

Problem: physically connecting hosts

Direct link networks:

- point-to-point links
- shared access networks

Point-to-Point Links

Hongwei Zhang

<http://www.cs.wayne.edu/~hzhang>



It is a mistake to look too far ahead. Only one link in the chain of destiny can be handled at a time.

--- Winston Churchill

Acknowledgement: this lecture is partially based on the slides of Dr. Larry Peterson

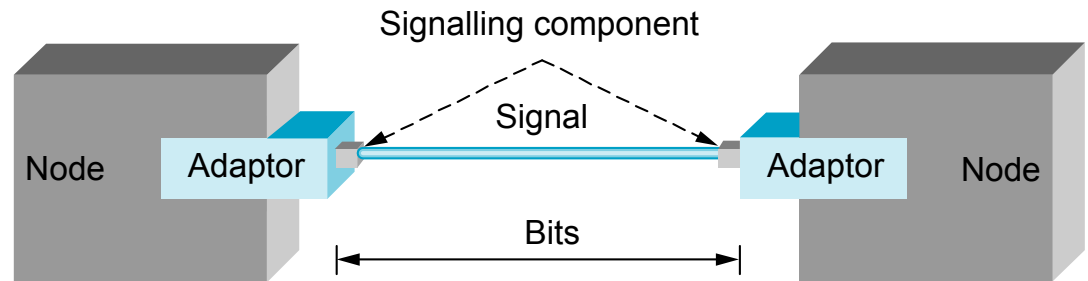
Outline

- Encoding
- Framing
- Error detection
- Reliable transmission

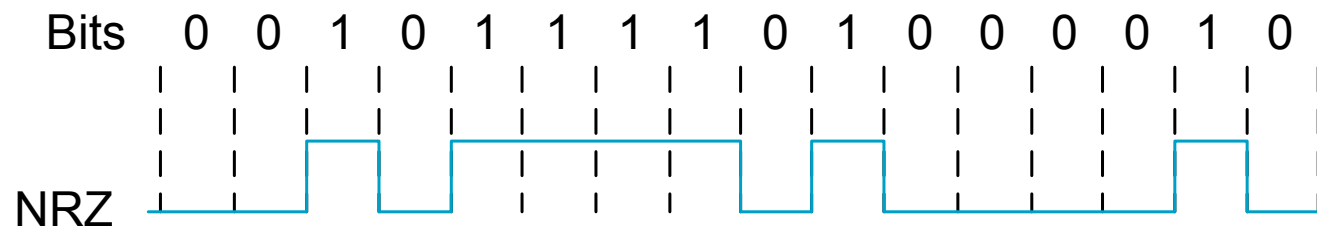
Outline

- Encoding
- Framing
- Error detection
- Reliable transmission

Encoding



- Signals propagate over a physical medium
 - modulate electromagnetic waves
 - e.g., vary voltage
- Problem: encode binary data onto signals
 - the obvious thing to do, for instance, is to encode 0 as low signal and 1 as high signal
 - known as Non-Return to zero (NRZ)



Problems of NRZ: consecutive 1s or 0s

- Unable to recover clock
 - Need frequent transitions to automatically synchronize sender and receiver without using extra links
- Baseline wander
 - Receiver keeps an average of the signal (i.e., its voltage level) it has seen so far, and use threshold-based method to distinguish between low and high signals
 - Too many consecutive 1s and 0s cause this average to change, making it more difficult to detect a significant change in signal

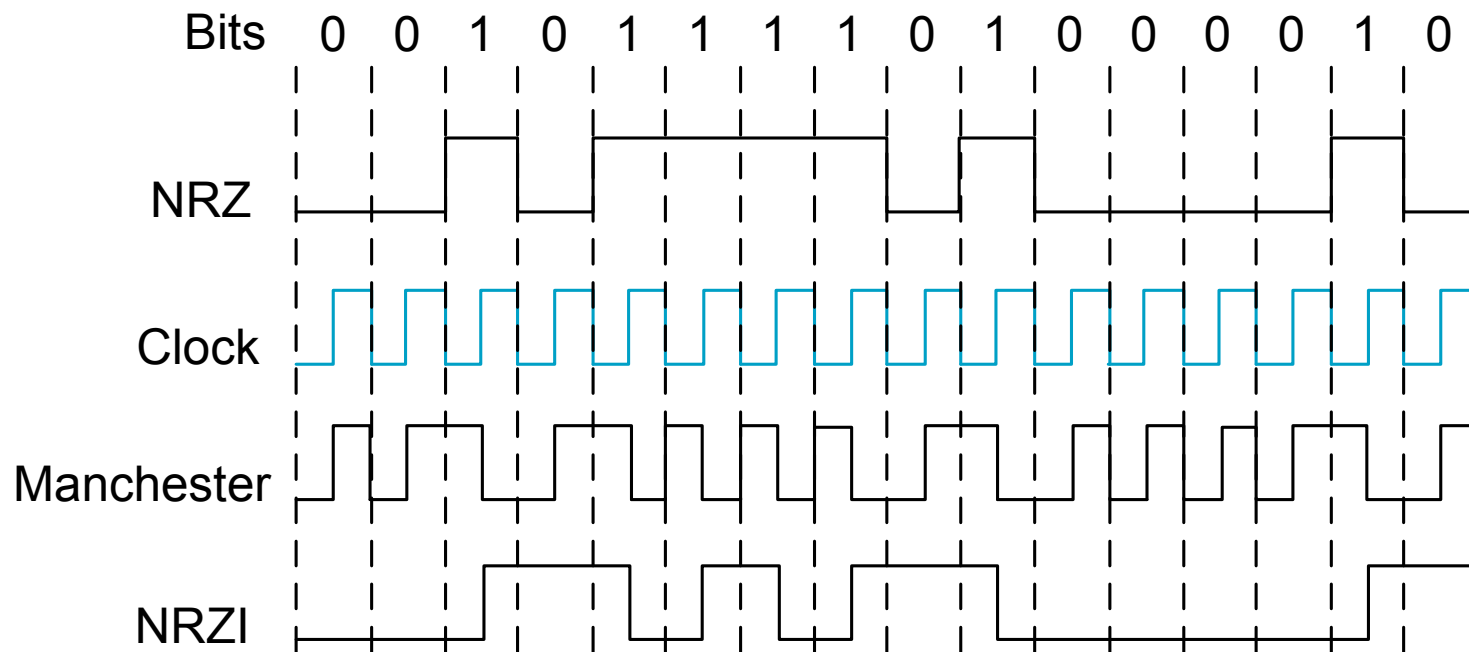
Alternative Encodings

- Non-return to Zero Inverted (NRZI)
 - make a transition from current signal to encode a 1; stay at current signal to encode a 0
 - (+) solves the problem of consecutive 1s
 - (-) does not solve the problem of consecutive 0s
- Manchester
 - transmit XOR of the NRZ encoded data and the clock
 - 0: low-to-high transition
 - 1: high-to-low transition
 - (-) only ~50% efficient in bandwidth usage (bit rate = 1/2 baud rate)
 - 2 transitions for a bit if it is the same as the preceding bit

Encodings (contd.)

- 4B/5B
 - every 4 bits of data are encoded in a 5-bit code
 - 5-bit codes selected to have no more than one leading 0 and no more than two trailing 0s
 - thus, never get more than three consecutive 0s
 - resulting 5-bit codes are transmitted using NRZI
 - (*) achieves 80% efficiency
- Used in FDDI (Fiber Distributed Data Interface)

Encodings (contd.)



Encoding (contd.)

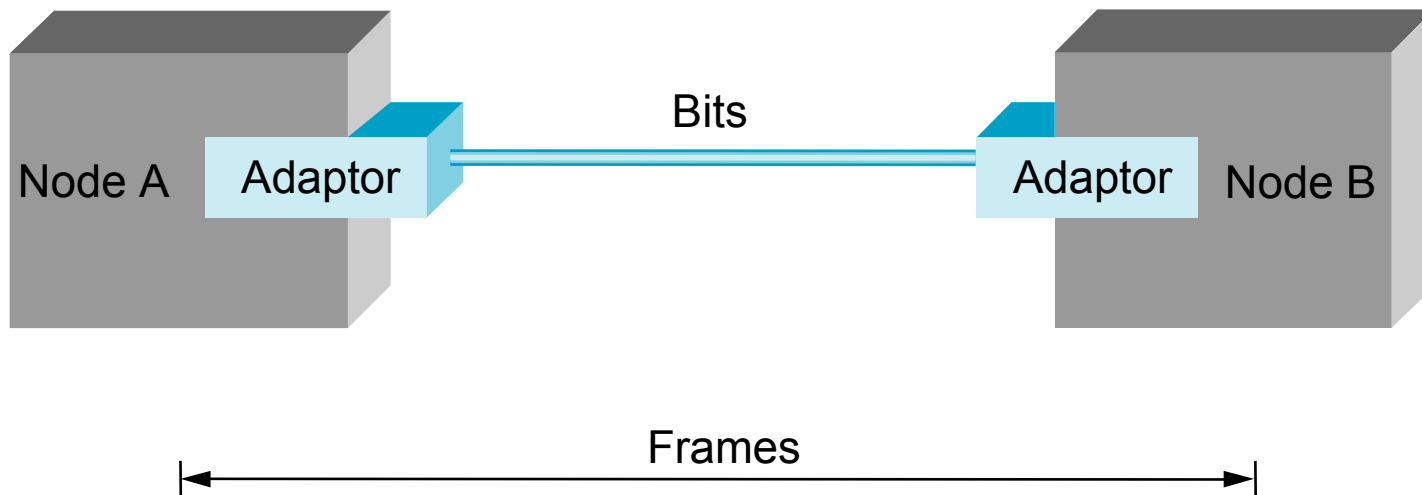
- In practice, several encoding schemes may be used in a single system depending on the traffic patterns
 - E.g., the “overhead” bytes of SONET frames are encoded using NRZ, but the data payload are encoding using other methods (i.e., XOR of data and a well-known bit pattern, which is 127-bit long and has plenty of $1 \leftrightarrow 0$ transitions)

Outline

- Encoding
- Framing
- Error detection
- Reliable transmission

Framing

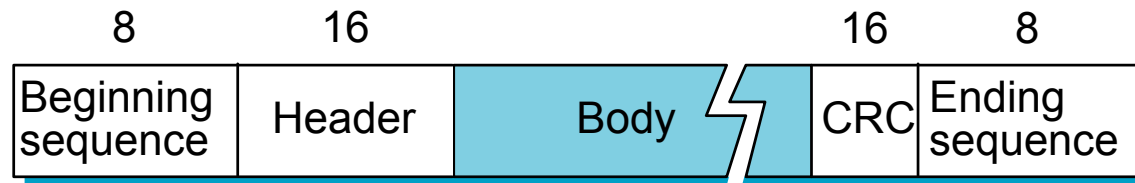
- Break sequence of bits into a frame
- Typically implemented by network adaptor



Approaches

- Sentinel-based

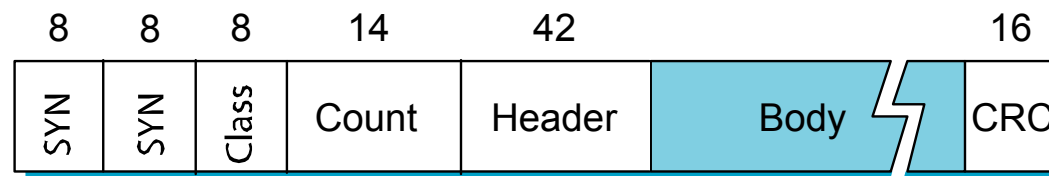
- delineate frame with special pattern: 01111110
- e.g., SDLC (IBM), HDLC (ISO), PPP (dial-up link)



- Problem-0: special pattern appears in the payload
 - solution: *bit stuffing*
 - sender: insert 0 after five consecutive 1s
 - receiver: delete 0 that follows five consecutive 1s
- Problem-1: special pattern field corrupted
 - solution: catch when CRC fails

Approaches (contd.)

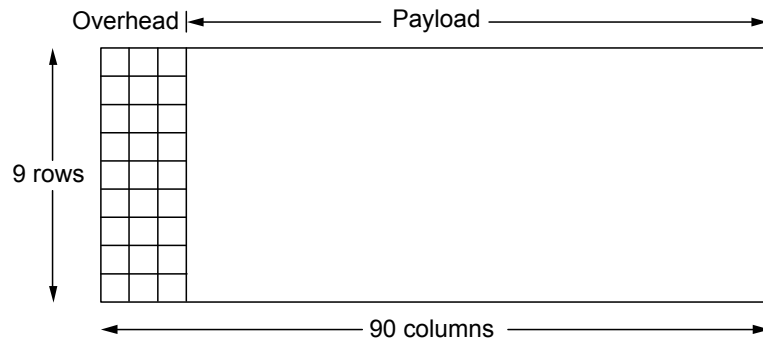
- Counter-based
 - include payload length in header
 - e.g., DDCMP (used in DECNET)



- problem: count field corrupted
- solution: catch when CRC fails

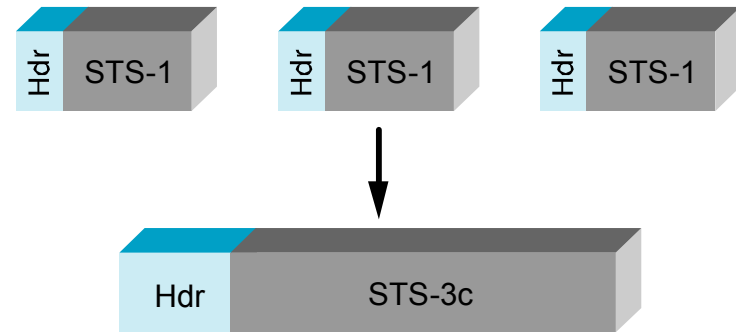
Approaches (cont)

- Clock-based
 - each frame is 125us long
 - e.g., SONET: Synchronous Optical Network
 - STS- n (STS-1 = 51.84 Mbps)



SONET STS-1 frame:

- 9 rows, with each row having 90 bytes
- overhead bytes are used not only for frame delineation, but also for resource reservation (for voice, for instance) and connection management
 - SONET runs over carrier network rather than a single link



Outline

- Encoding
- Framing
- Error detection
- Reliable transmission

Cyclic Redundancy Check

- Add k bits of redundant data to an n -bit message
 - want $k \ll n$
 - e.g., $k = 32$ and $n = 12,000$ (1500 bytes) --- CRC-32
- Represent n -bit message as $n-1$ degree polynomial
 - e.g., MSG=10011010 as $M(x) = x^7 + x^4 + x^3 + x^1$
- Let k be the degree of some *divisor polynomial*
 - e.g., $C(x) = x^3 + x^2 + 1$

CRC (cont)

- Transmit polynomial $P(x)$ that is evenly divisible by $C(x)$
 - shift left k bits, i.e., $M(x)x^k$
 - subtract remainder of $M(x)x^k / C(x)$ from $M(x)x^k$
 - Modulo 2 arithmetic: subtract \equiv XOR
 - Let's do an exercise: $MSG = 10011010$, $C(x) = x^3 + x^2 + 1$ (p.98)
- Receiver polynomial $P(x) + E(x)$
 - $E(x) = 0$ implies no errors
- Divide $(P(x) + E(x))$ by $C(x)$; remainder zero if:
 - $E(x)$ was zero (no error), or
 - $E(x)$ is exactly divisible by $C(x)$ (error goes undetected in this case)

Selecting $C(x)$

- Can detect
 - All single-bit errors, as long as the x^k and x^0 terms have non-zero coefficients.
 - All double-bit errors, as long as $C(x)$ contains 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 (i.e., sequence of consecutive error bits) for which the length of the burst is less than k bits.
 - Most burst errors of larger than k bits can also be detected
- See Table 2.5 on page 101 for common $C(x)$

Internet Checksum Algorithm

- View message as a sequence of 16-bit integers; sum using 16-bit ones-complement arithmetic; take ones-complement of the result.

```
u_short
cksum(u_short *buf, int count)
{
    register u_long sum = 0;
    while (count--)
    {
        sum += *buf++;
        if (sum & 0xFFFF0000)
        {
            /* carry occurred, so wrap around */
            sum &= 0xFFFF;
            sum++;
        }
    }
    return ~(sum & 0xFFFF);
}
```

In ones complement arithmetic,

- a negative integer $-x$ is represented as the complement of x
- a carryout from the most significant bit needs to be added to the results

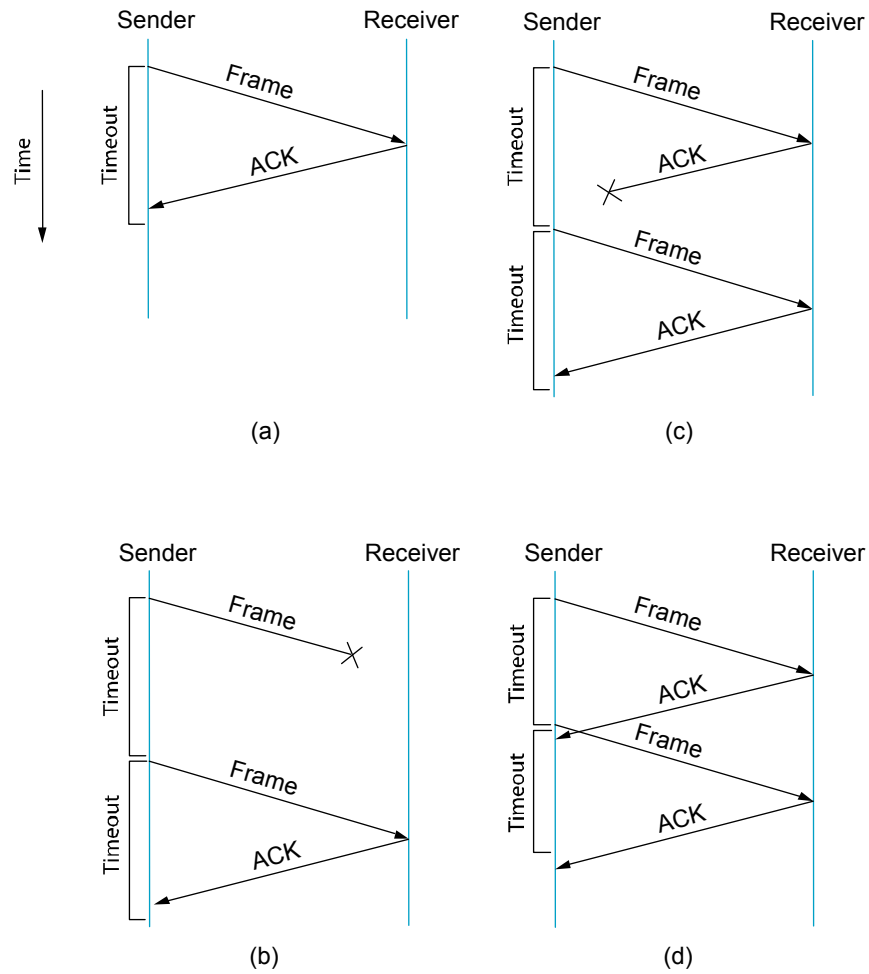
Outline

- Encoding
- Framing
- Error detection
- **Reliable transmission**

Reliable transmission in spite of bit and burst errors ?

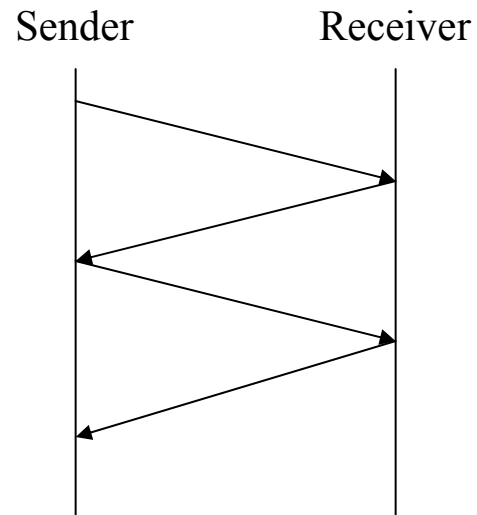
- Forward error correction does not guarantee reliable transmission or is not cost effective
- “Acknowledgment (from receiver)” and “timeout (at sender)” as the basic mechanism for reliable transmission
 - Stop-and-wait
 - Sliding window
 - Concurrent logical channels

Stop-and-Wait



4 different scenarios

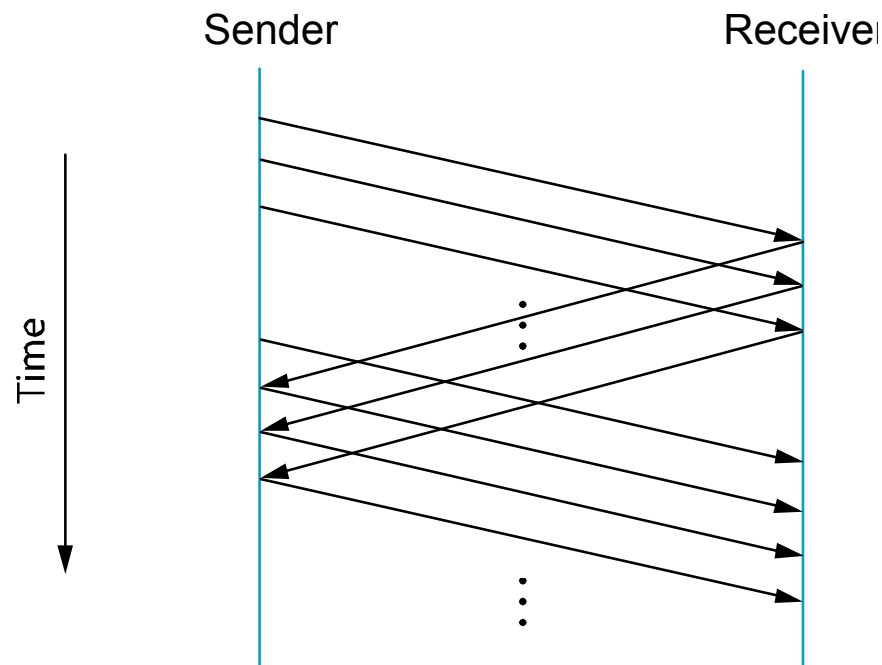
Stop-and-wait (contd.)



- Problem: how to keep the pipe full
- Example
 - $1.5\text{Mbps link} \times 45\text{ms RTT} = 67.5\text{Kb (8KB)}$
 - 1KB frames implies 1/8th link utilization

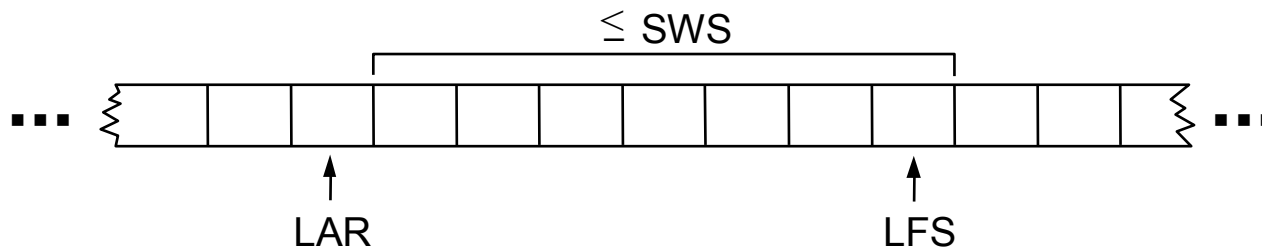
Sliding Window

- Allow multiple outstanding (un-ACKed) frames
- Upper bound on un-ACKed frames, called *window*



SW: Sender

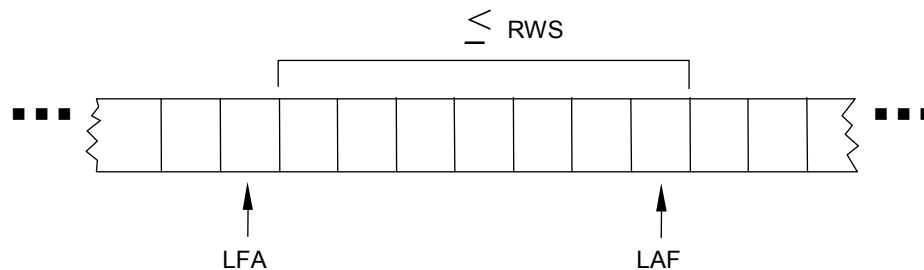
- Assign sequence number to each frame (**seqNum**)
- Maintain three state variables:
 - send window size (**sWS**): depends on channel capacity
 - last acknowledgment received (**LAR**)
 - last frame sent (**LFS**)
- Maintain invariant: **LFS - LAR ≤ sWS**



- Advance **LAR** when ACK arrives
- Buffer up to **sWS** frames

SW: Receiver

- Maintain three state variables
 - receive window size (**RWS**): depends on local memory size, and $\leq \mathbf{SWS}$
 - largest acceptable frame (**LAF**)
 - last frame acknowledged (**LFA**)
- Maintain invariant: $\mathbf{LAF} - \mathbf{LFA} \leq \mathbf{RWS}$



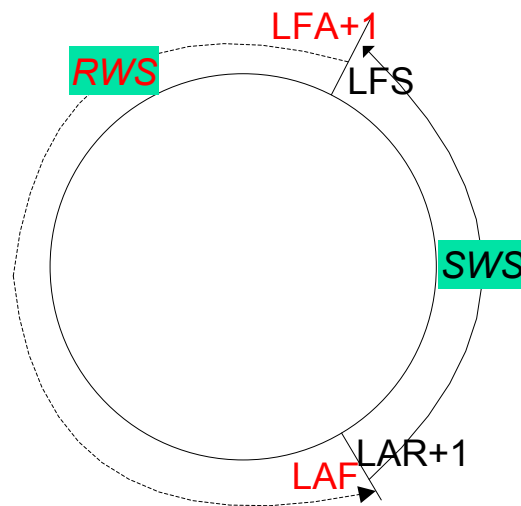
- Frame **SeqNum** arrives:
 - if $\mathbf{LFA} < \mathbf{SeqNum} \leq \mathbf{LAF}$ \longrightarrow accept
 - if $\mathbf{SeqNum} \leq \mathbf{LFA}$ or $\mathbf{SeqNum} > \mathbf{LAF}$ \longrightarrow discarded
- Send cumulative ACKs

Sequence Number Space

- **SeqNum** field is finite; sequence numbers wrap around
- Sequence number space must be larger than number of outstanding frames
- Is $SWS \leq |\text{SeqNumSpace}| - 1$ sufficient? Not always
 - suppose 3-bit **SeqNum** field (0..7)
 - $SWS=RWS=7$
 - sender transmit frames 0..6
 - *arrive successfully, but ACKs lost*
 - sender retransmits 0..6
 - receiver expecting 7, 0..5, but receives second incarnation of 0..5
- when $SWS == RWS$,
 - $SWS \leq |\text{SeqNumSpace}| / 2$ is the correct rule
 - Intuitively, **SeqNum** “slides” between two halves of sequence number space

Q: General rule on SWS, RWS, seq. number space?

- Expected frames should not overlap (in a wrap-around manner) with retransmitted frames
- Worst case: all ACKs are lost



LAR: last ack received

LFS: last frame sent

SWS: sender window size

LFA: last frame acked

LAF: largest acceptable frame

RWS: receiver window size

- General rule: $SWS + RWS \leq |\text{SeqNumSpace}| = \text{MaxSeqNum} + 1$ (i.e., size of seq. # space)

Other roles that sliding window protocol can play

- In-order frame delivery
- Flow control to avoid overwhelming receiver
 - Subject to constraint of the channel capacity and receiver memory size
- More important at higher layers such as TCP and application

Concurrent Logical Channels (ARPANET)

- Multiplex 8 logical channels over a single link
- Run stop-and-wait on each logical channel
- Maintain 3 state bits per channel
 - channel busy
 - current sequence number out
 - next sequence number in
- Header: 3-bit channel num, 1-bit sequence num
 - 4-bits total
 - same as sliding window protocol
- Separates *reliability* from *order*

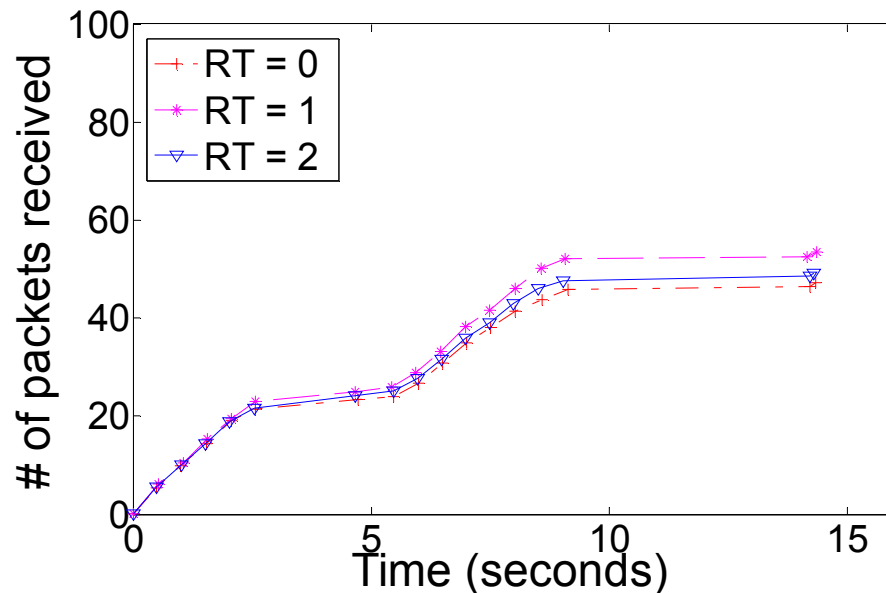
A Line in the Sand:



Error control in sensor networks

Metrics	$RT=0$	$RT=1$	$RT=2$
<i>Reliability (%)</i>	51.05	54.74	54.63
<i>Latency (sec)</i>	0.21	0.25	0.26
<i>throughput (pkt/sec)</i>	4.01	4.05	3.63

- Retransmission does not help much, and may even decrease reliability and throughput
- Similar observations when adjusting contention window of B-MAC and using S-MAC



Problem statement

- How to achieve
 - close to 100% reliability?
 - close to optimal event throughput (implies real-time packet delivery)?
- Elements of the solution RBC
 - Differentiated contention control
 - Window-less block acknowledgment
 - Fine-grain timer management

H. Zhang, A. Arora, Y.R. Choi, M. Gouda, [Reliable Bursty Convergecast in Wireless Sensor Networks](#), *ACM MobiHoc 2005, Computer Communications (Elsevier)*