Transport Layer Design Issues:

1. Transport Layer Functions
2. Multiplexing and Demultiplexing
3. Error Detection: Checksum
4. Flow Control
5. Efficiency Principle
6. Error Control: Retransmissions

Overview

Transport = End-to-End Services
Services required at source and destination systems
Not required on intermediate hops

Application
Transport
Network
Datalink
Physical

End System
Router
Router
End System

TCP and UDP

Audio/Video recordings of this lecture are available on-line at:
http://www.cse.wustl.edu/~jain/cse473-16/
Transport Layer Functions

1. Multiplexing and demultiplexing: Among applications and processes at end systems

2. Error detection: Bit errors

3. Loss detection:Lost packets due to buffer overflow at intermediate systems (Sequence numbers and acks)

4. Error/loss recovery: Retransmissions

5. Flow control: Ensuring receiver has buffers

6. Congestion Control: Ensuring network has capacity

Not all transports provide all functions

Multiplexing and Demultiplexing

User Datagram Protocol (UDP)

- Connectionless end-to-end service
- Provides multiplexing via ports
- Error detection (Checksum) optional. Applies to pseudo-header (same as TCP) and UDP segment. If not used, it is set to zero.
- No error recovery (no acks). No retransmissions.
- Used by network management, DNS, Streamed multimedia (Applications that are loss tolerant, delay sensitive, or have their own error recovery mechanisms)

Source Port Dest Port Checksum Length 16b 16b 16b 16b 16b

Protocol Layers

Top-Down Approach

Transport Layer Functions

<table>
<thead>
<tr>
<th>Network</th>
<th>Transport</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet</td>
<td>TCP/IP</td>
<td>HTTP, FTP, SMTP, DNS, Skype</td>
</tr>
<tr>
<td>Wi-Fi</td>
<td>UDP</td>
<td>DNS, Streamed multimedia</td>
</tr>
<tr>
<td>Fiber</td>
<td>P2P</td>
<td>Skype</td>
</tr>
<tr>
<td>Ethernet Point-to-Point</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ip</td>
<td>UDP, TCP</td>
<td></td>
</tr>
</tbody>
</table>
Error Detection: Checksum

T Cyclic Redundancy Check (CRC): Powerful but generally requires hardware
T Checksum: Weak but easily done in software

Example:
1's complement of 1's complement sum of 16-bit words with wraparound.

At receiver the sum is all 1's and the checksum is zero.

A. Consider the following two 16-bit words: ABCD 1234

Homework 3A

Consider the following two 16-bit words: ABCD 1234
B. What is the checksum as computed by the sender?
C. Now assume that the first bit of the packet is flipped due to an error. What is the checksum of the received three 16-bit words?

UDP: Summary
1. UDP provides flow multiplexing using port #s
2. UDP optionally provides error detection using the checksum
3. UDP does not have error or loss recovery
Lab 3A: UDP

T Download the wireshark traces from http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip

T Open the trace file http-ethereal-trace-5 in Wireshark, View Expand All and answer the following questions?

1. Examine the first packet. In the printout for this packet, select the line with UDP. The UDP header is highlighted in the trace. How many fields are there in the UDP header? Name these fields, their lengths and their values.

2. The value in the Length field is the length of what? Verify your claim with your captured UDP packet.

3. What is the maximum number of bytes that can be included in a UDP payload?

4. What is the largest possible source port number?

5. Examine the first two UDP packets. Describe the relationship between the port numbers in the two packets.

Flow Control

Flow Control Goals:

1. Sender does not flood the receiver, 2. Maximize throughput

Stop and Wait Flow Control

Throughput = \frac{L}{R}\frac{T+L}{R}

Here, \( \frac{T+L}{R} \) is the time to transmit one packet, \( \frac{L}{R} \) is the time to transmit one byte, and \( T \) is the round-trip time.

Receiver

Sender

First bit arrives

Last bit arrives, send ACK

ACK arrives, send next packet, \( t \geq RTT + \frac{L}{R} \)

Stop and Wait Flow Control Window

Throughput = \frac{L}{R}\frac{T+L}{R}

Here, \( D = \frac{t_{prop}}{t_{frame}} \)

3.13

Sliding Window Diagram

Stop and Wait Flow Control

first bit transmitted, \( t = 0 \)

Sender

Receiver

RTT

last bit transmitted, \( t = \frac{L}{R} \)

First bit arrives

Last bit arrives, send ACK

ACK arrives, send next packet, \( t = RTT + \frac{L}{R} \)

U = \frac{L}{R}\frac{T+L}{R}

Here, \( D = \frac{t_{prop}}{t_{frame}} \)
Sliding Window Protocol Efficiency

Data

Ack

$t_{frame}$

$t_{prop}$

$U = \frac{W}{2D + 1}$

Here, $D = \frac{t_{prop}}{t_{frame}}$

$t_{frame} = \frac{L}{R}$

$t_{prop} = \frac{270}{71} = 3.8$ ms

$U = 1 - \frac{1}{2D + 1} = 0.98$

Effect of Window Size

For all protocols, the maximum utilization is a non-increasing function of $D$.

Utilization: Examples

Satellite Link: One-way Propagation Delay = 270 ms

$RTT = 540$ ms

Frame Size $L = 500$ Bytes = 4 kb

Data rate $R = 56$ kbps

$t_{frame} = 400$ ms

$D = \frac{270}{400} = 0.012$

$U = 1 - \frac{1}{2D + 1} = 0.98$

Note: The textbook uses $RTT$ in place of $t_{prop}$ and $L/R$ for $t_{frame}$.
Error Control: Retransmissions

- Retransmit lost packets

**Automatic Repeat Request (ARQ)**

---

**Stop and Wait ARQ**
- Receiver does not cache out-of-order frames
- Receiver sends an ACK for each received packet
- If a packet is not acknowledged within a timeout period, the sender retransmits the packet

**Go-Back-N ARQ**
- Receiver does not cache out-of-order frames
- Sender has to go back and retransmit all frames after the lost frame

**Selective Repeat ARQ**
- Receiver caches out-of-order frames
- Sender retransmits only the lost frame
- Also known as selective repeat ARQ

---

**Selective Repeat: Window Size**
- Sequence number space > $2^n - 1$
- Window size < $2^{n-1}$
- Timeout
Performance: Maximum Utilization

Stop and Wait Flow Control:
\[ U = \frac{1}{1+2D} \]

Window Flow Control:

Stop and Wait ARQ:
\[ U = \frac{1-P}{1+2D} \]

Go-back-N ARQ:
\[ U = \frac{1}{2D+1} \] when \( W \geq 2D+1 \)
\[ U = \frac{1}{2D+1} \left( \frac{1}{1+2D} \right) \] when \( W < 2D+1 \)

Selective Repeat ARQ:
\[ U = \frac{1}{2D^2+1} \] when \( W \geq 2D+1 \)
\[ U = \frac{1}{2D^2+1} \left( \frac{1}{1+2D^2} \right) \] when \( W < 2D+1 \)

where \( D \) is the number of packets in the window and \( P \) is the probability of loss.

Performance Comparison

<table>
<thead>
<tr>
<th>Distance (km)</th>
<th>Stop and Wait</th>
<th>Go-back-N</th>
<th>Selective Repeat</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.80</td>
<td>0.75</td>
<td>0.85</td>
</tr>
<tr>
<td>100</td>
<td>0.60</td>
<td>0.55</td>
<td>0.65</td>
</tr>
<tr>
<td>1000</td>
<td>0.40</td>
<td>0.35</td>
<td>0.45</td>
</tr>
</tbody>
</table>

Transport Layer Design Issues

1. Multiplexing/demultiplexing by a combination of source and destination IP addresses and port numbers.
2. Window flow control is better for long-distance or high-speed networks.
3. Longer distance or higher speed: \( \frac{L}{L} \) Larger window is better.
4. Stop and wait flow control is better for short distance or low-speed networks.
5. Selective repeat is better than Go-back-N.

Efficiency Principle:

\[ C \] How big the window (in number of packets) is adjusted for the channel utilization to be:

\[ \frac{1}{1+2D} \]

Efficiency of the ACK:

\[ \frac{(1-P)}{(1+2D)} \]

Homework 3B

Problem 19 on page 302 of the textbook:

Consider the GBN protocol with a sender window size of 3 and a sequence number range of 1,024. Suppose that at time \( t \), the next in-order packet that the receiver is expecting has a sequence number of \( k \). Assume that the medium does not reorder messages. Answer the following questions:

A. What are the possible sets of sequence numbers inside the sender's window at time \( t \)? Justify your answer.

B. What are all possible values of the ACK field in all possible messages currently propagating back to the sender at time \( t \)? Justify your answer.

Window Flow Control:

C. How big window (in number of packets) is required for the channel utilization to be greater than 60% on a cross-country link of 4000 km running at 20 Mbps using 1-kByte packets?

Efficiency Principle:

D. Ethernet V1 access protocol was designed to run at 10 Mbps over 2.5 km using 1500-byte packets. This same protocol needs to be used at 100 Mbps at the same efficiency. What distance can it cover if the frame size is not changed?
TCP

Overview

3. Key Features of TCP

- Point-to-Point: One sender, one receiver
- Full Duplex: Bidirectional data flow in one connection
- Reliable: In-order byte delivery
- Flow Control: To avoid receiver buffer overflow
- Congestion Control: To avoid network router buffer overflow
- Maximum segment size (MSS)
- Byte stream: No message boundaries.

TCP Segment Format (Cont)

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>16b</td>
</tr>
<tr>
<td>Dest Port</td>
<td>16b</td>
</tr>
<tr>
<td>Seq No</td>
<td>32b</td>
</tr>
<tr>
<td>Ack No</td>
<td>32b</td>
</tr>
<tr>
<td>Data</td>
<td>Variable</td>
</tr>
<tr>
<td>Offset</td>
<td>16b</td>
</tr>
<tr>
<td>Window</td>
<td>16b</td>
</tr>
<tr>
<td>Urgent</td>
<td>16b</td>
</tr>
<tr>
<td>Options</td>
<td>Variable</td>
</tr>
<tr>
<td>Checksum</td>
<td>32b</td>
</tr>
<tr>
<td>Reserved</td>
<td>32b</td>
</tr>
<tr>
<td>Urgent Data</td>
<td>Variable</td>
</tr>
<tr>
<td>Pad</td>
<td>Variable</td>
</tr>
</tbody>
</table>

Key Services:

- Send: Please send when convenient
- Data stream push: Destination TCP, please deliver it immediately to the receiving application.
- Urgent data signaling: Destination TCP, please set on last packet of an application message.
- Send: Please send it now. Set on last packet of an application message.
- Urgent data signaling: Destination TCP, please give this urgent data to the user out-of-band.

TCP: Transmission Control Protocol

Principles of Congestion Control

1. TCP Header Format, Options, Checksum
2. TCP Connection Management
3. Round Trip Time Estimation
4. Principles of Congestion Control
5. Slow Start Congestion Control

Key Features of TCP

- Point-to-Point: One sender, one receiver
- Flow Control: To avoid receiver buffer overflow
- Reliable: In-order byte delivery
- Full Duplex: Bidirectional data flow in one connection
- Maximum Segment Size (MSS)
- Byte Stream: No message boundaries.
- Urgent data signaling: Destination TCP, please set on last packet of an application message.
- Send: Please send it now. Set on last packet of an application message.
- Urgent data signaling: Destination TCP, please give this urgent data to the user out-of-band.

TCP

Transmission Control Protocol

Key Services:

- Send: Please send when convenient
- Data stream push: Destination TCP, please deliver it immediately to the receiving application.
- Urgent data signaling: Destination TCP, please give this urgent data to the user out-of-band.

TCP Segment Format (Cont)

<table>
<thead>
<tr>
<th>Field</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>16b</td>
</tr>
<tr>
<td>Dest Port</td>
<td>16b</td>
</tr>
<tr>
<td>Seq No</td>
<td>32b</td>
</tr>
<tr>
<td>Ack No</td>
<td>32b</td>
</tr>
<tr>
<td>Data</td>
<td>Variable</td>
</tr>
<tr>
<td>Offset</td>
<td>16b</td>
</tr>
<tr>
<td>Window</td>
<td>16b</td>
</tr>
<tr>
<td>Urgent</td>
<td>16b</td>
</tr>
<tr>
<td>Options</td>
<td>Variable</td>
</tr>
<tr>
<td>Checksum</td>
<td>32b</td>
</tr>
<tr>
<td>Reserved</td>
<td>32b</td>
</tr>
<tr>
<td>Urgent Data</td>
<td>Variable</td>
</tr>
<tr>
<td>Pad</td>
<td>Variable</td>
</tr>
</tbody>
</table>
TCP Header Fields:

- **Source Port**: Identifies the source user process (16 bits).
- **Destination Port**: Identifies the destination user process (16 bits). Common ports include 21 (FTP), 23 (Telnet), 53 (DNS), 80 (HTTP), ...
- **Sequence Number**: Sequence number of the first byte in the segment. If SYN is present, this is the initial sequence number (ISN) and the first byte is ISN+1.
- **Ack Number**: Next byte expected (32 bits).
- **Data Offset**: Number of 32-bit words in the header (4 bits).
- **Reserved**: Reserved for future use (6 bits).
- **Control**: Indicates control flags (3 bits): URG, ACK, PSH, RST, SYN, FIN.
- **Window**: The window size is 16 bits.
- **Checksum**: Covers the segment plus a pseudo header. Includes source and destination addresses, protocol, segment length.
- **Urgent Pointer**: Points to the byte following urgent data (16 bits). Used to signal how much data should be delivered out-of-band.
- **Options**: Variable fields include:
  - Maximum Segment Size (MSS): Does not include TCP header (variable length in bytes)
  - Window Scale Factor
  - Selective Acknowledgments
  - Timestamp
  - No-Operation

TCP Options (Cont):

- **End of Options**: Stop looking for further options (0 bits).
- **No-Operation**: Ignore this byte. Used to align the next option on a 4-bytes boundary.
- **Max Segment Size (MSS)**: Does not include TCP header.
- **Options (variable)**: Stop looking for further options.
- **Checksum (16 bits)**: Covers the segment plus a pseudo header from IP: source and destination addresses, protocol, segment length.
- **Header Checksum**: 16 bits.
- **Options (variable)**: Does not include TCP header.
- **Checksum (16 bits)**: Covers the segment plus a pseudo header.

TCP Header (Cont):

- **Reserved (6 bits)**
- **Window**: Number of 32-bit words in the header (16 bits).
- **Data Offset**: Number of 32-bit words in the header (4 bits)
- **ACK number**: Next byte expected (32 bits). The first byte expected is ISN+1. The initial sequence number (ISN) and the first byte is ISN+1.
- **Sequence Number**: Sequence number of the first byte in the segment. If SYN is present, this is the initial sequence number (ISN).
- **Destination Port (16 bits)**: Identifies the destination user process.
- **Source Port (16 bits)**: Identifies the source user process.
TCP Checksum

Checksum is the 16-bit one's complement of the one's complement sum of a pseudo header of information from the IP header, the TCP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

- Checksum field is filled with zeros initially.
- TCP length (in octet) is not transmitted but used in calculations.
- Efficient implementation in RFC1071.

TCP Connection Management

- Connection Establishment
  - Three way handshake
    - SYN flag set
      - Request for connection
        - SYN, ISN = 100
      - ACK, ISN = 350, Ack 101
      - Ack 351

- Connection Termination
  - Close with FIN flag set
    - Abort
      - FIN
      - Ack
      - Ack

Round Trip Time Estimation

Example RTT estimation:

<table>
<thead>
<tr>
<th>Time (seconds)</th>
<th>RTT (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>15</td>
</tr>
<tr>
<td>22</td>
<td>22</td>
</tr>
<tr>
<td>29</td>
<td>29</td>
</tr>
<tr>
<td>36</td>
<td>36</td>
</tr>
<tr>
<td>43</td>
<td>43</td>
</tr>
<tr>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td>57</td>
<td>57</td>
</tr>
<tr>
<td>64</td>
<td>64</td>
</tr>
<tr>
<td>71</td>
<td>71</td>
</tr>
<tr>
<td>78</td>
<td>78</td>
</tr>
<tr>
<td>85</td>
<td>85</td>
</tr>
<tr>
<td>92</td>
<td>92</td>
</tr>
<tr>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>106</td>
<td>106</td>
</tr>
</tbody>
</table>

SampleRTT

Estimated RTT=(1-D)EstimatedRTT+D SampleRTT

DevRTT = (1-E)DevRTT+E |SampleRTT-EstimatedRTT|

TimeoutInterval=EstimatedRTT+4 DevRTT

Value

Very low probability

TCP Header

TCP data

Source Addr

Destination Addr

TCP Length

TCP checksum

Efficient implementation in RFC1071.

TCP length (in octet) is not transmitted but used in checksum field is filled with zeros initially.

- Three way handshake
  - Close with FIN flag set
  - Connection Termination
  - Connection Establishment

TCP is the 16-bit one's complement of the one's complement sum of a pseudo header of the IP header, the TCP header, and the data, padded with zero octets at the end (if necessary) to make a multiple of two octets.

TCP checksum
Early 1980s Digital Equipment Corporation (DEC) introduced Ethernet products. Noticed that throughput decreases with a higher-speed link in middle (because no congestion mechanisms in TCP).

Results:
1. Timeout
   - Congestion window decreases exponentially with each time out.
   - Decreases by 1 MSS in one round trip.
   - After a long idle period (e.g., two round-trip times), reset

2. Routers should set a bit when congested (DEC bit).
   - [Jain, Ramakrishnan, Chiu 1988]

3. Introduced the term “Congestion Avoidance”
4. Additive increase and multiplicative decrease (AIMD principle)
   - [Chiu and Jain 1989]

There were presented to IETF in 1986.

Slow Start Congestion Control:
- Window = Flow control avoids receiver overruns.
- Need congestion control to avoid network overruns.
- The sender maintains two windows.
- The sender maintains two windows.
- Flow control avoids receiver overruns.

**Slow Start (Cont)**

If segments lost, remember slow start threshold (SSThresh) to:
- CWND/2
- Set CWND to 1 MSS
- Increment by 1 MSS per ack until SSThresh
- Increment by 1 MSS/CWND per ack afterwards

Time

Congestion Window

CWND

Receiver Window

SSThresh

Idle

Timeout

1 MSS

SSThresh

Slow Start based on Timeout and AIMD [Van Jacobson 1988]
Additive Increase, Multiplicative Decrease

\[ W_1 + W_2 = \text{Capacity} \]

\[ \text{Efficiency}, W_1 = W_2 \]

\[ (W_1, W_2) \rightarrow (W_1 + W', W_2 - W') \]

Linear increase (45° line)

\[ (W_1, W_2) \rightarrow (kW_1, kW_2) \]

Multiplicative decrease (line through origin)

Fast Retransmit

Optional – implemented in TCP Reno

When a new ack (not a duplicate ack) is received

\[ \text{CWND} = \text{SSthresh} + 3 \text{MSS} \]

Every subsequent duplicate ack: \( \text{CWND} = \text{CWND} + \text{MSS} \)

Set \( \text{CWND} = \text{SSthresh} + 3 \text{MSS} \)

Exit Fast Recovery

Fast Retransmit

CWND = \text{SSthresh}

\[ \text{CWND} = \text{CWND} + \text{MSS} \]

Set \( \text{CWND} = \text{SSthresh} \)

\[ \text{CWND} = \text{CWND} + \text{MSS} \]

Fast Retransmit mode

On reception of duplicate acks (\( n \)th ack for the same segment):

Duplicate ack increments a loss-of-order segment

(Cumulative version was TCP Tahoe)

TCP Average Throughput

\[ \text{Average Throughput} = \frac{\text{RTT} \times P}{1 - 2 \times \text{MSS}} \]

Here, \( P \) = Probability of Packet loss

Note 1: The textbook uses \( L \) for probability of packet loss but it

\[ \text{TCP Average Throughput} = 1.22 \times \text{MSS} \times \text{RTT} \]

Note 1: The formula is an approximation which does not apply

\[ C = \min\{1, \text{Receiver Window}/\text{RTT}\} \]

Note 1: The textbook uses \( L \) for probability of packet loss but it

\[ \text{TCP Average Throughput} = 1.22 \times \text{MSS} \times \text{RTT} \]

Note 1: The textbook uses \( L \) for probability of packet loss but it

\[ \text{TCP Average Throughput} = 1.22 \times \text{MSS} \times \text{RTT} \]

Note 1: The textbook uses \( L \) for probability of packet loss but it

\[ \text{TCP Average Throughput} = 1.22 \times \text{MSS} \times \text{RTT} \]

Note 1: The textbook uses \( L \) for probability of packet loss but it

\[ \text{TCP Average Throughput} = 1.22 \times \text{MSS} \times \text{RTT} \]

Note 1: The textbook uses \( L \) for probability of packet loss but it
Explicit Congestion Notification (ECN)

Explicit congestion notification (ECN) is based on DECbit research. Two bits in IP Header:

00: Transport is not capable of ECN (e.g., UDP)
01: Transport is capable of ECN
10: Transport is capable of ECN
11: Congestion experienced (CE)

When a router encounters congestion, instead of dropping the datagram, it marks the two bits as “11” congestion experienced.

On receiving “CE” code point, the receiver sends “ECN Echo (ECE)” flag in the TCP header
On seeing the ECE flag, the source reduces its congestion window, and sets “Congestion Window Reduced (CWR) flag in outgoing segment
On receiving “CWR” flag, the receiver stops setting ECE bit

TCP Window Size

Transmission Round

TCP Summary

1. TCP uses port numbers for multiplexing
2. TCP is stream based and has window flow control
3. TCP provides reliable full-duplex connections
4. Slow-start congestion control works on timeout
5. Explicit congestion notification works using ECN bits

TCP: Summary

TCP uses port numbers for multiplexing
Homework 3C (Cont)

A. Identify the interval of time when TCP slow start is operating.

B. Identify the intervals of time when TCP congestion avoidance is operating.

C. After the 16th transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

D. After the 22nd transmission round, is segment loss detected by a triple duplicate ACK or by a timeout?

E. What is the initial value of ssthresh at the first transmission round?

F. What is the value of ssthresh at the 18th transmission round?

G. What is the value of ssthresh at the 24th transmission round?

H. During what transmission round is the 70th segment sent?

I. If triple duplicate ACKs are received at the 16th round, what will be the value of ssthresh?

J. Suppose TCP Tahoe is used (instead of TCP Reno), and assume that triple duplicate ACKs are received at the 16th round. What are the values of ssthresh and the congestion window size at the 19th transmission round?

K. Consider a scenario where TCP congestion avoidance is activated at the 16th round, and assume triple duplicate ACKs are received at the 17th round. What are the values of ssthresh and the congestion window size at the 18th transmission round?

L. Assume a packet loss is detected after the 26th round by the receiver of a triple duplicate ACK, what will be the value of the congestion window size?

Summary

1. Multiplexing/demultiplexing by a combination of source and destination IP addresses and port numbers.

2. Window flow control is better for long-distance or high-speed networks.

3. TCP provides full-duplex connections with flow control. Has error detection.

4. UDP is connectionless and simple.

Lab 3B: TCP

In Wireshark, open the trace file tcp-ethereal-trace-1 in http://gaia.cs.umass.edu/wireshark-labs/wireshark-traces.zip and answer the following:

1. What is the IP address and TCP port number used by the client computer to transfer the file to gaia.cs.umass.edu?

2. What is the IP address and port number used by gaia.cs.umass.edu to receive the file?

3. What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the client computer and gaia.cs.umass.edu? What is it in the segment that identifies the segment as a SYN segment?

4. What is the sequence number of the SYN Ack segment sent by gaia.cs.umass.edu to the client computer in reply to the SYN? What is the value of the ACKnowledgement field in the SYN Ack segment? How did gaia.cs.umass.edu determine that value? What is it in the segment that identifies the segment as a SYN Ack segment?
5. What is the sequence number of the TCP segment containing the HTTP POST command? Note that in order to find the POST command, you'll need to dig into the packet content field at the bottom of the Wireshark window, looking for a segment with a "POST" within its DATA field.

6. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What are the sequence numbers of the first six segments in the TCP connection (including the segment containing the HTTP POST)? At what time was each segment sent? When was the ACK for each segment received? Given the difference between when each TCP segment was sent and when its acknowledgement was received, what is the RTT value for each of the six segments?

7. What is the length of each of the first six TCP segments?

8. What is the minimum amount of available buffer space advertised at the receiver for the entire trace? Does the lack of receiver buffer space ever throttle the sender?

9. Are there any retransmitted segments in the trace? What did you check for in the trace in order to answer this question?

10. How much data does the receiver typically acknowledge in an ACK? Can you identify cases where the receiver is ACKing every other received segment (see Table 3.2 on page 247 in the text).

11. What is the throughput (bytes transferred per unit time) for the TCP connection? Explain how you calculated this value.

---

Optional Homework 3D

Try but do not submit.

A TCP entity opens a connection and uses slow start.

Approximately how many round-trip times are required before TCP can send N segments?

CWND=1

CWND=2

CWND=4

Hint:

- How many segments can be sent in each RTT?
- What is the maximum number of segments that can be sent in slow start?

- Calculate the total number of segments sent in slow start.

6. Consider the TCP segment containing the HTTP POST as the first segment in the TCP connection. What is the sequence number of the TCP segment containing the HTTP POST?

- "POST" within the DATA field
- Find the segment with a "POST" command in the trace
- Look for the segment that contains the HTTP POST command
- Note that in order to find the POST command, you need to look into the packet content field at the bottom of the Wireshark window.

5. What is the sequence number of the TCP segment containing the HTTP POST command?
<table>
<thead>
<tr>
<th>Acronyms (Cont)</th>
</tr>
</thead>
<tbody>
<tr>
<td>TCP</td>
</tr>
<tr>
<td>HTTP</td>
</tr>
<tr>
<td>FTP</td>
</tr>
<tr>
<td>SMTP</td>
</tr>
<tr>
<td>MSS</td>
</tr>
<tr>
<td>ISN</td>
</tr>
<tr>
<td>SYN</td>
</tr>
<tr>
<td>FIN</td>
</tr>
<tr>
<td>RST</td>
</tr>
<tr>
<td>RST</td>
</tr>
<tr>
<td>SST</td>
</tr>
<tr>
<td>RTT</td>
</tr>
<tr>
<td>SACK</td>
</tr>
<tr>
<td>URG</td>
</tr>
<tr>
<td>NAK</td>
</tr>
<tr>
<td>ACK</td>
</tr>
<tr>
<td>NUL</td>
</tr>
<tr>
<td>VCN</td>
</tr>
<tr>
<td>VCL</td>
</tr>
<tr>
<td>VCR</td>
</tr>
<tr>
<td>VCP</td>
</tr>
<tr>
<td>VID</td>
</tr>
</tbody>
</table>

Related Modules

CSE 473: Introduction to Computer Networks
CSE 571S: Network Security (Fall 2014)
CSE 574S: Wireless and Mobile Networking (Spring 2016)
CSE 474S: Introduction to Computer Networks (Fall 2016)

Scan This to Download These Slides

Audio/Video Recordings and Podcasts of Professor Raj Jain's Lectures, https://www.youtube.com/channel/UCN4-5wzNP9-ruOzQMs-8NUw

Related Modules

CSE 473: Introduction to Computer Networks
CSE 571S: Network Security (Fall 2014), http://www.cse.wustl.edu/~jain/cse571-14/index.html

Scan This to Download These Slides

Audio/Video Recordings and Podcasts of Professor Raj Jain's Lectures, https://www.youtube.com/channel/UCN4-5wzNP9-ruOzQMs-8NUw