Layering

- Layers: Application => Transport (TCP) => Routing (IP) => Data Link (Ethernet) => Physical
- Layer numbering starts at 1 for physical
- Protocols are the rules governing horizontal communication
- Interfaces are the rules governing vertical communication
- PDU => horizontal communication
    - TPDU (segment), NPDU (packet), DPDU (frame)
    - N-PDU is an N-SDU (Service Data Unit) together with a layer N header
    - An N-PDU is normally an N-SDU with a layer N header, usually the exact same as the SDU passed to layer N - 1
- Strict layering states that each layer should only look at its header and interface data to do its job

Physical Layer

- Physical Sublayers: transmission, media dependent, coding/decoding

Transmission Sublayer

- The two facets of channel distortion are inertia (lack of bandwidth) and noise
- The Nyquist limit states that we cannot send symbols faster than 2 * bandwidth w/o ISI
- The rate of sending symbols is the signaling/baud rate
    - The bit rate is the baud rate times # of bits/symbol
- Shannon limit tells us we cannot send faster than 2Blog(S/2N) bits/second
    - S = maximum signal strength, N = max noise amp

Clock Recovery/Coding Sublayer

- Compare codes with: guaranteed transitions, transmission efficiency, signal:noise ratio, DC balance, implementation complexity
- Phase locked loops use *many* transitions to generate a clock that is compared to the actual receiver clock => noise will not significantly impact sampling times

Media Sublayer

- Wired media require right of way to install
    - Fiber is light and easy to install and has excellent electrical noise immunity
- Broadcast media (coax/satellite) make possible cheap broadcast protocols, but are insecure
    - Get around need for infrastructure or right of way
- Microwaves are absorbed by rain and infrared is easily absorbed by obstacles
- Low power radio waves and infrared don't require wires or much power

	Bandwidth	Span	Disadv	Adv
Twisted Pair	< 1Mbps	1-2Km	low speed	cheap, easy to install
Digital Coax	10-100Mbps	1-2km	hard to tap, install	broadcast
Analog Coax	100-500Mbps	100km	exp. analog amplifiers	cable companies use it now!
Satellite	100-500Mbps	worldwide	prop delay antennas	no right-of-way cost does not depend on distance
Microwave	10-100Mbps	100 km	fog outages	no right-of-way
Fiber	terabits	100km	no broadcast no mobility	security isolation bandwidth
Infrared RF	< 4 Mbps 115 kbps	3 m 1 km	obstacles for infrared	wireless

Figure 12: Pros and Cons of Various Media: a summary

-

Data Link Layer

- 2 kinds of errors: random bit and burst
- Difference between point-to-point and broadcast links
- PTP: Bits => Framing => Error Detection => Error Recovery (O)
    - Error recovery is historical, when it made sense to transmit every hop
- BC: Bits => Framing => Error Detection => Media Access => Multiplexing (O)
- Accepts packets from the network layer, adds DL header => frame
- E2E argument: only worthwhile guarantees are E2E guarantees
    - DL acks are an optimization, transport acks are the only guarantee
    - Good bit error rate => unreliable DL, bad bit error rate => reliable DL
- Quasi-reliability: undetected error probability and frame loss probability
    - E2E argument ignored in practice => slows down performance

$$\text{Eff.} = \frac{\text{useful bits}}{\text{total bits}}$$

→ $$\text{Overhead} = \frac{(\text{Len(encoded)} - \text{Len(orig)})}{\text{Len(orig)}}$$

Framing

- Needed for timesharing + to create small, manageable units for error recovery/detection
- Flags/Bit Stuffing, Start Flags/Character Count, Start/End Flags from Physical Layer

↳ Stuffing: can't create end ~~flag~~ flag + can't create flag without stuffing

w/stuffing

0111100

$0111110\underbrace{0111}_{\text{new flag}}10$

010101 01

$\underbrace{0101010}_{\text{flag bad}}10101$

E(x) is odd-bit error => E(x) has odd # of terms
↳ E(1) = 1
↳ If E(x)/x+1 => E(1) = Y(x+1) = 0 <->

Error Detection

- A codeword with Hamming distance d can detect all d - 1 bit errors
  - A 2d + 1 distance code can *correct* up to d code errors
  - Designer must choose between correction and detection

Error Recovery

- Sent packets must be delivered to the receiver without duplication, loss, or misordering
- Assume: undetected error rate is negligible, FIFO physical layer, frames can be lost, arbitrary delay on physical links
- Stop and Wait:
  - ```
    Sender has a variable SN (for sender number) initially 0.
    Sender repeats the following loop:

    1) Accept a new packet from the higher layer if available and store it
       in Buffer B
    2) Transmit a frame (SN, B)
    3) If an error-free (ACK,R) frame is received and R is not equal to SN then
         SN = R
         Go to Step 1.
     Otherwise if the previous condition does not occur with an arbitrary
     timeout period,  go to Step 2 after the timeout period.
    ```
  - ```
    Receiver has a variable RN (for receiver number) initially 0.
    Receiver does the following code:
    When an error-free data frame (S, D) is received:
      If S = RN then
         Pass D to higher layer
         RN = RN + 1
      Send (Ack, RN)
    ```
  - There are only two possible consecutive sequences #s in any state
  - We can use a single bit => alternating bit
  - ```
    The channel from S to R is either empty or contains a frame M.
    The channel from R to S is either empty or contains a frame A.
    If a channel contains a frame, then the other channel is empty.
    ```
- In doing a proof, you can only use the assumption all invariants hold in the previous state => use invariants to rule out cases

Sliding Window

- Throughput => jobs per second, Latency => time to complete a job
- Transmission rate => rate at which the physical layer sends bits, propagation delay => time it takes a bit to arrive at the receiver
- Stop and wait limits us to 1 frame per round trip delay => problem for all links where transmission rate * propagation delay (bandwidth-delay product/pipe size is large relative to frame size
- Selective reject is important for large pipe sizes with a large chance of losing frames
- ```
  Sender code for Go-back N:

  Assume all counters are large integers that never wrap.
  The sender keeps a lower window L, initially 0.

  Send (s,m)  (* sender sends or resends s-th data packet *)
     The sender can send this frame if and only if:
        m corresponds to data packet number s given to sender by client AND
        L <= s <= L + w - 1 (* only transmit within current window *)

  Receive(r, Ack) (* sender absorbs acknowledgement *)
    On receipt, sender changes state as follows:
       L := R

  The receiver keeps an integer R which represents the next sequence number
  it expects, initially 0.

  Receive(s,m) (* receiver gets a  data frame *)
      On receipt, receiver changes state as follows:
         If s = R then (* next frame in sequence *)
           R := s + 1
           deliver data m to receiver client.

  Send(r, Ack) (* we an allow receiver to send an ack any time *)
         r must be equal to receiver number R at point ack is sent
         most implementations send an ack only when a data frame is received

  We assume that any unacknowledged frame in current window is periodically
  resent.  In particular, the lowest frame in the current window must be
  periodically sent to avoid deadlock.
  ```

  ```
  Sender code for selective-reject:

  Assume all counters are large integers that never wrap.
  The sender keeps a lower window L, initially 0 and a table that
  indicates which numbers have been acked (this can be optimized to store
  only a windows worth of such state).

  Send (s,m)  (* sender sends or resends s-th data packet
      The sender can send this frame if and only if:
         m corresponds to data packet number s given to sender by client AND
         L <= s <= L + w - 1 (* only transmit within current window *) AND
         s has not been acked.

  Receive(R, List, Ack) (* sender absorbs acknowledgement *)
     On receipt, sender changes state as follows:
        L = R
        Mark every number in List as being acked in table.

  The receiver keeps an integer R which represents the next sequence number
  it expects, initially 0.  The receiver also keeps a table that, for
  each sequence number, stores a bit indicating whether it has been received
  and a pointer to the data, if any, being buffered.  Once again, this table
  can be optimized to reduce the amount of storage to be proportional to
  a window size.

  Receive(s,m) (* receiver gets a  data frame *)
      On receipt, receiver changes state as follows:
         If s >= R then
           Store m in table at position s and set bit in position s
           While the bit at position R is     set do
              Deliver data at position R
              R = R + 1

  Send(R, List, Ack) (* we an allow receiver to
         R must be equal to receiver number R at point ack is sent
         List consists of numbers greater than L that have been received.
         most implementations send an ack only when a data frame is received

  We assume that any unacknowledged frame in current window is periodically
  resent.  In particular, the lowest frame in the current window must be
  ```
- Go back N requires a modulus of size w + 1, selective reject requires 2w
- Reliable restarts require either non-volatile memory, probabilistic protocols, or time limits

Scanned with CamScanner