

Lecture 4: CRC & Reliable Transmission

CSE 123: Computer Networks
Chris Kanich



Quiz 1: Tuesday July 5th



Lecture 4: CRC & Reliable
Transmission

Lecture 4 Overview



- CRC – toward a better EDC
- Reliable Transmission
 - How do we ensure that a message was received?
 - Automatic Repeat Request (ARQ)
 - » Acknowledgements (ACKs) and timeouts
 - Stop-and-Wait
 - Sliding Window
 - Forward Error Correction

Lecture 4: CRC & Reliable Transmission

2

Checksum review



- Sum up all data in frame, Check when receiving
 - Transmit that sum as the EDC
- Extremely lightweight
 - Easy to compute fast in hardware
 - Fragile: Hamming Distance of 2
- Also easy to modify if frame is modified in flight
 - Happens a lot to packets on the Internet
- IP packets include a 1's compliment checksum

Lecture 4: CRC & Reliable Transmission

3

From Sums to Remainders



- Checksums are easy to compute, but very fragile
 - In particular, **burst** errors are frequently undetected
 - We'd rather have a scheme that "smears" parity
- Need to remain easy to implement in hardware
 - So far just shift registers and an XOR gate
- We'll stick to Modulo-2 arithmetic
 - Multiplication and division are XOR-based as well
 - Let's do some examples...

Modulo-2 Arithmetic



- Multiplication
- Division

$$\begin{array}{r} 1101 \\ 110 \\ \hline 0000 \\ 11010 \\ 110100 \\ \hline 101110 \end{array}$$

$$\begin{array}{r} 1101 \\ 110 \overline{) 101110} \\ \underline{110} \\ 111 \\ \underline{110} \\ 011 \\ \underline{000} \\ 110 \end{array}$$

Cyclic Remainder Check



- Idea is to *divide* the incoming data, D , rather than add
 - The divisor is called the **generator**, g
- We can make a CRC resilient to k -bit burst errors
 - Need a generator of $k+1$ bits
- Divide $2^k D$ by g to get remainder, r
 - Remainder is called **frame check sequence**
- Send $2^k D + r$
 - Note $2^k D$ is just D shifted left k bits
 - Remainder must be at most k bits
- Receiver checks that $(2^k D + r)/g = 0$

CRC: Rooted in Polynomials

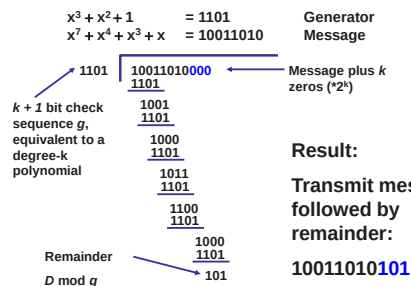


- We're *actually* doing polynomial arithmetic
 - Each bit is actually a coefficient of corresponding term in a k^{th} -degree polynomial

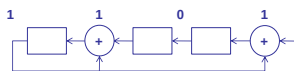
1101 is $(1 * X^3) + (1 * X^2) + (0 * X^1) + (1 * X^0)$

- Why do we care?
 - Can use the properties of finite fields to analyze effectiveness
 - Says any generator with two terms catches single bit errors

CRC Example Encoding



CRC in Hardware

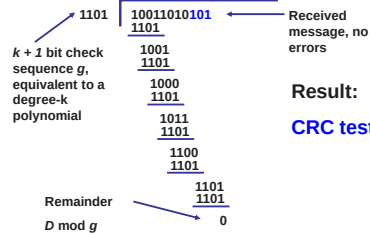


- Key observation is only subtract when MSB is one
 - Recall that subtraction is XOR
 - No explicit check for leading one by using as input to XOR
- Hardware cost very similar to checksum
 - We're only interested in remainder at the end
 - Only need k registers as remainder is only k bits

CRC Example Decoding



$$\begin{aligned} x^3 + x^2 + 1 &= 1101 && \text{Generator} \\ x^{10} + x^7 + x^6 + x^4 + x^2 + 1 &= 10011010101 && \text{Received Message} \end{aligned}$$



Result:
CRC test is passed

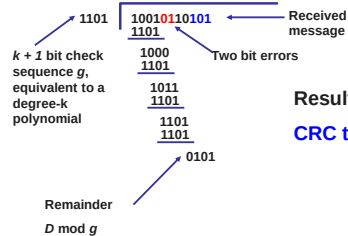
Lecture 4: CRC & Reliable Transmission

10

CRC Example Failure



$$\begin{aligned} x^3 + x^2 + 1 &= 1101 && \text{Generator} \\ x^{10} + x^7 + x^5 + x^4 + x^2 + 1 &= 10010110101 && \text{Received Message} \end{aligned}$$



Result:
CRC test failed

Lecture 4: CRC & Reliable Transmission

11

Common Generators



CRC-8	$x^8 + x^2 + x^1 + 1$
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x^1 + 1$
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x^1 + 1$
CRC-16	$x^{16} + x^{15} + x^2 + 1$
CRC-CCITT	$x^{16} + x^{12} + x^5 + 1$
CRC-32	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^4 + x^3 + 1$

Lecture 4: CRC & Reliable Transmission

12

Picking up the Pieces

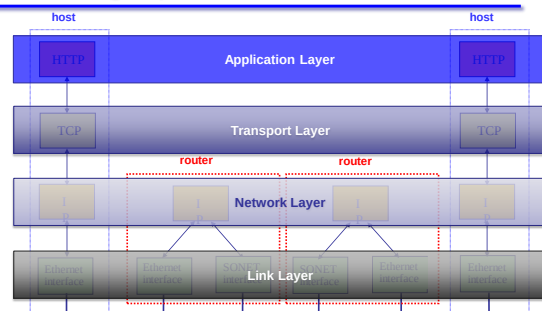


- Link layer is lossy
 - We deliberately throw away corrupt frames
 - Infrequent bit errors still lead to occasional frame errors
 - » 10,000+ bits in each frame
- Things get even harrier if we consider multiple links
 - In a few lectures, we'll start sending frames on long trips
 - Each intermediate stop might lose, corrupt, *reorder*, etc.
 - Regardless of cause, we'll call loss events **drops**
- We want to provide reliable, in-order delivery
 - Can—and will—do this at multiple layers

Lecture 4: CRC & Reliable Transmission

13

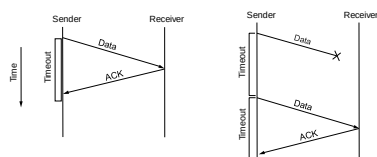
Moving up the Stack



Lecture 4: CRC & Reliable Transmission

14

Simple Idea: ARQ

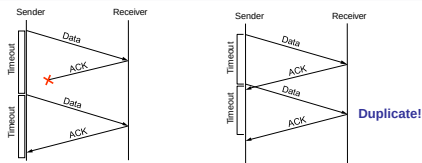


- Receiver sends **acknowledgments** (ACKs)
 - Sender "times out" and retransmits if it doesn't receive them
- Basic approach is generically referred to as **Automatic Repeat Request** (ARQ)

Lecture 4: CRC & Reliable Transmission

15

Not So Fast...

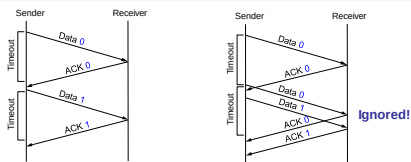


- Loss can occur on ACK channel as well
 - Sender cannot distinguish data loss from ACK loss
 - Sender will retransmit the data frame
- ACK loss—or early timeout—results in duplication
 - The receiver thinks the retransmission is new data

Lecture 4: CRC & Reliable Transmission

16

Sequence Numbers



- Sequence numbers solve this problem
 - Receiver can simply ignore duplicate data
 - But must still send an ACK! (Why?)
- Simplest ARQ: **Stop-and-wait**
 - Only one outstanding frame at a time

Lecture 4: CRC & Reliable Transmission

17

Stop-and-Wait Performance

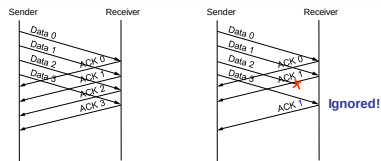


- Lousy performance if $x_{\text{mit}} 1 \text{ pkt} \ll \text{prop. delay}$
 - How bad?
- Want to utilize all available bandwidth
 - Need to keep more data "in flight"
 - How much? Remember the bandwidth-delay product?
- Also limited by quality of timeout (how long?)

Lecture 4: CRC & Reliable Transmission

18

Pipelined Transmission

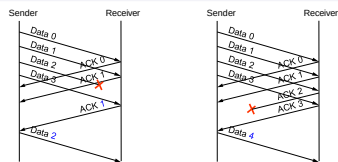


- Keep multiple packets “in flight”
 - Allows sender to make efficient use of the link
 - Sequence numbers ensure receiver can distinguish frames
- Duplicate acknowledgements signal loss
 - ACK the highest *consecutive* frame received
 - Ignore (for now) non-sequential frames

Lecture 4: CRC & Reliable Transmission

19

Go-Back-N

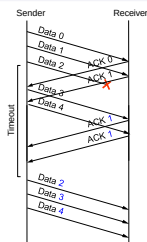


- Retransmit from point of loss upon **duplicate ACK**
 - Packets between loss event and retransmission are ignored
 - Also “go-back-N” if a timeout event occurs
- ACKs are **cumulative**
 - Acknowledge current frame and *all previous ones*

Lecture 4: CRC & Reliable Transmission

20

Send Window



Go-Back-N Example
with window size 3

- Bound on number of outstanding packets
 - Window “opens” upon receipt of *new* ACK
 - Window resets entirely upon a timeout
- Limits amount of waste
 - Still lots of duplicates
 - We can do better with **selective retransmission**

Lecture 4: CRC & Reliable Transmission

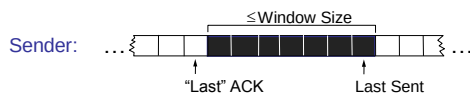
21

Sliding Window



- Single mechanism that supports:
 - Multiple outstanding packets
 - Reliable delivery
 - In-order delivery
 - Flow control
- At the core of all modern ARQ protocols
- Go-Back-N is a special case
 - Receive window size of one

Sliding Window – Sender



- Window bounds outstanding unACKed data
 - Implies need for buffering at sender
- “Last” ACK applies to in-order data
- What to do on a timeout?
 - Go-Back-N: send all unacknowledged data on timeout
 - Selective Repeat: timer per packet, resend as needed

Sliding Window – Receiver



- Receiver buffers too:
 - data may arrive out-of-order
 - or faster than can be consumed—**flow control**
- Receiver ACK choices:
 - Cumulative, Selective (exempt missing frames), Negative

Is ARQ the Only Way?



- No. We could use redundancy
 - Send additional data to compensate for lost packets
- Why not use retransmission?
 - Broadcast media with lots of receivers
 - » If each one ACK/NAK then hard to scale
 - Lots of messages
 - Lots of state
 - » Heterogeneous receivers
 - E.g., variable quality wireless reception
 - Highly lossy or very long delay channels (e.g., satellite)

Lecture 4: CRC & Reliable Transmission

28

Forward Error Correction

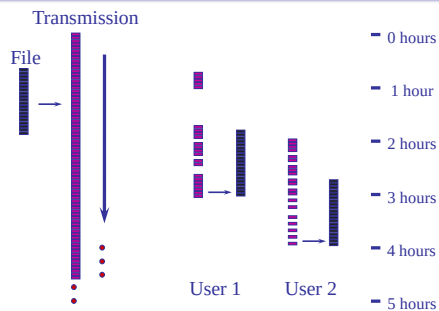


- Use **erasure codes** to redundantly encode k data frames into $m > k$ encoded frames
 - Why do it at the frame level?
 - E.g., Reed Solomon Codes, Tornado codes
- Multicast/broadcast encoded frames speculatively
- A receiver can reconstruct message from *any* k frames in the set of m encoded frames

Lecture 4: CRC & Reliable Transmission

29

A “Digital Fountain”



Lecture 4: CRC & Reliable Transmission

30

For Next Time



- Read 2.6 in P&D – Media Access
 - Now that we can reliably transmit from point A to point B, how do we share the transmission medium nicely?
- Have a great holiday weekend!
