Chapter 5

Peer-to-Peer Protocols

Purpose of Peer-to-Peer Protocols

- Chapter examines two cases:
  - providing reliable communication across noise digital transmission lines
    - shows general function of the protocols
  - operating end-to-end across a network
    - shows providing reliable stream services and other services to adapt to impairments in the network
Basic purpose of a protocol

- Provide a service to a higher layer, called a **service model**
- Peer-to-peer protocols occur in two settings:
  - across a single hop in a network
  - end-to-end across an entire network
Hop-by-hop

- Example shows implementing reliable communication and error-control procedures at every hop
- Every node required to implement
- Quicker error-recovery, more reliable service, but more complex implementation, and vulnerable to internal node errors

End-to-end

- Removes error-recovery responsibility from intermediate nodes
- Simpler for intermediate nodes, but slower to recover
- Which is implemented? Both. TCP for end-to-end, HDLC for hop-by-hop, etc.
Single hop - protocol stack view

Single hop - spatial view
End-to-end peer transport

- Can be implemented higher than transport layer
- Must take into account all characteristics of segment transfer

End-to-end - protocol stack view
Service Models

- Connection-oriented services
  - setup procedure initializes state
  - data transfer occurs, state used to track PDUs between processes, SDUs to higher layers
  - connection release removes state, releases resources
Connectionless services

- no connection setup
- address info included in each PDU
- may not provide acknowledgment
  - in this case, no retransmission
  - appropriate when
    - transfer of PDUs is generally reliable, or
    - when higher layer is more sensitive to delay
- if does provide acknowledgment, then is it still connectionless?

Transfer Capabilities

- Service models may transfer blocks (connectionless), streams or sequences of blocks (connection-oriented)
- Some may transfer at a constant or variable bit rate
- May include quality-of-service (QoS) requirement - e.g. probability of errors, loss, incorrect delivery or transfer delay - or simply offer best-effort service
Adaptation functions

![Diagram showing adaptation functions between network and application](image)

- Functions introduced between network and application, handle:
  - arbitrary message size, reliability and sequencing, pacing and flow control, timing, addressing, privacy, integrity and authentication

Segmentation and reassembly

1 call = 1 message = entire sequence of speech samples

1 call = sequence of 1-byte messages

1 long message

2 or more short messages

2 or more blocks

1 block
ARQ

- Automatic Repeat Request - technique to overcome transmission errors
- Requires header with control information, CRC check bits at end
- Traditionally implemented over single noisy communications channel
- As network reliability improved, moved to edges for end-to-end reliability

Basic elements of ARQ

![Diagram of ARQ elements]

- Packet sequence
  - Transmitter
  - Information frames
  - Station A
- Control frames
- Information Frame
  - CRC
  - Header
  - Information packet
- Receiver
  - Station B
  - Error-free packet sequence
  - Control frame
  - CRC
  - Header

Leon-Garcia Notes  COSC 6377  5-17
1. Stop-and-Wait ARQ

- Transmitter and receiver work on delivery of single frame at a time
- Uses ACKs and I-frame timer
- Susceptible to loss of ACK - can cause duplicate delivery
- Susceptible to premature timeout - can cause gaps in transmission

Ambiguities from unnumbered frames

(a) Frame 1 lost

(b) ACK lost

- In parts (a) and (b) transmitting station A acts the same way, but part (b) receiving station B accepts frame 1 twice.
- Solution is to number frames.
Unnumbered ACKs

- Premature timeouts cause gaps in transmission sequence
- Also caused by delayed ACKs
- Solution is adding sequence number to ACK

System state in Stop-and-Wait ARQ

- Global State: $(S_{last}, R_{next})$
  - Error-free frame 0 arrives at receiver
  - ACK for frame 1 arrives at transmitter
  - Error-free frame 1 arrives at receiver
  - ACK for frame 0 arrives at transmitter
Use of checkpointing

- An enquiry frame (ENQ) can be sent to cause the other side to retransmit
- Useful in avoiding retransmissions of long frames

Stop-and-Wait ARQ

- Inefficient when time to transmit across network (round-trip delay) is longer than time to place the frame on the network (serialization delay)

1000 bit frame on 1.5Mb/s link with 40ms serialization
Without SaW ARQ:
\[ 40 \times 10^{-3} \text{ sec} \times 1.5 \times 10^6 \text{ bits/sec} = 60,000 \text{ bits} \]
With SaW ARQ:
\[ 40 \times 10^{-3} \text{ sec} \times 1000 \text{ bits/40ms} = 1000 \text{ bits} \]
Delay-Bandwidth Product

- Bit rate times delay to act
  - example: 1.5Mbps x 40ms
- Measurement of lost opportunity
- Can overcome by allowing transmitter to continue to send frames
- Go-Back-N ARQ chooses number of outstanding frames $W_s$ that is greater than delay-bandwidth product

Basic Go-Back-N ARQ

- Go-Back-4: 4 frames are outstanding; so go back 4

- $W_s = 4$, send frame n only after ACK for frame n-4 is received
- If ACK not received when ready to send n, go back $W_s$ and start sending again (n-4)
Problem

- Error recovery begins only when transmitter has $W_s$ frames outstanding
- Ready to send frame $n$ and ACK for $n-W_s$ is not received
- What happens if there aren’t $n$ frames?
- If $n-W_s$ is not received but there are only $n-x$ frames, $x < W_s$?
- Answer: give each frame a timer
Go-Back-N ARQ

- Frames transmitted and ACKed
- Send Window
- buffers
- Transmitter
- Receiver
- Receive Window
- The receiver will only accept a frame that is error-free and that has sequence number $R_{next}$

Sliding Window Protocol

- Figure 5.15 is an example of a sliding-window protocol
- Left side of window is $S_{last}$, current pointer is $S_{recent}$, right side of window is $S_{last} + W_s - 1$
- ACK of $R_{next}$ move $S_{last}$ forward
Sequence numbers

- Use header bits to represent sequence number, say $m$ bits
- $2^m$ combinations, $m=3$ gives 0,1,…,7
- Problems arise if all combinations are used -- complete sequence of lost ACKs is undetectable
- Solution is to use $2^m-1$ combinations
Inefficiency of sliding window

- If error occurs, entire window can still be transmitted before protocol will go back
- Example: if ACK is lost for frame 0, further packets 1 to $W_s-1$ will be sent, but if received, will be discarded
- To save time, NAK message can be sent when 1 is received (out of order) to go back immediately

NAK Error Recovery

Go-Back-7:

Transmitter goes back to frame 1
Piggybacking

- ACKs, NAKs, etc. are control frames
- If transmissions are bidirectional, then control frames can be added to headers of outgoing frames, *piggybacking*
- This minimizes separate control frame transmissions
- If no outgoing frame is available, control frame can still be sent
Out-of-sequence handling

- In bidirectional case, out-of-sequence handled differently
- Frames that arrive in error are ignored
- Subsequent frames that are error-free but out-of-order are discarded *after* piggybacked control information is examined (e.g. the ACK, $R_{\text{next}}$ is still used to update local $S_{\text{last}}$)

Calculation of time-out values

- I-frame timer adjusted to allow a packet to traverse the network (2 x propagation delay), the time to process the packet once received, and twice the return time
Is this still inefficient?

- With high error rate channels, yes
  - after bad frame, good frames discarded
  - good frames retransmitted instead of specific bad frame
- Solution is Selective Repeat ARQ
  - increase receive window size > 1
  - modify retransmission scheme to resend only frame in error

Selective Repeat ARQ

Frames transmitted and ACKed

Frames received

Transmitter

Receiver

Buffers

Send Window

Receive Window

\( S_{\text{last}} \)

\( S_{\text{recent}} \)

\( S_{\text{last} + W_t - 1} \)

\( R_{\text{next} + 1} \)

\( R_{\text{next} + 2} \)

\( R_{\text{next} + W_t - 1} \)

Leon-Garcia Notes  COSC 6377  5-39

Figure 5.20  5-40
Error recovery in Selective Repeat ARQ

- NAKs or expired timers result in specific frame being retransmitted, here frame 2

M=2^3=8, Selective Repeat: Send Window = Receive Window = 3

Frame 0 resent

Receive Window \{3,0,1\}

Send Window = Receive Window = 2

Frame 0 resent

Receive Window \{2,3\}

Leon-Garcia Notes   COSC 6377
Maximum window size

- Section 5.2.3 shows that when $W_S = W_R$, maximum size window is $2^{m-1}$

Transmission Efficient of ARQ

- Stop-and-Wait, Go-Back-N, and Selective Repeat ARQ
- Key parameters in determining performance are:
  - delay-bandwidth product
  - frame error rate
  - frame length
Delay components in Stop-and-Wait ARQ

- First bit arrives at B after $t_{prop}$, last bit after $t_f$
- Station B processes response, and sends an ACK, which is processed by A

Efficiency of Stop-and-Wait ARQ

<table>
<thead>
<tr>
<th>$n(f)$</th>
<th>$1K = 1024$ bytes = 8192 bits</th>
<th>$n(overhead) = 64$, $n(ack) = 64$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{prop}$ + $t_{ack}$</td>
<td>$5ms$, $1.500$ km inter-city 0.95 0.97 0.97 0.35 1.77E-02 3.39E-04</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$50ms$, $15.000$ km inter-continental 0.72 0.77 1.04E-01 2.18E-03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$500ms$, $150.000$ km satellite 0.21 0.26 1.15E-02 2.16E-04</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$n(f)$</th>
<th>$64KB = 65.535$ bytes = 524.288 bits</th>
<th>$n(overhead) = 64$, $n(ack) = 64$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{prop}$ + $t_{ack}$</td>
<td>$5ms$, $1.500$ km inter-city 0.99 0.97 0.97 0.53 2.14E-02</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$50ms$, $15.000$ km inter-continental 0.99 0.77 1.04E-01 2.18E-03</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$500ms$, $150.000$ km satellite 0.21 0.26 1.15E-02 2.16E-04</td>
<td></td>
</tr>
</tbody>
</table>
Impact of errors

- For Stop-and-Wait, delay x bandwidth product is main factor. Inefficiency scales up as errors increase.
- For Go-Back-N, (Ws-1) time wasted only when error occurs.
- For Selective-Reject, each error introduces a single additional frame to transmit

Required window sizes

<table>
<thead>
<tr>
<th>Method</th>
<th>n(f) = 1K = 1024 bytes</th>
<th>n(overhead) = 64, n(ack) = 64</th>
<th>Modem 30,000 bps</th>
<th>T1 1.5Mbps</th>
<th>T3 45Mbps</th>
<th>T4 2.4Gbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>50ms 1,500 km inter-city</td>
<td>2</td>
<td>4</td>
<td>57</td>
<td>2932</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50ms 15,000 km inter-continental</td>
<td>2</td>
<td>20</td>
<td>551</td>
<td>29299</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500ms 150,000 km satellite</td>
<td>6</td>
<td>185</td>
<td>5495</td>
<td>29271</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>n(f) = 64KB = 524,288 bytes</th>
<th>n(overhead) = 64, n(ack) = 64</th>
<th>Modem 30,000 bps</th>
<th>T1 1.5Mbps</th>
<th>T3 45Mbps</th>
<th>T4 2.4Gbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>50ms 1,500 km inter-city</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>48</td>
<td></td>
<td></td>
</tr>
<tr>
<td>50ms 15,000 km inter-continental</td>
<td>2</td>
<td>2</td>
<td>11</td>
<td>460</td>
<td></td>
<td></td>
</tr>
<tr>
<td>500ms 150,000 km satellite</td>
<td>2</td>
<td>5</td>
<td>88</td>
<td>4580</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- To maximize efficiency of Go-Back-N and Selective Reject (to keep the channel busy), frames must fill the channel while waiting for ACKs
Transmission efficiency of ARQ

Optimum frame size
Sliding-Window Flow Control

- Used when transmitter can send faster than receiver can process
- Can cause buffer overflow
- Can be used between bridges, routers, gateways, switches, applications, etc.

ON-OFF flow control

- From the time B asks A to stop, $2T_{\text{prop}}R$ more bits are transmitted (the delay x b.w. product)
- Signal sent when buffer reaches threshold, and is turned on again when low threshold is reached
Sliding-window flow control

- Simple system where permission to send again is generated after Ws=2 frames are received.
- Problem arises when ACK does not arrive and retransmission occurs -- buffer overrun can still occur
- Combining error-control and flow-control can be risky
- Can separate transmitter credits separately from acknowledgments -- this is what TCP does

Timing Recovery

- For apps that generate traffic in a synchronous and periodic fashion, network timing jitter must be addressed
- Adaptation functions can restore original timing
Adaptive clock recovery

- Receiver’s clock sped up or slowed down to match difference between measured interval and calculated interval - attempts to match clocks
- Must take into account network delay
Clock recovery with synchronous network

- Ties all network elements to common clock
- Clock can be delivered on separate constant-jitter network or interleaved
- Local clock still maintained, compared constantly against reference
- If reference is lost, local clocks continue, but can drift

TCP Preview

- Of note is possibility for old messages from previous connections to arrive at receiver
- Sequence numbers are large, and initial numbers are random
Data Link Controls

- Requires additional functions than to transmit binary stream (chapter 3)
- Framing -- introducing boundaries
- Error control
- Flow control - possibly required
- Address information
- Data Link Control provides these and maintenance and security functions
- Assumes wirelike connection - frames remain in order
Types of services

- HDLC uses layer one for bit transport service
- Can provide connection-oriented service to provide error-free, ordered delivery
- Can provide connectionless - acknowledged or unacknowledged (LAN)

HDLC Configurations

Unbalanced Point-to-point link
- Primary
- Secondary
- Commands
- Responses

Unbalanced Multipoint link
- Primary
- Secondary
- Commands
- Responses

Balanced Point-to-point link between Combined Stations
- Primary
- Secondary
- Commands
- Responses
HDLC Frame Format

- Address always contains secondary address
- Flags are fixed: 01111110. Bit-stuffing prevents this pattern from appearing elsewhere in the frame

Control field format

- Information Frame
  - Flag | Address | Control | Information | FCS | Flag
  - 1    | 2-4     |      | 5           | 6-8 |
  - 0    | N(S)    | P/F  | N(R)        |

- Supervisory Frame
  - Flag | Address | Control | Information | FCS | Flag
  - 1    | 0      | S      | S           | P/F | N(R)

- Unnumbered Frame
  - Flag | Address | Control | Information | FCS | Flag
  - 1    | 1      | M      | M           | P/F | M
Connection setup and release

Exchange of frames using normal response mode
**Frame exchange using async balanced mode**

<table>
<thead>
<tr>
<th>Combined Station A</th>
<th>Combined Station B</th>
</tr>
</thead>
<tbody>
<tr>
<td>B, 1, 0, 0</td>
<td>A, 1, 0, 0</td>
</tr>
<tr>
<td>B, 1, 1, 0</td>
<td>A, 1, 1, 1</td>
</tr>
<tr>
<td>B, 1, 2, 1</td>
<td>A, 1, 2, 1</td>
</tr>
<tr>
<td>B, 1, 3, 2</td>
<td>B, REJ, 1</td>
</tr>
<tr>
<td>B, 1, 4, 3</td>
<td>A, 1, 3, 1</td>
</tr>
<tr>
<td>B, 1, 1, 3</td>
<td>B, RR, 2</td>
</tr>
<tr>
<td>B, 1, 2, 4</td>
<td>B, RR, 3</td>
</tr>
<tr>
<td>B, 1, 3, 4</td>
<td></td>
</tr>
</tbody>
</table>

**PPP Frame Format and Example**

<table>
<thead>
<tr>
<th>Flag</th>
<th>Address</th>
<th>Control</th>
<th>Protocol</th>
<th>Information</th>
<th>CRC</th>
<th>flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>01111110</td>
<td>11111111</td>
<td>00000011</td>
<td></td>
<td></td>
<td></td>
<td>01111110</td>
</tr>
</tbody>
</table>

- All stations are to accept the frame
- Unnumbered frame
- Specifies what kind of packet is contained in the payload, e.g., LCP, NCP, IP, OSI CLNP, IPX

- Similar to HDLC - has frames, but does not need bit-stuffing -- indicates integer number of bytes
Typical PPP Phase Diagram for Link Control Protocol (LCP)

1. Carrier Detected
2. Options Negotiated
3. Authentication Completed
4. NCP Configuration
5. Open
6. Done
7. Carrier Dropped

Network

Terminates

Dead

Home PC to Internet Service Provider
1. PC calls router via modem.
2. PC and router exchange LCP packets to negotiate PPP parameters.
3. Check on identities.
4. NCP packets exchanged to configure the network layer, e.g., TCP/IP (requires IP address assignment).
5. Data transport, e.g. send/receive IP packets.
6. NCP used to tear down the network layer connection (free up IP address); LCP used to shut down data link layer connection.
7. Modem hangs up.