occurs in data stream?

Byte stuffing

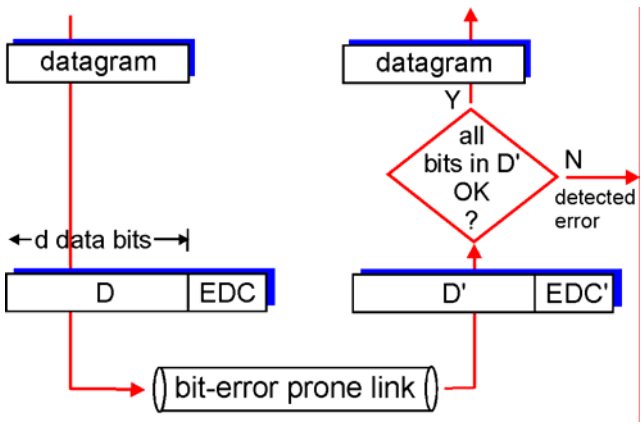
- Example: solution followed by PPP
- Sender:
  - If 0x7E appears in data it is replaced with 0x7D,0x5E
  - If 0x7D appears in the data it is replaced with 0x7D,0x5D
- Receiver:
  - If 0x7D is found in data, it is discarded and XORs the following byte with 0x20 to recreate original input
- Stuffing changes the total length of data to be sent
- Header, Trailer vs. Byte Count: First easier to implement in hardware

Byte Stuffing Example



Error Detection

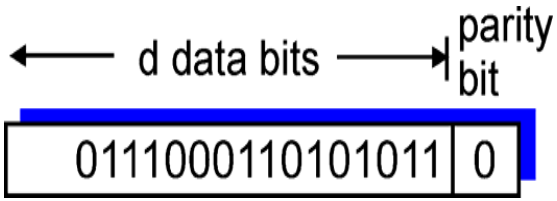
- EDC= Error Detection and Correction bits (redundancy)
- D = Data protected by error checking
- Error detection not 100% reliable!
  - protocol may miss some errors, but rarely
  - larger EDC field yields better detection and correction
- Tradeoff: Larger EDC provides higher protection but adds overhead



Parity Checking

Single Bit Parity:

Detect single bit errors

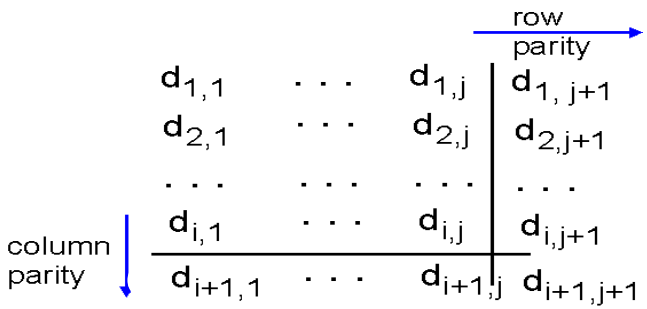


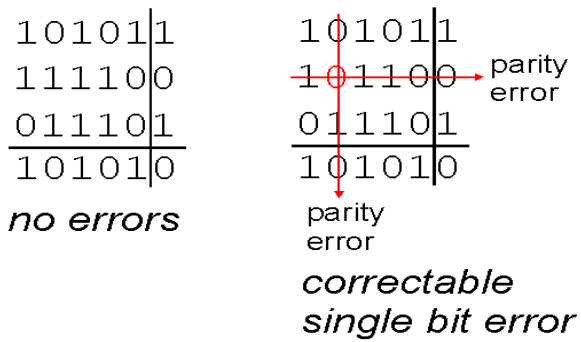
consider a data frame as a $m \times n$ matrix

- A parity bit for each row $\rightarrow$ n-bit checksum, and
- A parity bit for each column $\rightarrow$ m-bit checksum

Two Dimensional Bit Parity:

Detect *and correct* single bit errors





Cyclic Redundancy Check (CRC)

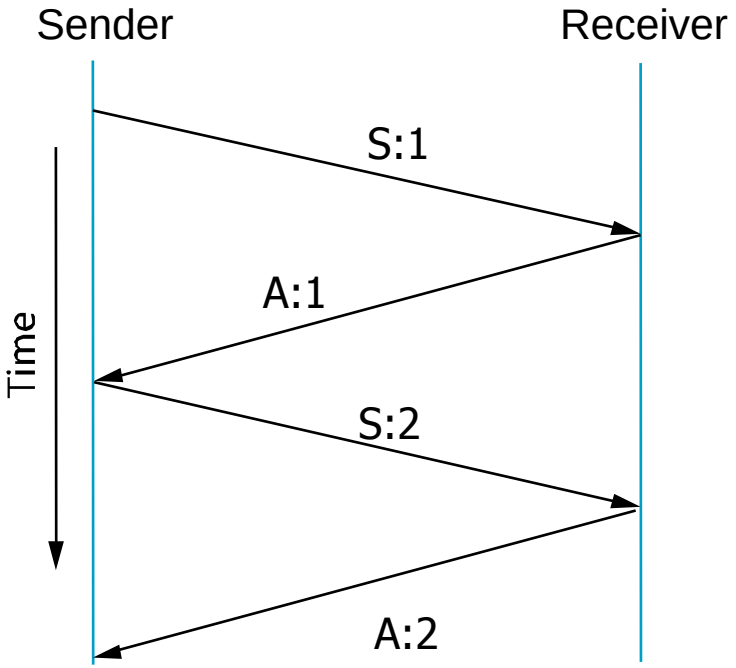
- Consider a data frame as a bit sequence $M(x)$
  - e.g. 10011010 --> $M(x) = x^7 + x^4 + x^3 + x$
  - *n*-bit message is represented by a polynomial with degree of *n-1*
    - e.g. 10011010 represents a 8-term polynomial with degree 7
- Use a generator (divisor) polynomial $C(x)$ of degree k to compute the checksum for $M(x)$
  - Sender makes the transmitted bit sequence divisible by $C(x)$
  - Receiver divides the received sequence by $C(x)$
    - If perfectly divisible then no errors (with high probability)
    - Otherwise, frame contains bit errors

CRC Characteristics

- Errors detected
  - All single-bit errors as long as x^k and x^0 terms have nonzero coefficient
  - All double-bit errors as long as $C(x)$ has a factor with at least three terms
  - Any odd number of errors as long as $C(x)$ contains the factor $x+1$
  - Any burst error for which the length of the burst is less than k
- Easily implementable in hardware

Reliable transmission

- All ARQ protocols use the same three mechanisms
  - Sequence numbers
  - Acknowledgements
  - Timeouts
- Simplest ARQ protocol: Send-and-Wait
  - Sender Side Protocol
    - Send a frame, wait for ACK or timeout
    - If ACK arrives send next frame
    - Otherwise retransmit current frame
  - Receiver side Protocol
    - Send ACK for each received frame

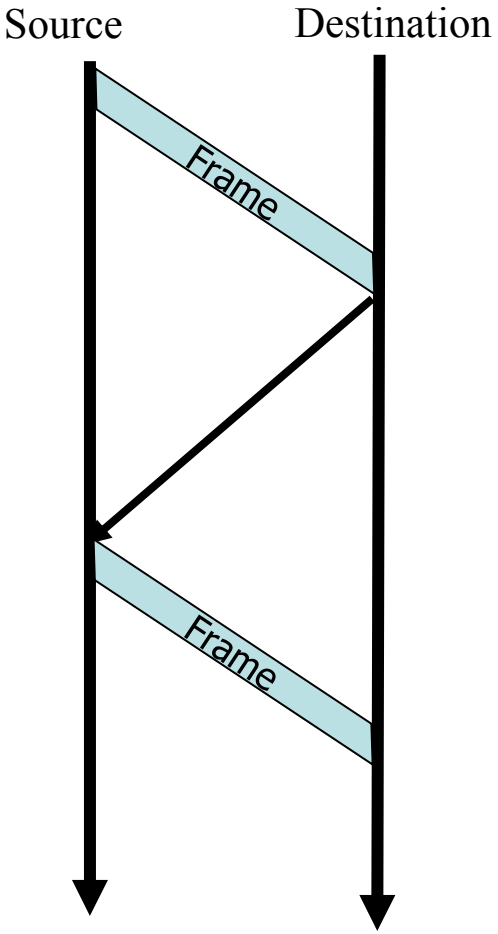


Link Utilization

- Round-Trip Time (*RTT*): Two-way propagation delay from sender to receiver and back
- Throughput of a link: Total number of frames successfully transmitted during "transmission cycle" T_s
- Link utilization = percentage of time the link is used to send user data
- Bandwidth-delay product
  - Storage capacity of the link: *C*RTT*

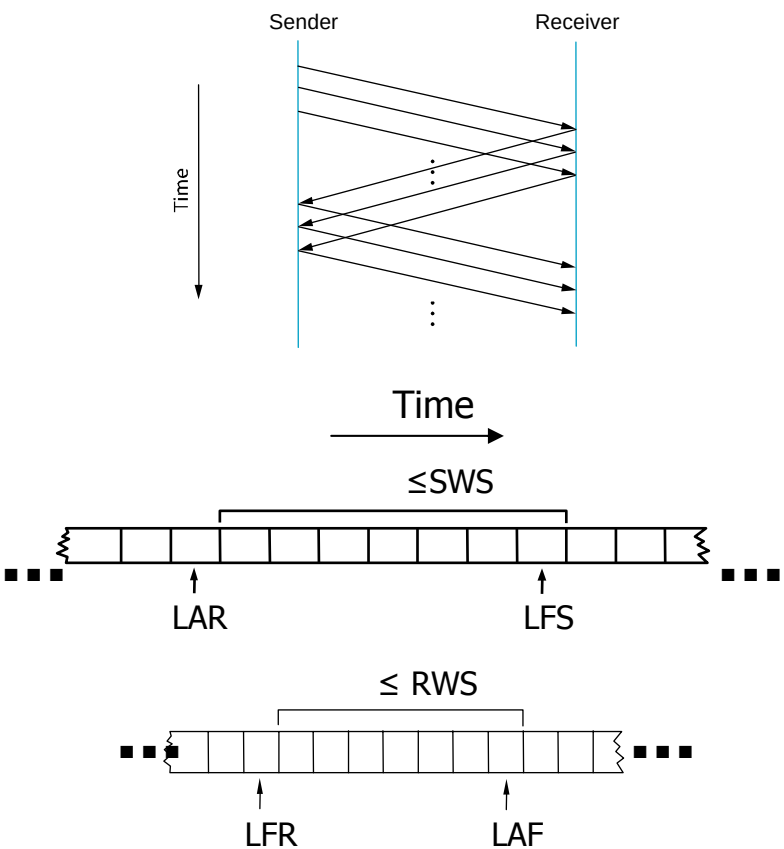
Sliding Window Protocols

- Problem: Send-and-Wait has low utilization in links with long RTT
- Solution: Sliding window protocols
  - Send W frames before ACK arrives



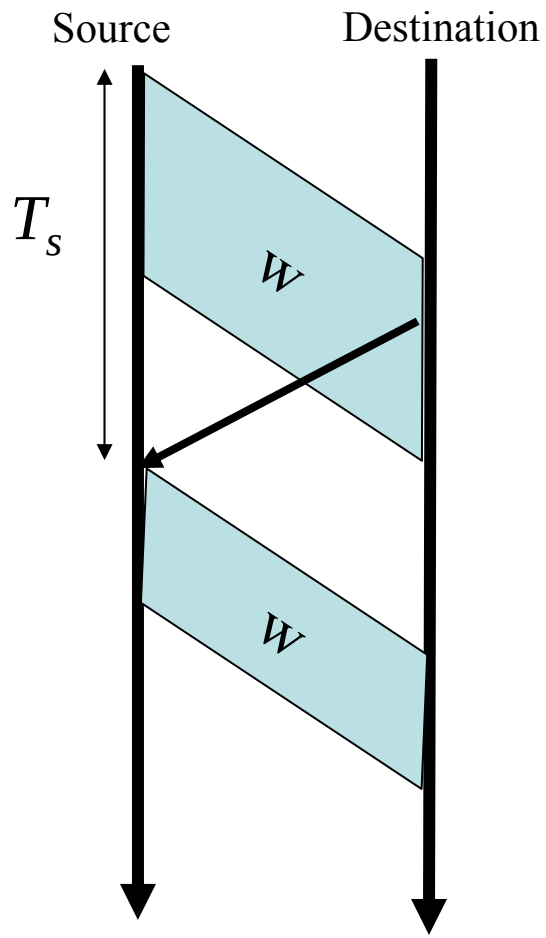
Acknowledgement Strategies

- Sender and receiver implement sliding window of frames in transit
  - Sender side holds window of un-acknowledged frames
    - LFS-LAR ≤ SWS
  - Receiver side holds window of frames received out of order
    - LAF-LFR ≤ RWS
- Acknowledgement Strategies
  - Cumulative Ack: send LFR
  - Negative Ack
  - Selective Ack



Efficiency of Sliding Window Protocols

- Sliding window protocol:
  - Send W bits (window size)
  - Wait for ACKs
  - Repeat
- Throughput $= W/T_s$
  - W=64 Kbytes
  - T_s=200ms
  - Throughput = W/T_s = 2.6Mbps

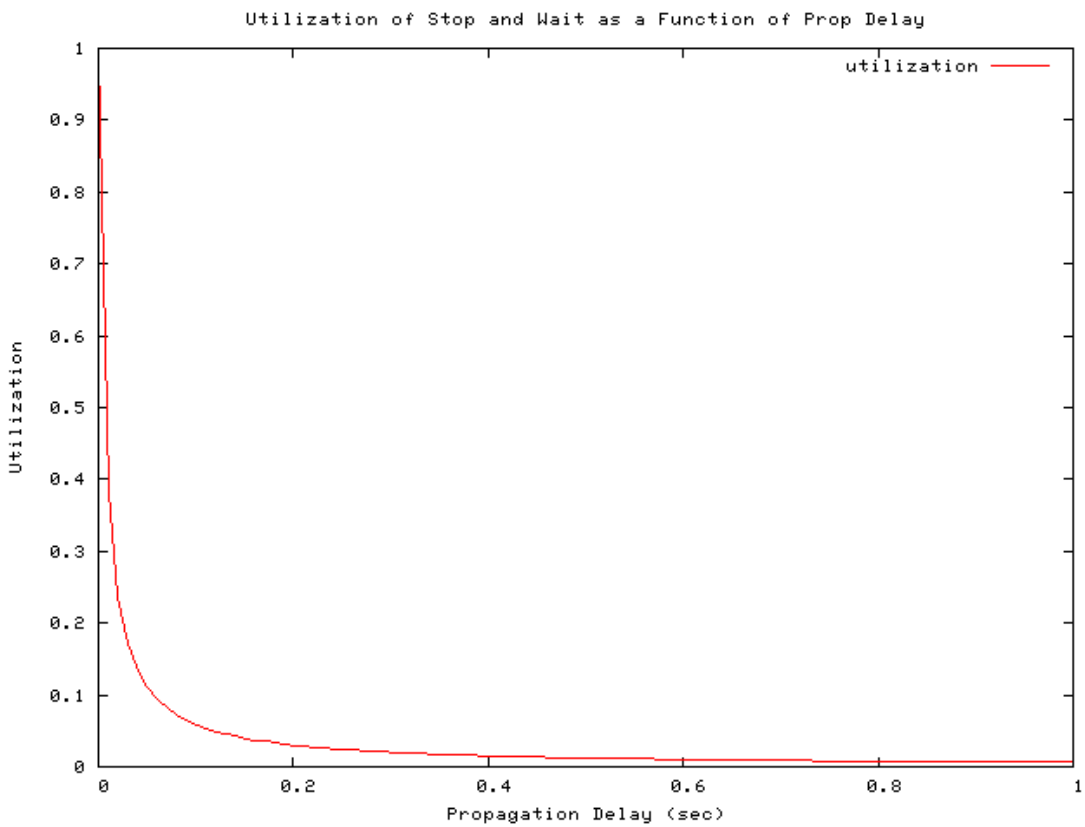


Link Utilization (no loss)

- Send and Wait: $$U_s = \frac{D}{D+H+A} \bullet \frac{1}{1+\frac{2C(P_r+P_g)}{D+H+A}}$$
- Sliding Window:

$$U_w = \begin{cases} \frac{DW}{D+H+A} \cdot \frac{1}{\frac{1+2C(P_r+P_g)}{D+H+A}}, W < W_c \\ \frac{D}{D+H}, W \geq W_c \end{cases}$$

Utilization Plot



Utilization in the presence of errors

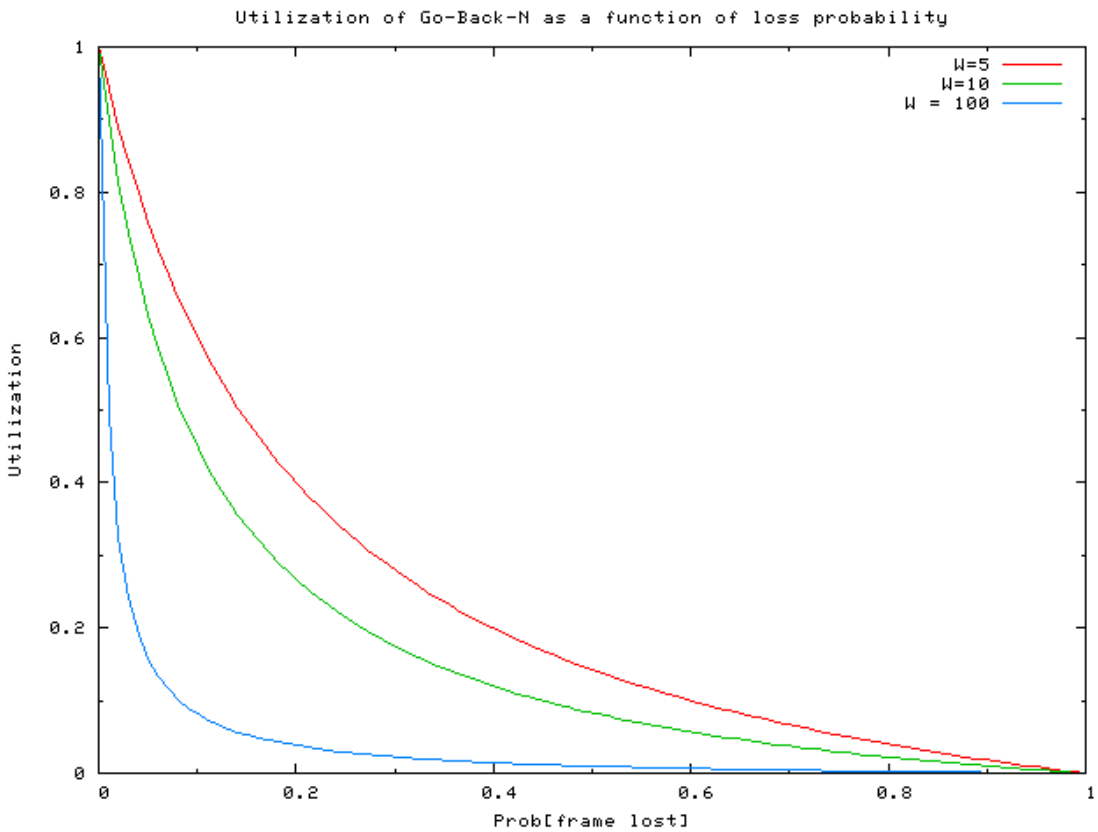
Prob[packet lost]=p

Send and Wait:

$$U_s = \frac{(1-p)\,D/C}{2(P_g + P_r) + \dfrac{D+H+A}{C}}$$

Go-Back-N:

$$U_w = \frac{D/C \cdot (1-p)}{\dfrac{D+H}{C} + p\left(\dfrac{D+H+A}{C} + 2(P_g + P_r)\right)}, W > W_c$$

$$U \le \frac{1-p}{1+W_c p}$$

Go-Back-N Utilization