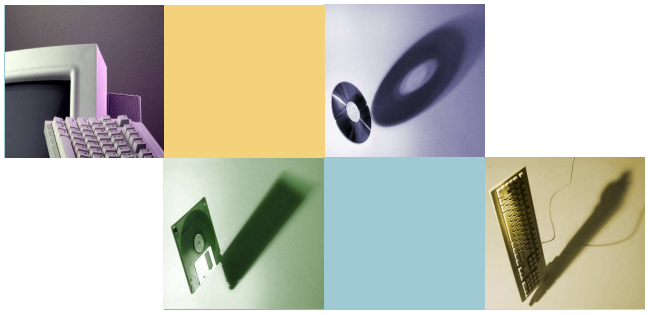
CSE123A discussion session

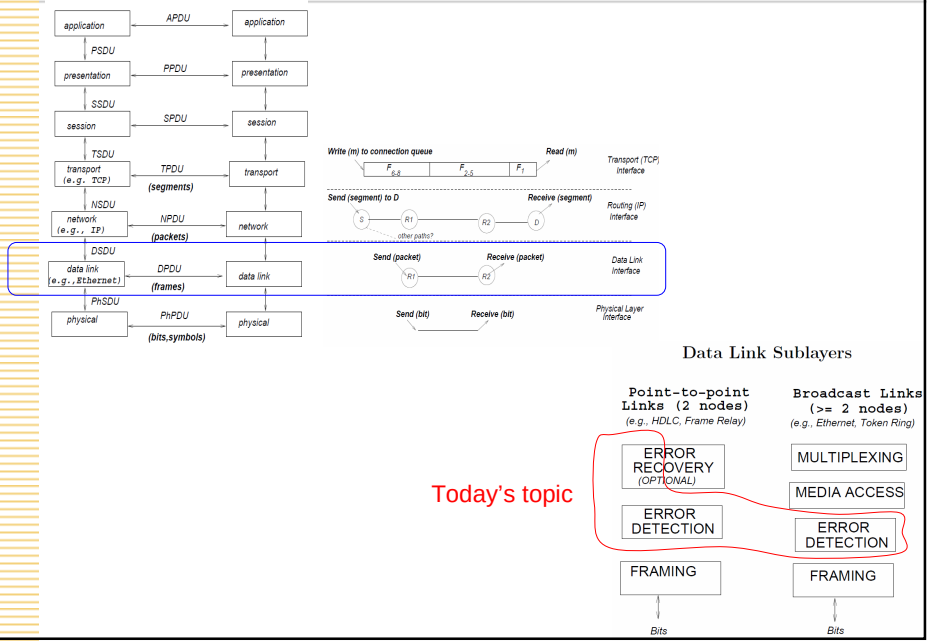


2007/02/09 Ryo Sugihara

Topics

- Review
  - Data Link Layer (3): Error detection sublayer
    - CRC
      - Polynomial representation
      - Implementation using LFSR
  - Data Link Layer (4): Error recovery sublayer
    - Protocol
      - Various errors
    - Simple protocol:
      - Stop and wait code
      - Alternating bit protocol
    - Throughput and latency

Where are we now?



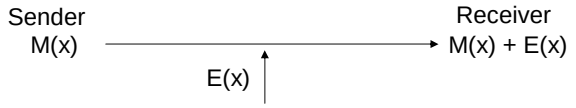
CRC (Cyclic Redundancy Check)

- Quick review
  - For detecting errors
    - Much more powerful than parity
  - Basic idea
    - Make all codeword divisible by G ("generator")
      - If a received bitstring is not divisible by G, it's corrupted
    - How? By subtracting the remainder of division
      - ex) msg=45, G=7: 45 % 7 = 3, so (45-3) is divisible by 7
      - In CRC, everything's done in "modulo-2 arithmetic", though

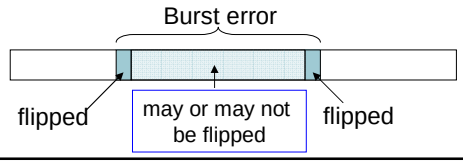
	data	Original message
data	CRC checksum	Codeword (to be sent to PHY layer)

CRC: Polynomial representation

- Example: “1011” = “x^3 + x + 1”
  - Addition is EXOR: x^2 + x^2 = (1+1) * x^2 = 0
  - Equivalent to arithmetic view
- What for?
  - For easier <u>analysis</u>
    - Now you see why remainder is always (r-1) bit when G is r bit
- What matters is “error polynomial”
  - Whether error polynomial E(x) is divisible by G(x) or not



- Burst error
  - Def:



CRC: Performance analysis (1/2)

Assume G(x) = x^r + ... + 1

- Random errors
  - 1bit:
    - E(x) = x^k
      - Not divided by any G(x) = x^r + ... + 1 (hint: at least two terms in G(x)(x^(k-r)+...)
  - 2bits:
    - E(x) = x^k (x^m + 1)
      - x^k is not divisible by G(x) (as seen above)
      - Whether x^m + 1 is divisible by G(x) or not is the problem
    - <u>G(x) should be designed NOT to divide x^m + 1 for large m</u>
  - Any odd number bit errors:
    - E(x) = x^k (x^m + ... + 1), where (x^m + ...) has odd number of terms
      - Never be divisible by “x + 1” (hint: substitute x = -1 to see if x^m + ... + 1 = 0 has x = -1 as a solution or not)
    - <u>G(x) should be designed to have “x + 1” as a factor</u>

CRC: Performance analysis (2/2)

Assume G(x) = x^r + ... + 1

- Burst errors
  - m bit (m-1<r):
    - E(x) = x^k (x^(m-1) + ... + 1)
      - x^k not divided by G(x)
      - x^(m-1) + ... + 1 not divided by G(x)
        - » Because m-1 < r
  - What about "m bit (m-1>=r)"?
    - x^(m-1) + ... + 1 (=Q(x)) might be divided by G(x)
    - How many expressions whose highest degree is (m-1)?
    - How many expressions can we make by "G(x) (x^(r-m-1) + ...)"?
    - Thus, <u>most errors can be detected</u>
      - prob of undetected err = 1/2^r

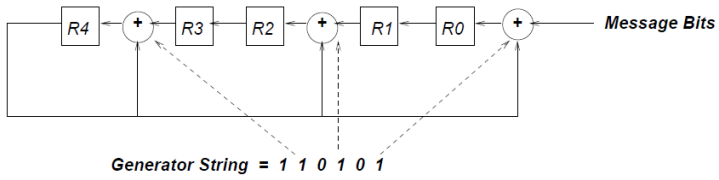
CRC: Implementation

- Software implementation
  - Emulate ordinary division
    - Hold current remainder in the register
    - If MSB=1, EXOR current remainder with divisor
    - Shift current remainder by 1bit to the left, and shift-in the next message bit
- Naive hardware implementation
  - Same as SW impl
    - Compare MSB
    - Calc EXOR
    - Shift 1bit
  - 3 clocks per each iteration

```
          110
1101)101000
     1101
      1110
      1101
       0110
```

- LFSR (Linear Feedback Shift Register)
  - Same thing, but faster
  - 1 clock per each iteration
    - Do everything simultaneously

CRC: LFSR



- 1 clock per each iteration
  - Compare MSB
  - Calc EXOR
  - Shift 1bit
- Parts
  - R0-R4: register to store 1 bit
  - Generator string is encoded in circuit
- Simple example: G=1101, M=101
  - How many registers?
  - How to connect?

CRC: LFSR example

G = 1101 M = 101

clock	R2	(XOR)	R1	R0	(XOR)	Message
0	(0)		(0)	(0)		101000
1	(0)		(0)	1		01000
2	(0)		1	0		1000
3	1		0	1		000
4	1		1	1		00
5	0		1	1		0
6	1		1	0		

Topics

- Review
  - Data Link Layer (3): Error detection sublayer
    - CRC
      - Polynomial representation
      - Implementation using LFSR
  - Data Link Layer (4): Error recovery sublayer
    - Protocol
      - Various errors
    - Simple protocol:
      - Stop and wait code
      - Alternating bit protocol
    - Throughput and latency

Error recovery sublayer

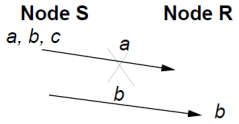
- Error recovery is OPTIONAL in Layer 2
  - Usually taught at Layer 4
    - Along with TCP
  - However, it's increasing importance
    - e.g.) wireless
- What must error recovery guarantee?
  - No duplication: a,<u>a</u>,b,c,d,...
  - No loss: a,c,d,...
  - No misordering: a,<u>c,b</u>,d,...
- Design "protocol" to realize error recovery
  - Protocol = "The <u>rules</u> governing horizontal communication between peer layer entities" (Remember lecture on layering)

Before starting protocol design…

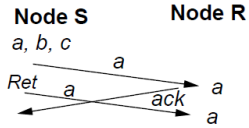
- Assumptions
  - Perfect error detection
    - No undetected error
  - Whole frame can be lost
    - Receiver may not know anything was sent
  - PHY layer is FIFO
    - Frame sent earlier arrives earlier
      - Not a valid assumption in TCP!
  - Arbitrary delay
    - May vary frame-to-frame

Protocol design (by trial-and-error)

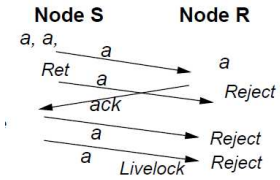
- Naive
  - Loss



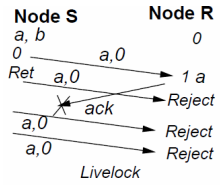
- + ACK & retransmit
  - Dup



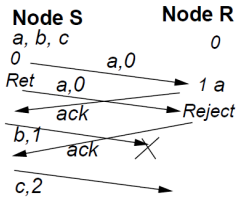
- + Reject duplicates
  - Livelock



- + Seq. number
  - Livelock



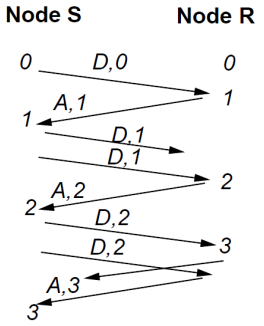
- + ACK for duplicate frames
  - Loss



- + Numbered ACK
  - → Stop and wait protocol

Stop and wait protocol

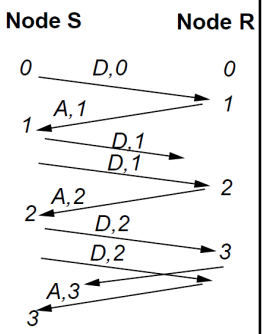
- Each of sender and receiver keeps a "counter"
  - SN (sender number), RN (receiver number)
  - Large enough (e.g. 32bit)
  - 0, initially



- Sender
  - send (D, SN)
  - update SN with R if it receives (ACK, R) where R != SN
    - R = SN+1, in this case
    - (Don't update SN when R=SN)
  - retransmit after timeout period
- Receiver
  - on receiving (D, S) where S = RN,
    - pass D to upper layer
    - update RN with RN+1 (=S+1)
  - send (ACK, RN)
    - in any case, even for dup frames

"Stop & wait" to "Alternating bit"

- Do we need 32bits for SN & RN?
  - No, 1bit would suffice
- Why?
  - Intuitively because
    - SN = RN or SN + 1 = RN
  - For precise reasons,
    - Use "invariants" (see lecture notes)

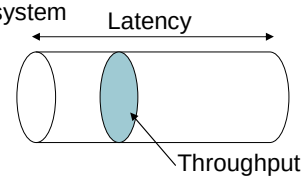


Throughput & Latency

- "Throughput"
  - "Jobs completed per second"
    - i.e. Num of jobs coming out of the system per sec
  - System owners want to maximize

Analogy: Pipeline (in architecture class)
- Throughput: Num of pipelines
  - cf. superscalar, VLIW
- Latency: Length of pipelines

- "Latency"
  - "Time to complete one job"
    - i.e. Time that one job stays inside the system
  - Users want to maximize



- Other performance metrics
  - "Propagation delay"
    - Time for <u>one bit</u> to reach the receiver
    - Different from transmission rate (cf. telephone line vs. satellite)
  - "Pipe size"
    - (transmission rate) * (round-trip propagation delay)
    - When large, you might want sliding window protocol

BACKUP