CS2105 Lecture 4
Reliable Protocols

3 February, 2014
After this class, you are expected to:

- understand the interface between the transport layer and the network layer

- be able to design your own reliable protocols with ACK, NAK, sequence numbers, timeout, and retransmission.

- know how to calculate the utilization of a channel.

- understand the working of Go-Back-N and Selective Repeat protocols
``Sending Data Reliably Over the Internet is Harder Than You Think. The Intricacy Involved in Ensuring Reliability Will Make Your Head Explode."
Application
Transport
Network
Link
Physical
Transport layer resides on end hosts and provides process-to-process communication.
Network layer provides host-to-host, best-effort, unreliable communication.
How to build a reliable transfer protocol on top of unreliable communication?
Unreliable: may not deliver at all or deliver with error
Finite State Machine

A

B
rdt 1.0
Assume underlying channel is reliable
rdt 1.0 sender

wait for call
rdt 1.0 receiver

wait for call
rdt 2.0
Underlying channel may introduce bit errors
ARQ
Automated Repeat reQuest
Receiver detects errors
Receiver feeds back to sender
Sender retransmits
We will talk about checksum, the technique to detect errors, in more detail in the subsequent lectures.
rdt 2.0 sender

wait for call

wait for ack/nak
rdt 2.0 receiver

wait for call
Stop-and-Wait Protocol
Bug:
What if ACK/NAK is corrupted?
Bug Fix:
Add a sequence number
rdt 2.1 sender

- Wait for call 0
- Wait for ack / nak 0
rdt 2.1 receiver

wait for 0

wait for 1
rdt 2.2
Replace NAK with ACK of the last correctly received packet.
rdt 3.0
Packet can be loss or corrupted
Challenge: If ACK is lost, how to tell if the packet has been received?
Resend after timeout (may lead to dups, but OK since we have seq numbers).
rdt 3.0 sender

wait for call 0

wait for ack 0
The state diagram for the receiver in rdt 3.0 is given as an exercise.
Alternating Bit Protocol
RTT = 30ms
R = 1 Gbps
L = 1000 bytes
Pipelining:
need buffering and larger range of sequence numbers.
Go-Back-N
send a packet
receive ACK 3
receive ACK 5
window is full
window is empty
Selective Repeat
1. one timer per packet
2. receiver needs buffer
at sender:

receive ACK 5
at sender:

receive ACK 3
at receiver:
at receiver:

receive packet 5
at receiver:

receive packet 3
<table>
<thead>
<tr>
<th></th>
<th>GBN</th>
<th>SR</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>cumulative</td>
<td>selective</td>
</tr>
<tr>
<td>out-of-order</td>
<td>ignore</td>
<td>keep</td>
</tr>
<tr>
<td>retransmit</td>
<td>all unack</td>
<td>one unack</td>
</tr>
<tr>
<td>timer</td>
<td>earliest unack</td>
<td>one per unack</td>
</tr>
</tbody>
</table>
How to build a reliable protocol on top of unreliable communication?
How to build a reliable protocol on top of unreliable communication?

The same techniques can be used in other layers, such as in an application layer protocol over UDP, or in a link-layer protocol.
error detection
retransmission
timers
sequence numbers
ACK/NAK
window and pipelining