Error and Flow Control

Required reading:
Garcia  5.2

CSE 3213,  Fall 2010
Instructor: N. Vlajic
Error Control

Error Control Approaches

(1) **Forward Error Correction (FEC)**

(2) **Error Detection + Automatic Retrans. Req. (ARQ)**

- not enough redundant info to enable error correction
  
  case (a) receiver detects no errors
  - an ACK packet is sent back to sender
  
  case (b) receiver detects errors
  - no ACK sent back to sender
  - sender retransmits frame after a 'time-out'
Challenges of ARQ-based Error Control

- **send one frame at the time, wait for ACK**
  - easy to implement, but inefficient in terms of channel usage

- **send multiple frames at once**
  - better channel usage, but more complex to implement - sender must keep (all) sent but unACKed frame(s) in a buffer, as such frame(s) may have to be retransmitted

How many frames should be sent at any point in time?
How should frames be released from the sending buffer?
Flow Control – set of procedures used to restrict the amount of data that sender can send while waiting for acknowledgment

- two main strategies
  1. **Stop-and-Wait**: sender waits until it receives ACK before sending next frame
  2. **Sliding Window**: sender can send W frames before waiting for ACKs

**Error + Flow Control Techniques**

1. Stop-and-Wait ARQ
2. Go-Back-N ARQ
3. Selective Repeat ARQ

Error Detection + ARQ (error detection with retransmissions) must be combined with methods that intelligently limit the number of ‘outstanding’ (unACKed) frames.

Fewer unACKed frames ⇒ fewer packets buffered at sender and receiver.
(1) Stop-and-Wait ARQ
Stop-and-Wait ARQ – simplest flow and error control mechanism

• sender sends an information frame to receiver
• sender, then, stops and waits for an ACK
• if no ACK arrives within time-out, sender will resend the frame, and again stop and wait
  ▪ time-out period > roundtrip time

• abnormalities (and how to fix them)
  ▪ lost acknowledgment
  ▪ delayed acknowledgment
Lost Acknowledgment

- frame received correctly, but ACK undergoes errors / loss
  - after time-out period, sender resends frame
  - receiver receives the same frame twice
- frames must be numbered so that receiver can recognize and discard duplicate frames
  - sequence # are included in packet header

Without packet numbering

How will receiver know that this is NOT a new packet?!

With packet numbering

Receiver has already received frame 2 - it resends an ACK and discards the duplicate.
Delayed Acknowledgment (Premature Timeout)

- ACKs can be delayed due to problems with links or network congestion
  - time-out expires early, sender resends frame
  - when delayed ACK arrives, sender assumes that given ACK is for the last frame sent

- ACKs must be numbered to prevent gaps in delivered packet sequence

How large should the packet / ACK sequence be?  Only 1-bit long !!!
Stop-and-Wait ARQ (cont.)

**Stop-and-Wait Efficiency**

- \( t_0 = \text{basic Stop-and-Wait delay} \) – from time when frame is transmitted into channel until time when ACK arrives back to receiver, and another frame can be sent

\[
t_0 = 2 \cdot t_{\text{prop}} + 2 \cdot t_{\text{proc}} + t_{\text{frame}} + t_{\text{ACK}} = 2 \cdot t_{\text{prop}} + 2 \cdot t_{\text{proc}} + \frac{n_f}{R} + \frac{n_{\text{ACK}}}{R}
\]

- \( R_{\text{eff}} = \text{effective transmission (data) rate:} \)

\[
R_{\text{eff}} = \frac{\text{number of info bits delivered to destination}}{\text{total time required to deliver info bits}} = \frac{n_f - n_{\text{header}}}{t_0}
\]
• η_{SW} = transmission efficiency: ratio of actual and effective transmission (data) rate - ideally, η_{SW} \approx 1

- where do we lose channel efficiency, and how can η_{SW} \rightarrow 1 be achieved？!

\[
η_{SW} = \frac{R_{\text{eff}}}{R} = \frac{n_f - n_{\text{header}}}{t_0} = 1 - \frac{n_{\text{header}}}{n_f} + \frac{n_{\text{ACK}}}{n_f} + \frac{2(t_{\text{prop}} + t_{\text{proc}})R}{n_f}
\]

(1) \(\frac{n_{\text{header}}}{n_f}\) - loss in efficiency due to (need for) header

(2) \(\frac{n_{\text{ACK}}}{n_f}\) - loss in efficiency due to (need for) ACKs

(3) \(2(t_{\text{prop}} + t_{\text{proc}})R\) - bandwidth-delay product

- max number of bits in transit at any given time

- in Stop-and-Wait ARQ delay-bandwidth product is a measure of lost opportunity in terms of transmitted bits
Stop-and-Wait ARQ (cont.)

Bandwidth-delay product = 2*(t_{prop} + t_{proc})*R =
= capacity of the transmission pipe from the sender to the receiver and back.
Stop-and-Wait ARQ becomes inadequate when data is fragmented into small frames, such that \( \frac{n_f}{R} = t_{frame} \) is small relative to \( t_{prop} \).
Stop-and-Wait ARQ (cont.)

**Example** [impact of delay-bandwidth product]

\[ n_f = 1250 \text{ bytes} = 10000 \text{ bits} \]
\[ n_{ACK} = n_{header} = 25 \text{ bytes} = 200 \text{ bits} \]

\[ \frac{n_{ACK}}{n_f} = \frac{n_{header}}{n_f} = 0.02 \]

\[ \eta_{SW} = \frac{R_{eff}}{R} = 1 - \frac{n_{header}}{n_f} \times \frac{1 + \frac{n_{ACK}}{n_f} + \frac{2 \cdot (t_{prop} + t_{proc})}{n_f} R}{1.02 + \frac{2 \cdot (t_{prop} + t_{proc})}{n_f} R} = \frac{0.98}{10^3} + \frac{0.98}{10^4} + \frac{0.98}{10^5} + \frac{0.98}{10^6} \]

<table>
<thead>
<tr>
<th>Efficiency</th>
<th>200 km (t_{prop} = 1 \text{ ms})</th>
<th>2000 km (t_{prop} = 10 \text{ ms})</th>
<th>20000 km (t_{prop} = 100 \text{ ms})</th>
<th>200000 km (t_{prop} = 1 \text{ sec})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Mbps</td>
<td>(10^3) 88%</td>
<td>(10^4) 49%</td>
<td>(10^5) 9%</td>
<td>(10^6) 1%</td>
</tr>
<tr>
<td>1 Gbps</td>
<td>(10^6) 1%</td>
<td>(10^7) 0.1%</td>
<td>(10^8) 0.01%</td>
<td>(10^9) 0.001%</td>
</tr>
</tbody>
</table>

Stop-and-Wait does NOT work well for very high speeds or long propagation delays.
Stop-and-Wait Efficiency in Channel with Errors

- $P_f = \text{probability that transmitted frame has errors and need to be retransmitted}$
  - $(1-P_f)$ – probability of successful transmission
  - $\frac{1}{1-P_f}$ – average # of (re)transmission until first correct arrival
  - total delay per frame: $t_0 \cdot (\text{average # of retrans.}) = t_0 \cdot \frac{1}{1-P_f}$

$$\eta_{SW\_error} = \frac{R_{eff\_error}}{R} = \frac{n_f - n_{header}}{R} = \frac{t_0}{(1-P_f)} = (1-P_f) \cdot \frac{1}{1 + \frac{n_{ACK}}{n_f} + \frac{2(t_{prop} + t_{proc})R}{n_f}}$$

$\eta_{SW\_error} = (1-P_f) \cdot \eta_0$

$P_f$ increases $\Rightarrow \eta_{SW}$ decreases
Stop-and-Wait ARQ (cont.)

Probability that $i$ transmission are needed to deliver frame successfully (i-1 transmission in error and the $i$th transmission is error free):

$$P[ \text{# of trans. in error} = i-1 ] = (1-P_f) P_f^{i-1}$$

$$E[\text{# of transmissions in error}] = \sum_{i=1}^{\infty} (i-1) \cdot P[n_{\text{trans in error}} = i-1] = \sum_{i=1}^{\infty} (i-1) \cdot (1-P_f)P_f^{i-1} =$$

$$= (1-P_f) \cdot \sum_{i=1}^{\infty} (i-1) \cdot P_f^{i-1} = (1-P_f) \cdot \sum_{n=1}^{\infty} n \cdot P_f^n =$$

$$= (1-P_f) \cdot P_f \cdot \sum_{n=1}^{\infty} n \cdot P_f^{n-1} = (1-P_f) \cdot P_f \cdot \frac{1}{(1-P_f)^2} =$$

$$= \frac{P_f}{1-P_f}$$

Total average delay per frame:

$$t_0 + \text{time-out} \cdot E[\text{# of transmiss in error}] = t_0 + \text{time-out} \cdot \frac{P_f}{1-P_f} \approx \frac{1}{1-P_f} t_0$$
Stop-and-Wait ARQ (cont.)

**Piggybacking**

- Stop-and-Wait discussed so far was ‘unidirectional’
- In ‘bidirectional’ communications, both parties send & acknowledge data, i.e. both parties implement flow control
- **piggybacking method**: outstanding ACKs are placed in the header of information frames
- Piggybacking can save bandwidth since the overhead from a data frame and an ACK frame (addresses, CRC, etc) can be combined into just one frame
(2) Go-Back-N ARQ
Go-Back-N ARQ

Go-Back-N ARQ – overcomes inefficiency of Stop-and-Wait ARQ – sender continues sending enough frames to keep channel busy while waiting for ACKs

- a window of $W_s$ outstanding frames is allowed
- m-bit sequence numbers are used for both frames and ACKs, and $W_s = 2^{m-1}$

Assume: $W_s = 4$
1) **sender** sends frames one by one
2) frame 3 undergoes transmission error – **receiver** ignores frame 3 and all subsequent frames
3) **sender** eventually reaches max number of outstanding frames, and takes following action:
   - go back $N=W_s$ frames and retransmit all frames from 3 onwards
Go-Back-N ARQ (cont.)

**Sender Sliding Window**

- all frames are stored in a buffer, outstanding frames are enclosed in a window
  - frames to the left of the window are already ACKed and can be purged
  - frames to the right of the window cannot be sent until the window slides over them
  - whenever a new ACK arrives, the window slides to include new unsent frames
  - once the window gets full (max # of outstanding frames is reached), entire window gets resent

**Receiver Sliding Window**

- the size of receiver window is always 1
  - receiver is always looking for a specific frame to arrive in a specific order
  - any frame arriving out of order is discarded and needs to be resent

The complexity of the receiver in Go-Back-N is the same as that of Stop-and-Wait!!!

Only the complexity of the transmitter increases.
Go-Back-N ARQ (cont.)

Problems with Go-Back-N (Go-Back-N with Timeout)

- Go-Back-N works correctly (retransmission of damaged frames gets triggered) as long as the sender has an unlimited supply of packets that need to be transmitted
  - but, in case when packets arrive sporadically, there may not be $W_s - 1$ subsequent transmissions $\Rightarrow$ window will not be exhausted, retransmissions will not be triggered
  - this problem can be resolved by modifying Go-Back-N such that:
    1) set a timer for each sent frame
    2) resend all outstanding frames either when window gets full or when the timer of first frame expires
Example  [ lost frame in Go-Back-N with time-out ]

Note:

- **ACKs number always defines the number of the next expected frame !!!**
- in Go-Back-N, receiver does not have to acknowledge each frame received – it can send one *cumulative ACK* for several frames
Go-Back-N ARQ (cont.)

Sequence Numbers and Window Size

- $m$ bits allotted within a header for seq. numbers $\Rightarrow$ $2^m$ possible sequence numbers
  - how big should the sender window be!?
  - $W > 2^m$ cannot be accepted – multiple frames with same seq. # in the window $\Rightarrow$ ambiguous ACKs
  - $W = 2^m$ can still cause some ambiguity – see below
  - $W = 2^m - 1$ acceptable !!!

![Diagram of Go-Back-N ARQ](image)

- Window size $2^m = 4$
  - Correctly accepted
- Window size $2^{m-1} = 3$
  - Time-out
  - Correctly discarded
**Go-Back-N ARQ (cont.)**

- **Go-Back-N Efficiency**
  - completely efficient if $W_s$ is large enough to keep channel busy, and if channel is error free

  ![Diagram](image)

  • in case of error-prone channel, with $P_f$ frame loss probability, time to deliver a frame is:
    - $t_{frame}$ - if 1st transmission succeeds – prob. $(1-P_f)$
    - $t_{frame} + \frac{1}{1-P_f} W_s \cdot t_{frame}$ - if 1st transmission does NOT succeeds – prob. $P_f$

  ![Diagram](image)

  • total average time required to transmit a frame:

    $$t_{GBN} = (1-P_f) \cdot t_{frame} + P_f \left( t_{frame} + \frac{1}{1-P_f} \cdot W_s \right) = t_{frame} + \frac{P_f}{1-P_f} \cdot W_s \cdot t_{frame}$$

  ![Diagram](image)

  • transmission efficiency

    $$\eta_{GBN} = \frac{n_f - n_{header}}{t_{GBN}} = \frac{1 - n_{header}}{R} = \frac{n_f}{1 + (W_s - 1)P_f} (1-P_f)$$  (***)
Go-Back-N ARQ (cont.)

What is total average time required to transmit a frame, assuming $P_f$?

1\textsuperscript{st} attempt successful: $t_{GBN} = t_{\text{frame}}$

2\textsuperscript{nd} attempt successful: $t_{GBN} = t_{\text{frame}} + W_S \cdot t_{\text{frame}}$

average case: $t_{GBN} = t_{\text{frame}} + E[\# \text{ of transmissions in error}] \cdot W_S \cdot t_{\text{frame}}$

$$E[\# \text{ of transmissions in error}] = \frac{P_f}{1-P_f}$$

$$t_{GBN} = t_{\text{frame}} + \frac{P_f}{1-P_f} W_S \cdot t_{\text{frame}}$$

$$\eta_{GBN} = \frac{n_f - n_{\text{header}}}{R} = \frac{1 - n_{\text{header}}}{1 + (W_S - 1)P_f}$$
Example  [ Stop-and-Wait vs. Go-Back-N ]

\( n_f = 1250 \text{ bytes} = 10000 \text{ bits} \)
\( n_{ACK} = n_{header} = 25 \text{ bytes} = 200 \text{ bits} \)

Compare S&W with GBN efficiency for random bit errors with \( p_b = 0, 10^{-6}, 10^{-5}, 10^{-4} \) and bandwidth-delay product \( R \times 2 \times (t_{prop} + t_{proc}) = 1 \text{ Mbps} \times 100 \text{ ms} = 100000 \text{ bits} = 10 \text{ frames} \) → use \( W_s = 11 \).

<table>
<thead>
<tr>
<th>Efficiency</th>
<th>( p_b=0 )</th>
<th>( p_b=10^{-6} )</th>
<th>( p_b=10^{-5} )</th>
<th>( p_b=10^{-4} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>S&amp;W</td>
<td>8.9%</td>
<td>8.8%</td>
<td>8.0%</td>
<td>3.3%</td>
</tr>
<tr>
<td>GBN</td>
<td>98%</td>
<td>88.2%</td>
<td>45.4%</td>
<td>4.9%</td>
</tr>
</tbody>
</table>

- Go-Back-N provides significant improvement over Stop-and-Wait for large delay-bandwidth product
- Go-Back-N becomes inefficient as error rate increases
(3) Selective Repeat ARQ
Selective Repeat ARQ

• Go-Back-N is NOT suitable for ‘noisy links’ – in case of a lost/damaged frame a whole window of frames need to be resent
  ▪ excessive retransmissions use up the bandwidth and slow down transmission

• Selective Repeat ARQ overcomes the limitations of Go-Back-N by adding 2 new features
  (1) receiver window > 1 frame, so that out-of-order but error-free frames can be accepted
  (2) retransmission mechanism is modified – only individual frames are retransmitted

• Selective Repeat ARQ is used in TCP !!!

![Diagram showing sender and receiver windows](image-url)
Selective Repeat ARQ Operation

Receiver:
- window advances whenever next in-order frame arrives
- out-of-order frames are accepted only if their sequence numbers satisfy
  \[ R_{next} < R_{frame} < R_{next} + W_s \]
- a **negative ACK** (NAK) with sequence number \( R_{next} \) is sent whenever an out-of-sequence frame is observed

Sender:
- window advances whenever an ACK arrives
- if a timer expires, the corresponding frame is resent, and the timer is reset
- whenever a NAK arrives, \( R_{next} \) frame is resent
Selective Repeat ARQ (cont.)

**Window Sizes**

- \( W_s \) and \( W_r \)
  - \( m \) bits allotted within a header for sequence numbers
    \( \Rightarrow \) \( 2^m \) possible sequence numbers
  - **how big should the windows be!?**
  - \( W_s \) and \( W_r = 2^m-1 \) cannot be accepted due to possible ambiguity as shown below
    - \( W = 2^m/2 = 2^{m-1} \) acceptable !!!

---

**Diagram:**

- Window size \( 2^m-1 = 3 \)
- Window size \( 2^{m-1} = 2 \)
Selective Repeat ARQ (cont.)

Selective Repeat Efficiency

- completely efficient if $W_s$ is large enough to keep channel busy, and if channel is error free
  - of course, sequence number space must be 2X sequence number space of Go-Back-N

- in case of error-prone channel, total average time required to transmit a frame:

$$t_{SR} = t_{frame} = \frac{n_f}{1 - P_f} = \frac{n_f}{R \cdot (1 - P_f)}$$

- transmission efficiency

$$\eta_{SR} = \frac{R_{eff}}{R} = \frac{n_f - n_{header}}{R} = \left(1 - \frac{n_{header}}{n_f}\right) \cdot (1 - P_f)$$

(***
Selective Repeat ARQ (cont.)

What is total average time required to transmit a frame, assuming $P_f$?

1\textsuperscript{st} attempt successful: \quad $t_{SR} = t_{frame}$

2\textsuperscript{nd} attempt successful: \quad $t_{SR} = t_{frame} + t_{frame}$

average case: \quad $t_{SR} = t_{frame} + \mathbb{E}[^\# \text{ of transmissions in error}] \cdot t_{frame}$

\[ t_{SR} = t_{frame} + \frac{P_f}{1-P_f} \cdot t_{frame} = \frac{1}{1-P_f} \cdot n_f \cdot \frac{1}{R} \]

\[ \eta_{SR} = \frac{t_{SR}}{R} = \left(1 - \frac{n_{header}}{n_f}\right)(1-P_f) \]
Stop-and-Wait vs. Go-Back-N vs. Selective Repeat

Performance Comparison

- assume $n_{\text{ACK}}$ and $n_{\text{header}}$ are negligible relative to $n_f$, and
- efficiencies of three ARQ techniques are

$$L = \frac{2(t_{\text{prop}} + t_{\text{proc}})R}{n_f} = W_s - 1$$

$W_s$ is for 1 less than the number of frames currently in transit

size of the “pipe” in multiples of frames

$$\eta_{SW} = \frac{1}{1+L} \cdot (1-P_f)$$

$$\eta_{GBN} = \frac{1}{1+LP_f} \cdot (1-P_f)$$

$\eta_{SW} < \eta_{GBN} < \eta_{SR}$

$$\eta_{SR} = (1-P_f)$$

- for $0 < P_f < 1$, Selective Repeat provides best performance
- for $P_f \to 0$ Go-Back-N as good as Selective Repeat
ARQ Efficiency Comparison

Delay-Bandwidth product = 10, 100