Performance of RDT3.0 and of stop-and-wait in general

RDT3.0 works, but performance stinks
e.g.: 1 Gb/s link, 15 ms e2e prop. delay, 1KB packet:

\[ T_{\text{transmit}} = \frac{L}{R} = \frac{8\text{kb/packet}}{10^{10} \text{ b/sec}} = 8 \text{ microsec} \]

\[ U_{\text{sender}} = \frac{L}{R \cdot L / R} = \frac{30.008}{0.0027} \]

U_{\text{sender}}: utilization – fraction of time sender busy sending
1KB packet every 30ms -> 33kB/sec throughput over 1 Gbps link
The protocol limits the use of physical resources!

(RTT + L / R) = L / R (packet length in bits)
R (transmission rate, bps) = 10^{10} b/sec

† Assuming negligible time to transmit/receive ack, or this is absorbed into RTT.

Outline

Pipelined protocols

Pipelining: sender allows multiple, “in-flight”, yet-to-be-acknowledged packets
- range of sequence numbers must be increased
- buffering at sender and/or receiver

• But if one frame gets errored, what happens to frames that don’t get errored that were transmitted after it, but before the error was detected? 2 approaches: Go-back-N and Selective repeat
• Multiple outstanding frames also introduces the possibility of sending too much for the receiver to buffer ⇒ flow control.
Types of acknowledgements

In addition to positive vs negative acknowledgements discussed earlier:

Cumulative acknowledgments: Indicate that everything up to, and including, a certain point has been received.
If receiver relays something beyond that point, it may either send nothing or resend a cumulative ack, indicating the point to which it has received all information (e.g., TCP sends "dupacks" which, when repeated, act as a nack).

Selective acknowledgments: Indicate that a specific frame/info has been received. Other info transmitted earlier may not have been received.

Numerical value of ack:
- Theory: Sequence number matches that of received frame.
- Practice: (e.g., HDLC variants and TCP) Sequence number is that of the next data unit expected.

Unit of ack: Frames or bytes?
- Frames (acked by link layer protocols) impose less load on sequence number space.
- Bytes (acked by TCP) allow different units of transmission and retransmission.

Go-Back-N

Sender:
- k-bit seq # in packet header
- "window" of up to N consecutive unacked packets allowed
- ACK(n): ACKs all packets up to, including seq # n - "cumulative ACK?"
- timer for each in-flight packet
- timeout(n): retransmit packet n and all higher seq # packets in window

In the context of TCP, we’ll see what info can be gleaned from duplicate acks.
**GBN: sender extended FSM**

- `udt_send(sndpkt[base])`
- `udt_send(sndpkt[base+1])`
- `…`
- `udt_send(sndpkt[nextseqnum-1])`

**GBN: receiver extended FSM**

ACK-only: always send ACK for correctly-received packet with highest in-order seq #
- may generate duplicate ACKs
- need only remember `expectedseqnum`
- out-of-order packet:
  - discard (don’t buffer) -> no receiver buffering!
  - Re-ACK packet with highest in-order seq #

**Outline**
Selective Repeat

Receiver individually acknowledges all correctly received packets
- buffers packets, as needed, for eventual in-order delivery to upper layer
Sender only resends packets for which ACK not received
- sender timer for each unACKed packet
Sender window
- $N$ consecutive seq #’s
- again limits seq #’s of sent, unACKed packets

Strictly, it need only be able to ack individual packets. When consecutive packets are received, it may be more efficient to ack these as a group.
Send individual acks for packets received before predecessors.
Sliding windows (Circular)

Sender

Receiver

T Fig. 3-13

Initially

After sending first frame

After receiving first frame

After sender receives ack for first frame

Shaded = info transmitted but not yet acked

Shaded = info willing to accept (enough buffer space)

Selective repeat: dilemma

Example:
- seq #s: 0, 1, 2, 3
- window size=3
  - receiver sees no difference in two scenarios!
  - incorrectly passes duplicate data as new in (a)

Q: what relationship between seq # size and window size?
A: window \( \leq \frac{(\text{MAX SEQ}+1)}{2} \)

Duplex data transfer

The protocols examined so far provide unidirectional data transfer:

Sender

Receiver

\( \uparrow \) = header, e.g. framing & integrity check

To provide duplex data transfer, can use:

a. 2 simplex protocols operating independently
b. Share channels & extend header to differentiate frame types
c. Piggyback backward control with any forward data

Piggybacking

Advantages:
- Save bandwidth:
  - No framing & integrity check overheads for ACK – just another field added to the payload frame.
  - Saving is more pronounced at higher layers, where packets may have other overheads, e.g. large addresses and control fields
- Fewer frames to process:
  - Fewer interrupts at end-systems
  - Lower packet processing overhead in routers (many routing functions, e.g. looking up address in routing tables, are independent of packet length)

Disadvantages:
- Higher chance of ack being lost since it’s carried in a longer packet.
  - May want separate header and payload error checks so that incorrect frame can process header when payload is corrupt.
- ACKs may be delayed while waiting for data to piggyback on
- Source must factor this into its timeout
- Receiver must run a timer and send ack alone if data is not forthcoming.

Source of image unknown
Lecture summary

- Stop-and-wait performs poorly when packet length \( < \) bandwidth-delay product of link.
- Sliding window protocols enable pipelining: Multiple packets can be propagating to the receiver (or acks from receiver) at any instant.
- After detecting loss:
  - Go-back-N retransmits from first lost packet onwards. Receiver need not resequence information.
  - Selective repeat retransmits only the lost packet(s). Saves bandwidth.
- Implementation issues:
  - Bounded sequence numbers handled by circular rotation of window.
  - Piggybacking acknowledgements with data saves bandwidth but costs delay.

Chapter 5
Data Link Layer

A note on the use of these ppt slides:
We're making these slides freely available to all (faculty, students, readers). They're in powerpoint form. You can add, modify, and delete slides.

All material copyright 1996-2002 J.F Kurose and K.W. Ross, All Rights Reserved.