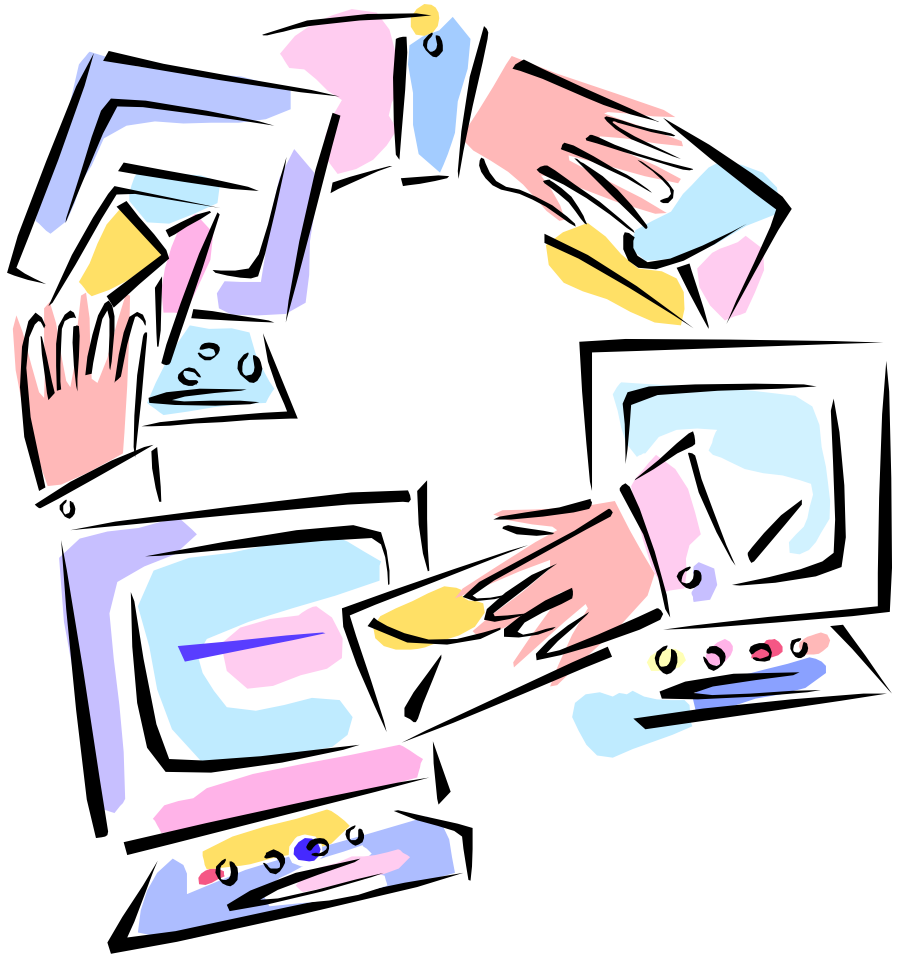
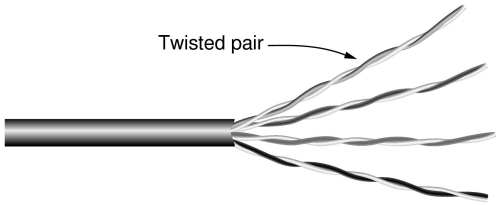
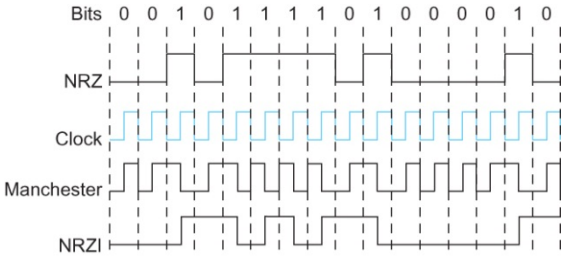
Reliable transmission



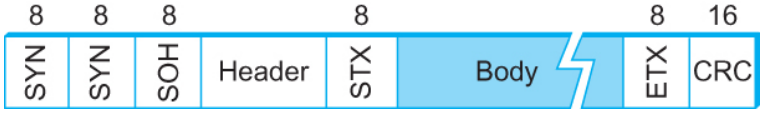
Getting connected thus far

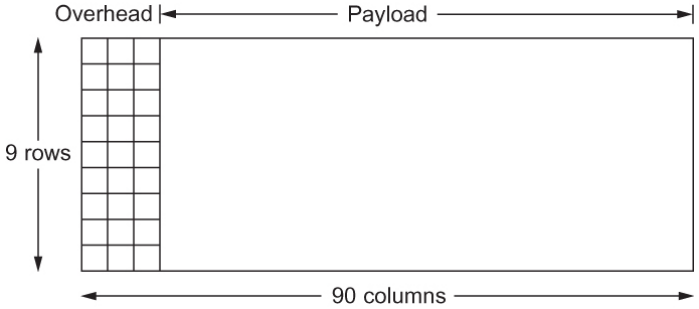
- Physical connectivity





- Aggregating bits into frames

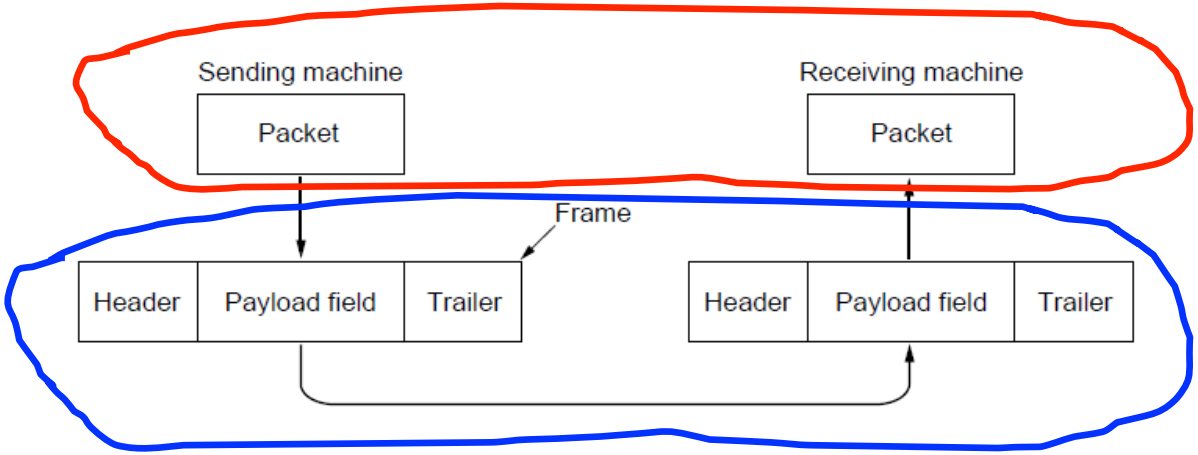




- Detecting errors in frames

Message	1001 1010	$M(x) = x^7 + x^4 + x^3 + x^1$
Generator	1101	$C(x) = x^3 + x^2 + 1$
CRC	101	

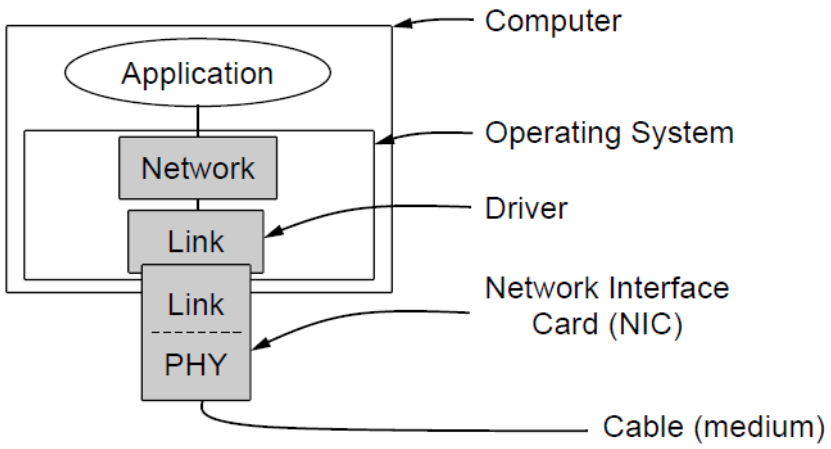
Data link layer



Network layer

Data link layer

Physical layer



Link layer

1) Well-defined service interface to network layer
2) May deal with transmission errors
3) May provide flow control, don't swamp the receiver

Reliable transmission

- Networks need reliable delivery
  - Forward error-correction
    - High overhead
    - Can only recover from some errors
  - Discard frames with bad checksum / CRC
  - May occur at link layer (e.g. G.hn, powerline net)
  - Often at higher layer (e.g. TCP at transport layer)
    - Basic algorithms and concepts the same

Reliable transmission

- Main mechanisms for reliable delivery:
  - Acknowledgements (ACK)
    - Control frame, informs peer frame(s) received okay
    - Different types
      - Selective acknowledgement, specifies received frame
      - Cumulative acknowledgement, received this frame and all previous
      - Negative acknowledgement (NACK), frame was corrupt or out of buffer space
  - Timeouts
    - Only wait so long for ACK (frame or ACK may be MIA)

Reliable transmission

- Automatic repeat request (ARQ) algorithm
  - Sender waits for acknowledgement (ACK) before advancing
  - If no ACK after timeout value, resend frame
  - Three main ARQ algorithms
    - Stop-and-wait
    - Concurrent logical channels
    - Sliding window

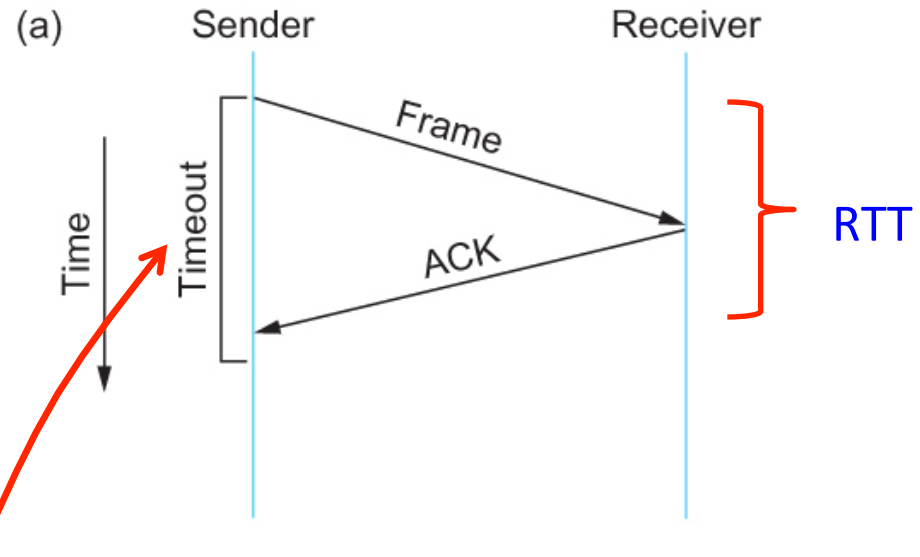
Goals of ARQ

- Reliable transmission
- Preserve order
  - Delivers data in same order to receiver's network layer that sender's network layer intended
- Flow control
  - Receiver can throttle sender
  - Sender can't overrun processing/buffer capacity of the receiver

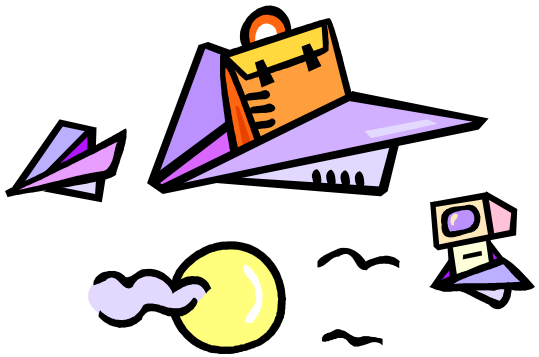
Stop-and-wait

- Stop-and-wait algorithm:
  1. Send a frame, start a timer
  2. Wait for an ACK
  3. If timeout before ACK, goto 1
  4. If ACK, get next frame, goto 1

Stop-and-wait: Success

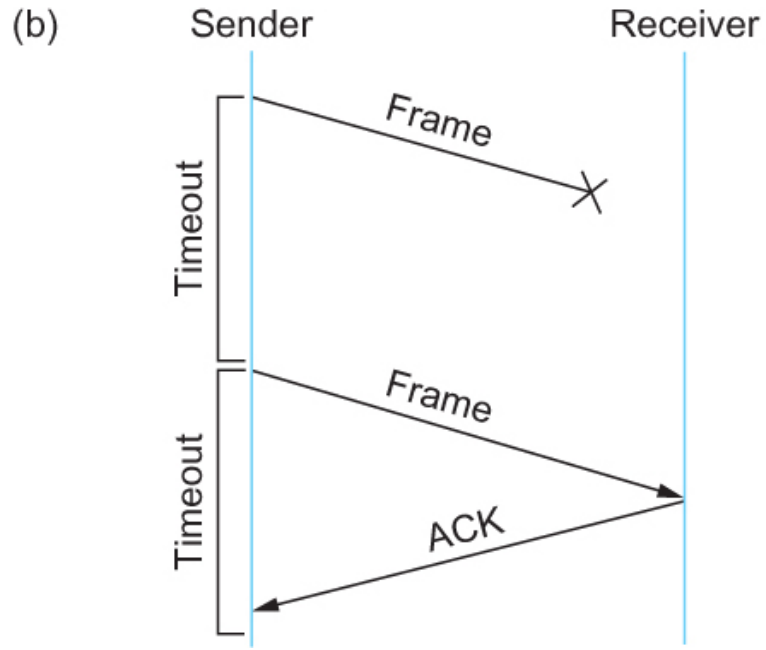


How long a timeout should we use?



Stop-and-wait: Lost frame

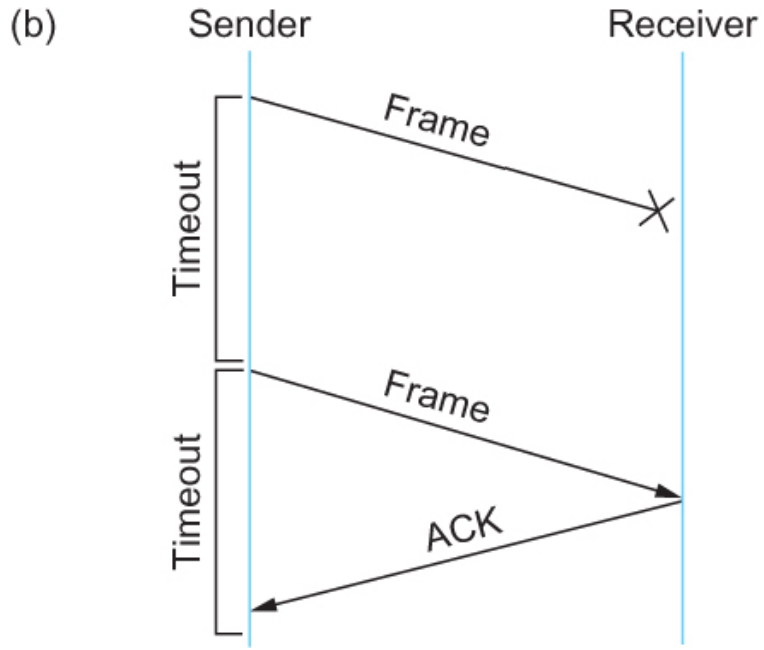
Sender eventually times out and resends the frame.



Receiver oblivious to lost frame.

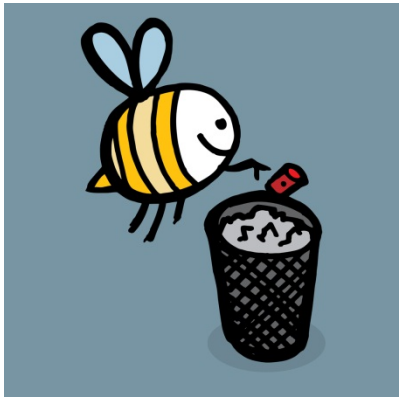
Stop-and-wait: Corrupt frame

Sender eventually times out and resends the frame.



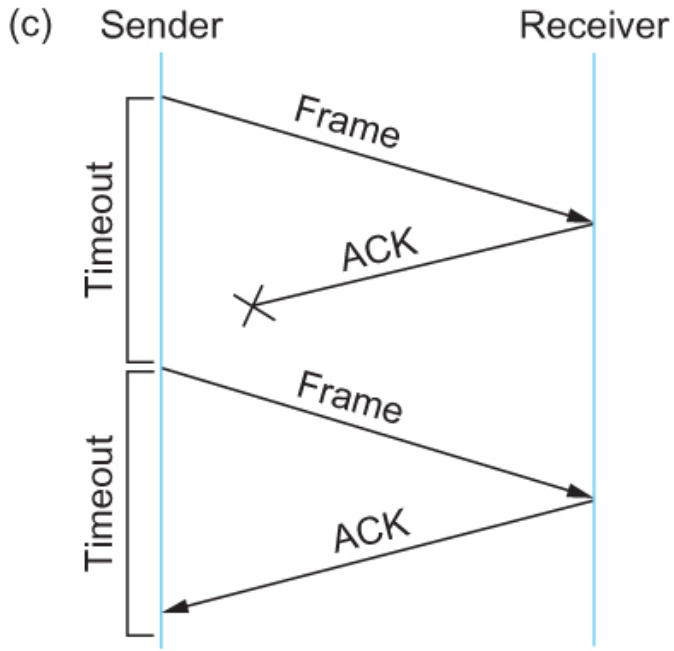
Receiver knows frame is corrupt (bad CRC or checksum)

Just waits for sender to timeout



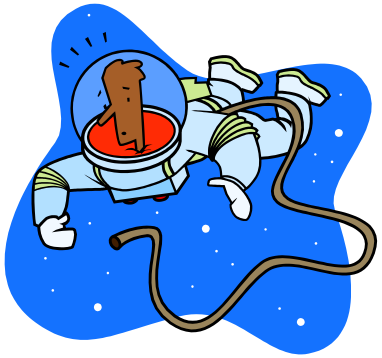
Stop-and-wait: Lost ACK

Sender never gets the first ACK. Eventually times out and resends that frame.



Receiver got frame and ACK'd it.

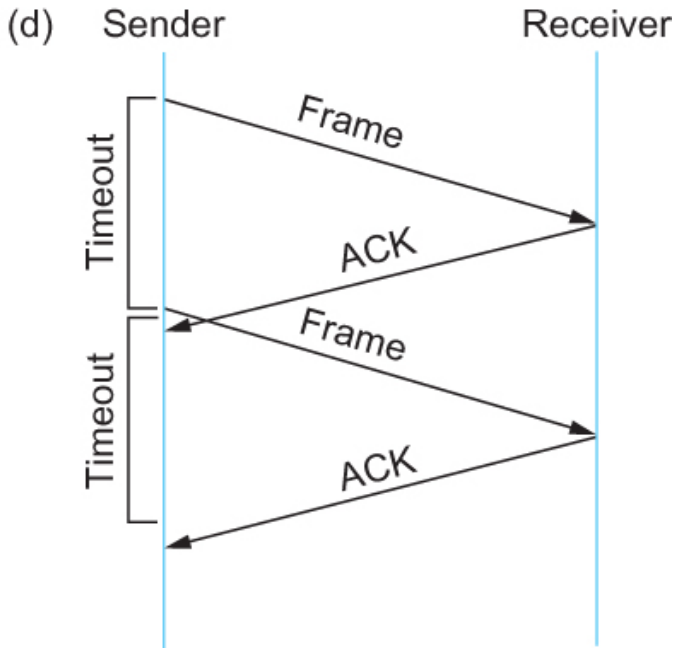
Get another frame, so ACK'd that as well.



Stop-and-wait: Delayed frame

Sender didn't get ACK before timeout, so resends the frame.

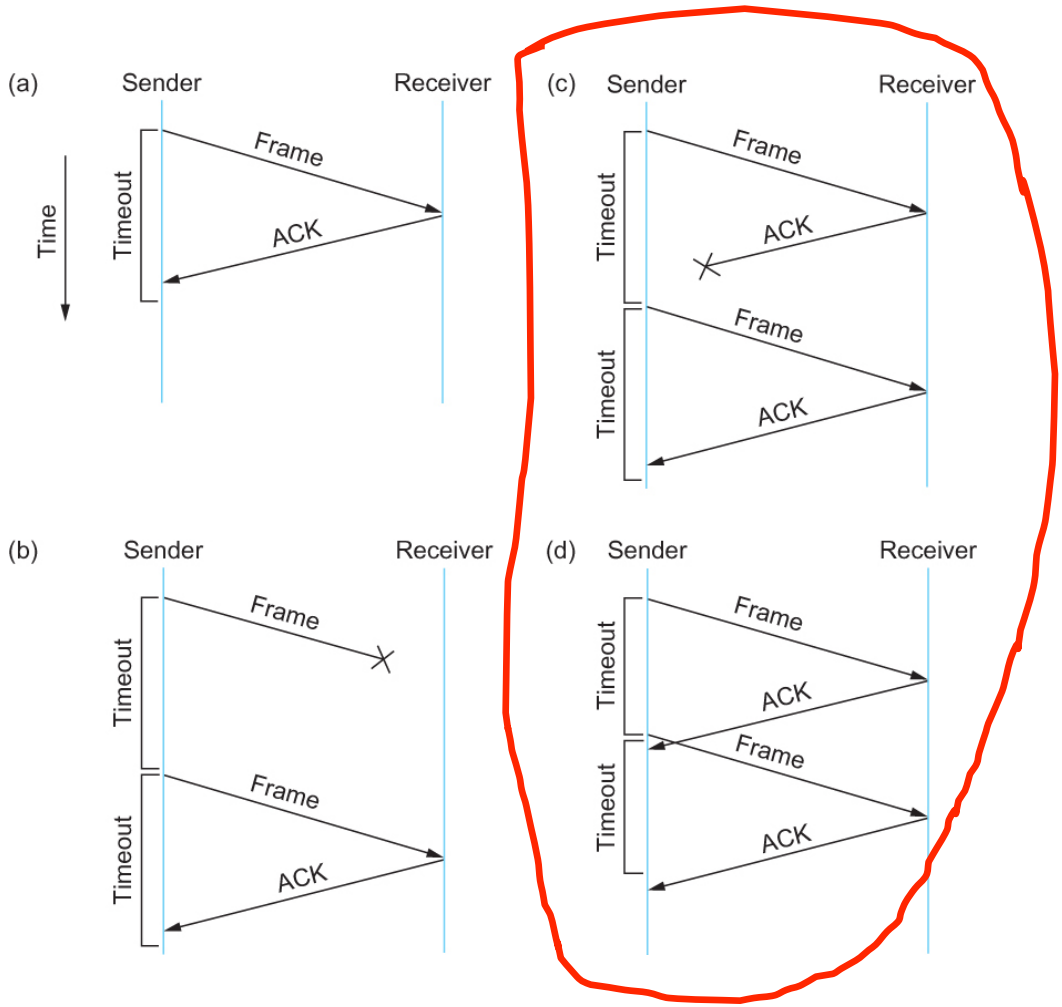
Sender gets duplicate ACKs.



Receiver got frame and ACK'd it.

Got another frame, so ACK'd that as well.

Stop-and-wait



Problem 1:
Receiver thinks the retransmission is a new frame, corrupting data passed up to network layer.

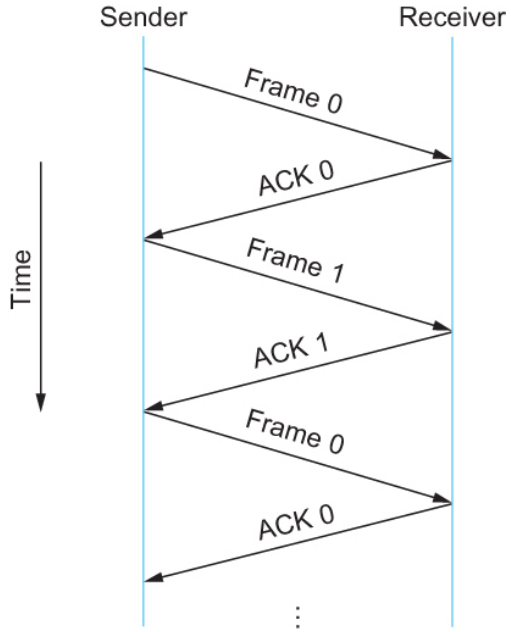
a) ACK received before timeout
b) original frame is lost
c) ACK is lost
d) timeout triggered too soon

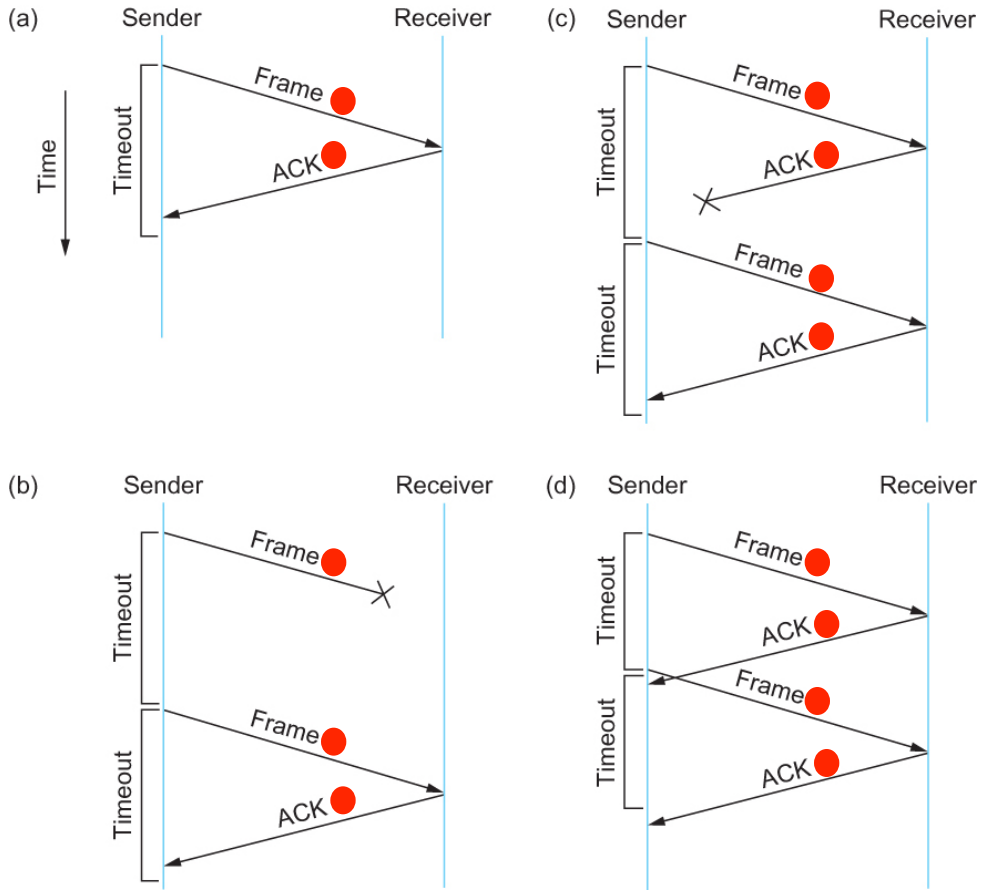
Stop-and-wait

Solution 1:

Use 1-bit sequence number.

Receiver can now determine if received frame is a duplicate.





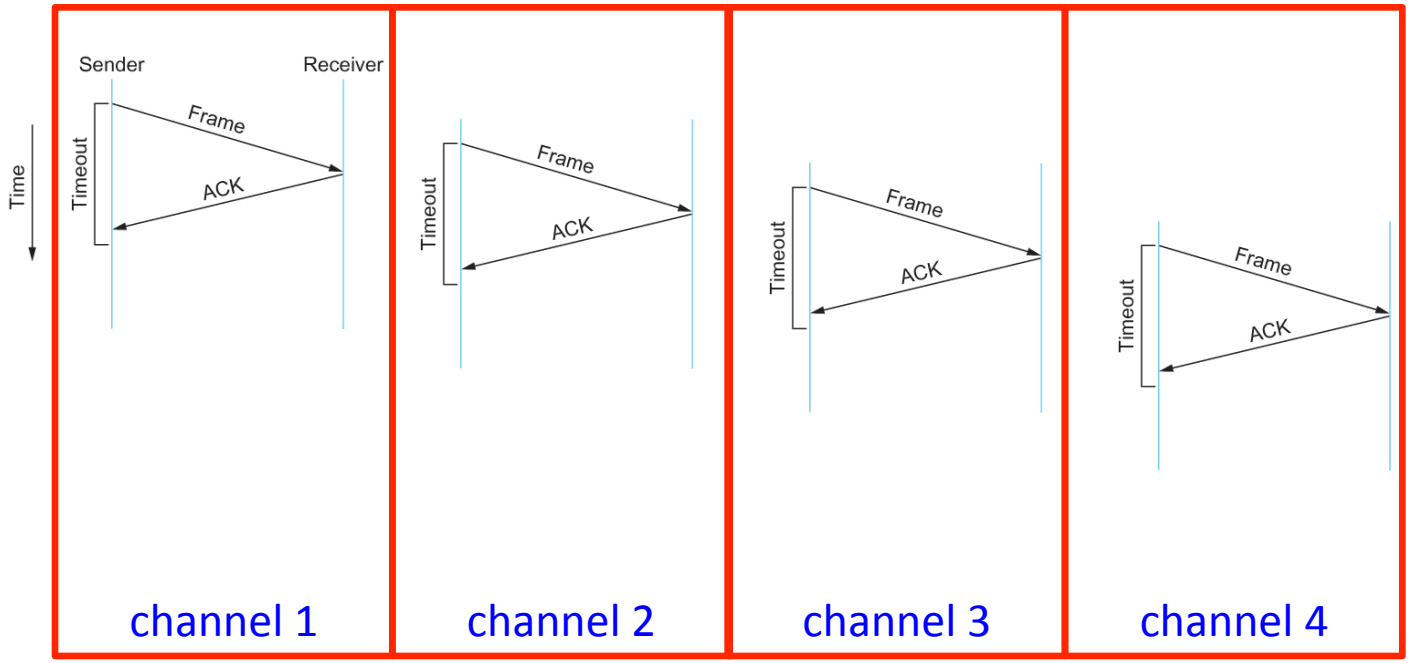
http://www.net-seal.net/animations.php?aid=37

Stop-and-wait

- Problem 2: Inefficient use of bandwidth
  - Only one frame in flight
  - Example 1:
    - 1.5 Mbps link, 45 ms RTT, 1K frame size
    - 1024 bytes x (8 bits / byte) / 0.045 s = 182 kbps
    - delay x bandwidth product:
      - » 0.045 s x 1.5 Mbps = 67500 bits x (1 byte / 8 bits) = 8.4K
  - Example 2:
    - 50 kbps satellite link, 500 ms RTT, 1K frame size
    - 1024 bytes x (8 bits / byte) / 0.500 s = 16.4 kbps
      - » 0.5 s x 50 kbps = 25000 bits x (1 byte / 8 bits) = 3.1K

Concurrent logical channels

- Concurrent logical channels
  - Allows more efficient use of bandwidth
  - Use stop-and-wait on multiple logical channels

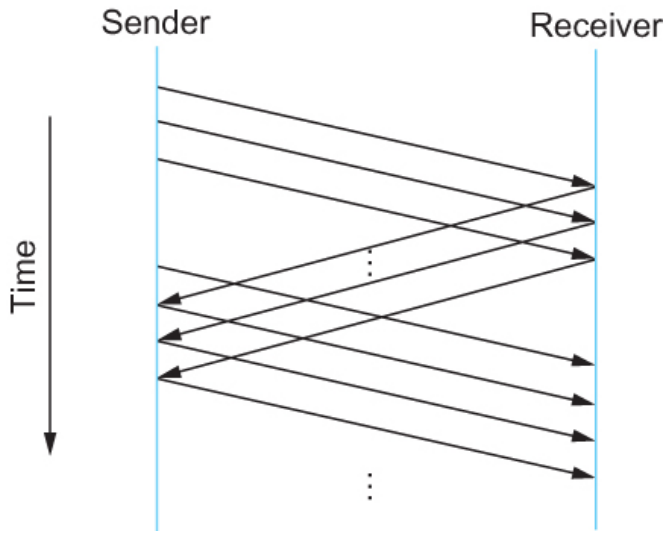


Concurrent logical channels

- Concurrent logical channels
  - Used in ARPANET
  - Different processes can be allocated different numbers of channels
  - Potentially can use full bandwidth
  - Problems:
    - A process might not fully utilize its channel
    - Splitting a process' communication across multiple channels may not maintain data ordering

Sliding window

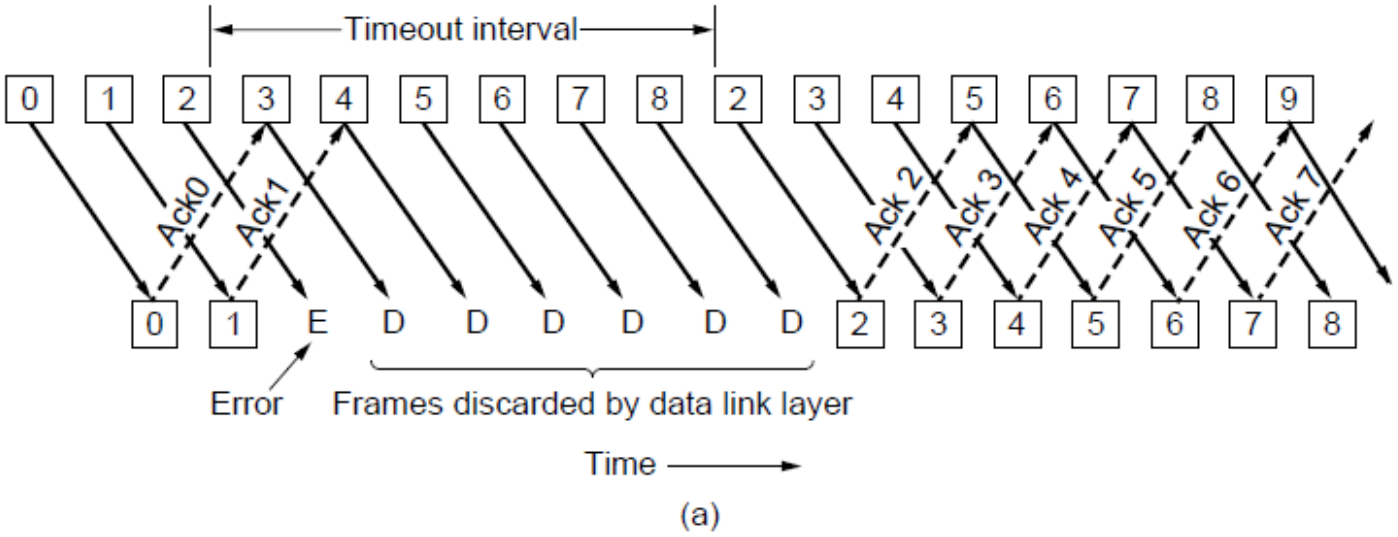
- Sliding window protocol
  - Better solution to bandwidth utilization problem
    - Put multiple frames in flight
    - Best known algorithm in networking
    - Several variations on this idea
    - Used in TCP



Sliding window: Go-back-n

- Sender:
  - Send window size
    - Sender can send this many frames without an ACK
    - Each frame has sequence number
  - Timeout: sends lowest unacknowledged frame and all subsequent frames
- Receiver:
  - Receive window size of 1
  - ACKs each good frame (or highest good frame)
  - Expects certain sequence number, drops all others

Go-back-n



- Problem:
  - Go-back-n wastes bandwidth re-sending frames that may have been received okay

http://www.eecis.udel.edu/~amer/450/TransportApplets/GBN/GBNindex.html
http://www.net-seal.net/animations.php?aid=38

Sliding window: Selective repeat

- Selective repeat
  - Sender:
    - Tracks which frames have been ACK'd
    - Unacknowledged frames must remain in buffer until acknowledged
    - Timer(s) track if frame needs resending
  - Receiver:
    - Hold out-of-order frames until in order section can be passed up to network layer

http://www.eecis.udel.edu/~amer/450/TransportApplets/SR/SRindex.html
http://www.net-seal.net/animations.php?aid=39

Other ARQ features

- Negative acknowledgement (NAK)
  - Receiver got the frame but error detected
  - Stimulates retransmission
    - Avoiding waiting for timeout
  - But adds complexity, timeouts can handle
- Piggybacking
  - Often two-way data exchange
  - Use ACK to both acknowledge and send data
  - Wait a bit hoping for data from network layer

Window sizes and sequence #'s

Algorithm	Send window size	Receive window size
Stop-and-wait	1	1
Go-back-n	N	1
Selective repeat (normally N=M)	N	M

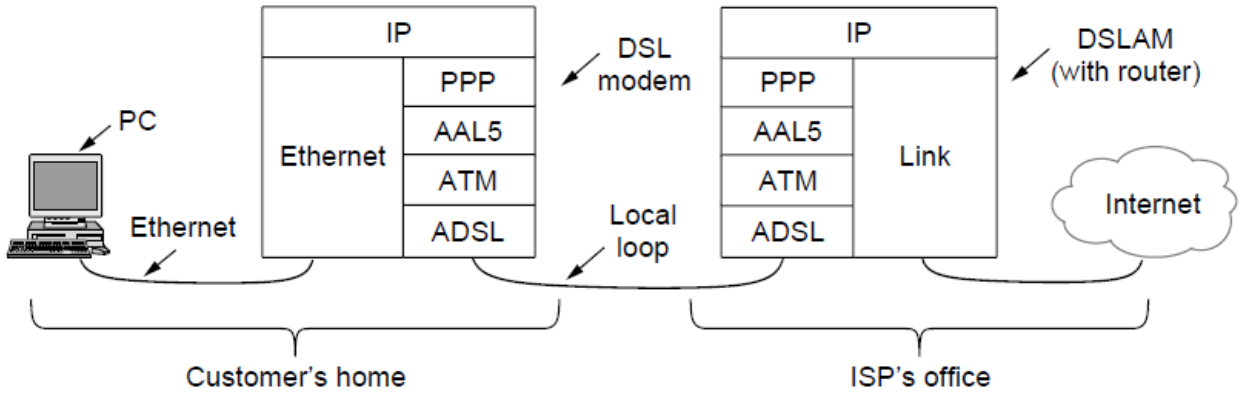
- Selective repeat
  - Normally window sizes same N=M
  - Sequence numbers have a max: MaxSeqNum
  - Send window size < (MaxSeqNum + 1) / 2

Data link protocols

- Computer networks built-up from point-to-point links
- Two example data link protocols:
  - Packet over SONET
    - Optical connectivity in WANs
    - Connect routers in different locations of an ISP
  - ADSL
    - Local loop of the telephone network
    - End-user last-mile connectivity

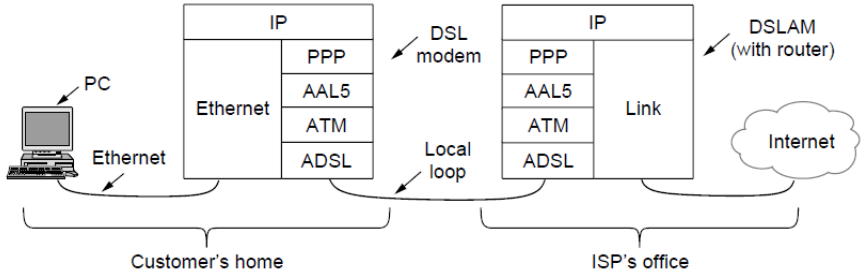
ADSL

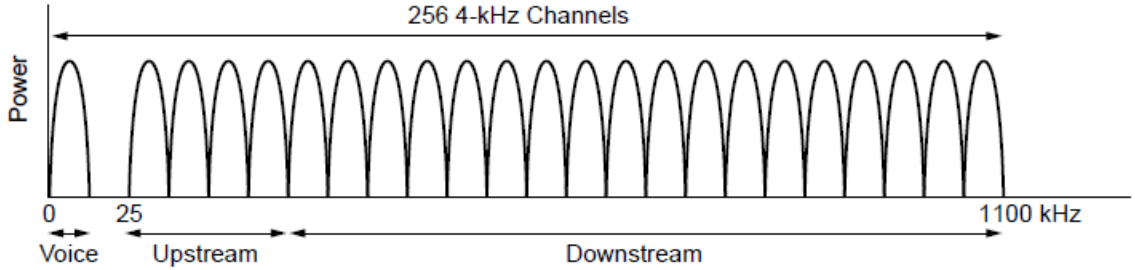
- Asymmetric digital subscriber loop (ADSL)
  - Last mile connectivity at Mbps speeds
  - Uses normal plain old telephone service (POTS)
    - Customer hooks DSL modem to phone line
    - Connects to "dee-slam" in the telephone local office

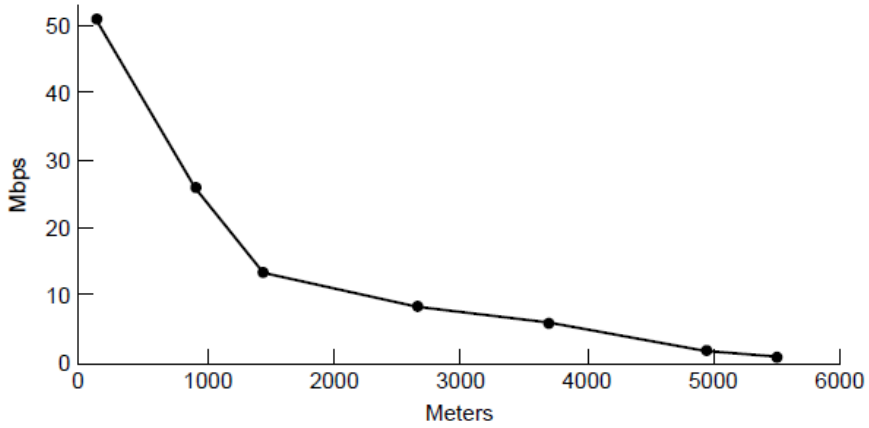


ADSL: Physical layer

- Signal modulated
  - Orthogonal frequency division multiplexing

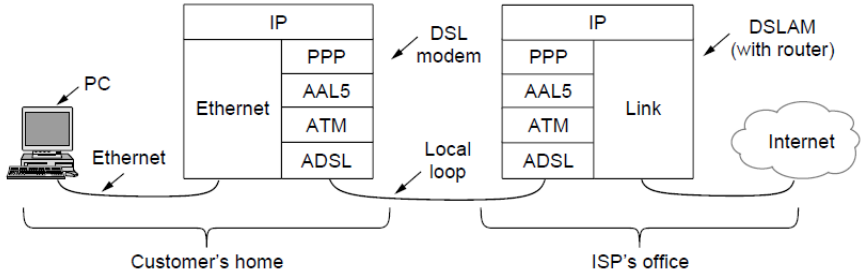






ADSL: Data link layer

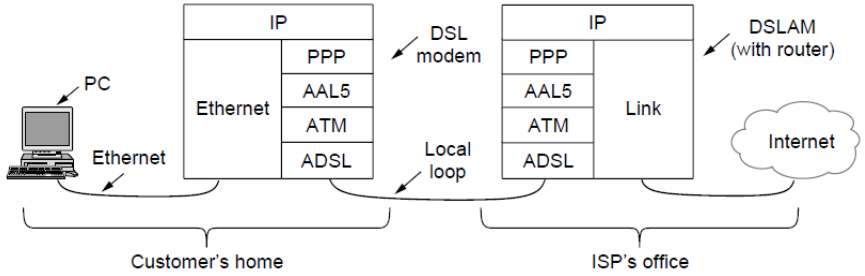
- ATM
  - Asynchronous Transfer Mode
  - Small fixed length cells
    - 53 bytes, 48 payload, 5 header
    - Europe wanted 32-bytes, US wanted 64-bytes
  - Asynchronous, cells not always sent (unlike SONET)



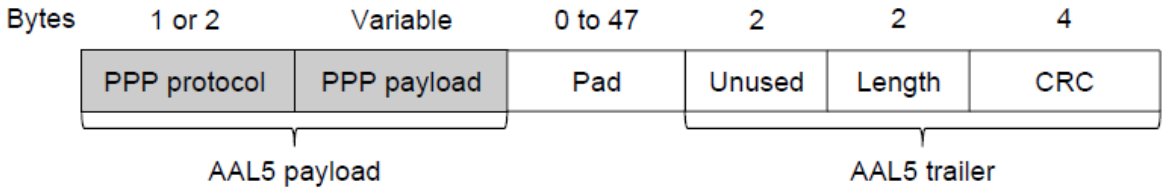
"ATM was…launched with incredible hype. It promised a network technology that would solve the world's telecommunications problems by merging voice, data, cable television, telegraph, carrier pigeon, tin cans connected by strings, tom toms, and everything else…"

-Andrew Tanenbaum, Computer Networks 5th edition

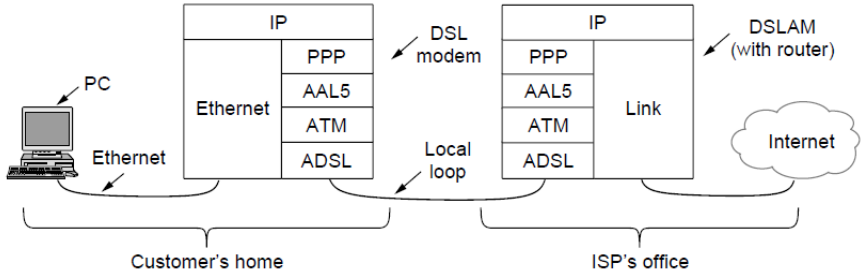
ADSL: Data link layer



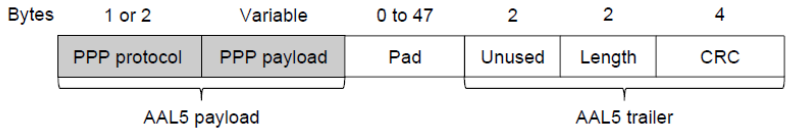
- AAL5
  - ATM Adaptation Layer 5
  - Map data into sequence of ATM cells
  - Pads out to 48 bytes
  - No address, each ATM has a virtual circuit ID
  - Frame format:



ADSL: Data link layer



- PPPoA
  - Point-to-point protocol over ATM
  - PPP protocol and payload placed in AAL5 payload
  - Protocol field:
    - I'm an IP packet
    - I'm a link control message (LCP)
  - PPP framing & CRC not needed
    - Already provided by ATM/AAL5



Summary

- Network needs:
  - Reliable delivery
  - In order delivery
  - Flow control
- Building blocks:
  - Acknowledgements (ACKs), timeouts
  - Algorithms: stop-and-wait, go-back-n, selective repeat
- Data link layer example: ADSL