## Chapter 3-4 Network Exam Fall 2007                                      Name: Key

1. A socket is made up of a **IP Address** and a **Port**. A connection is made up of **2** sockets.
2. UDP connections to a server by two different clients are made to the **same** socket(s) on the server.
3. Name the four things that identify a TCP connection

   **Host 1 IP/Port , Host 2 IP/Port**
4. An application on host A sends a segment to an application on host B. If the two hosts are on different subnets, what is the minimum number of interfaces that the segment must traverse? If you have doubts about the number you choose write a very short explanation.

   **4**

5. Suppose you are given an IP of 216.55.37.77 and a subnet mask of 255.255.255.224. Find the network address and broadcast addresses.

   Network: **216.55.37.64**      Broadcast**: 216.55.37.95**

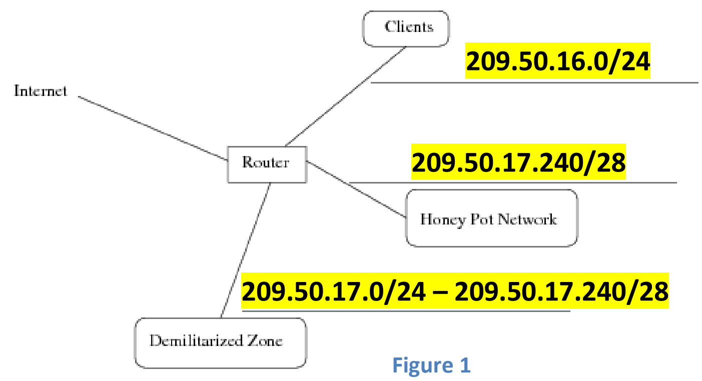



Figure 1

6. Suppose you are given the range of IP address: 209.50.17.0/23. You want to have a network with 250 clients, a demilitarized zone (DMZ) with 200 addresses and a honey pot network with 16 addresses. Break the range given into ranges of the form w.x.y.z/s on Figure 1.

7. Suppose we label the interfaces on the router in Figure 1 as follows: Clients=0, Honey Pot=1, DMZ=2, Internet=3. Give a forwarding table using your answer from Question 7.

| | Prefix | Interface |
|---|---|---|
| Client | 11010001.00110010.0001 | 0 |
| Honey Pot Network | 11010001.00110010.00010001 | 1 |
| DMZ | 11010001.00110010.00010001.1111 | 2 |
| Internet | Otherwise | 3 |

8. Write a short paragraph describing RIP and OSPF. Make sure to discuss the problems of Link state and distance vector and how RIP and OSPF mitigate the problems.

| **RIP → Distance Vector** | **OSPF → Link State** |
|---|---|
| **Count to infinity problem** | **Oscillating route problem** |
| **Poisoned Reverse: Tell the node you route to X through you don't know how to get to X** | **Allow multiple path routing** |
| **Does not fix the problem** | **Does fix the problem** |
| | |

9. Using Figure 2 starting at node A highlight or darken the lines that Dijkstra's algorithm would use in forming a Minimum Spanning Tree

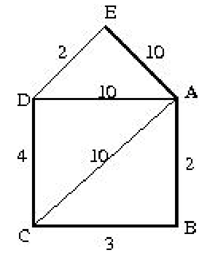
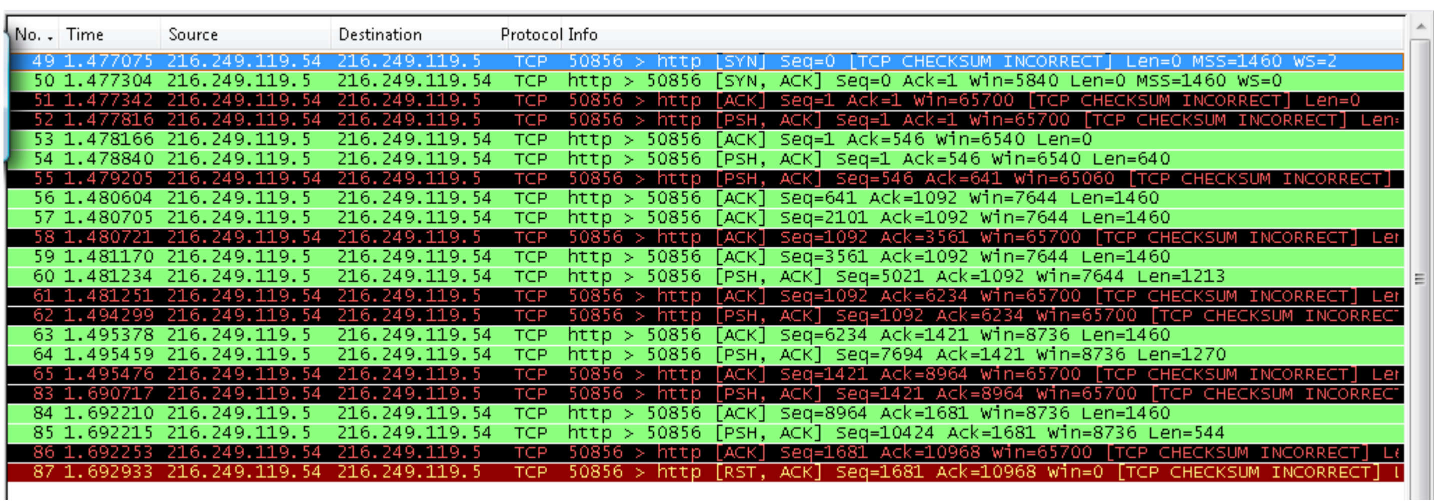



Figure 2

Matching

| | | | |
|---|---|---|---|
| **I** | Go-Back-N | A | Protocol(s) that provide process to process communication |
| **S** | Selective Repeat | B | Fast Recovery (no Slow Start) |
| **C** | Slow Start | C | Congestion window doubles every round trip |
| **P** | Slow start phase | D | Translates routable IPs to private IPs |
| **N** | Congestion avoidance phase | E | Delivering data in a segment to the correct socket |
| **B** | Triple duplicate acknowledgement (phase trans) | F | Routing IP using Distance Vector |
| **T** | Time out (phase trans) | G | The number of bytes received + 1 |
| **J** | Threshold value | H | TCP uses it to keep the sender from overwhelming the receiver |
| **G** | Sequence number | I | Sliding window protocol |
| **U** | Bottleneck link | J | Where congestion avoidance starts |
| **H** | Flow control | K | Every multicast sender gets its own tree |
| **X** | Congestion control | L | Routing protocol that uses Link State |
| **A** | TCP/UDP | M | Process of collecting data and building forwarding tables |
| **E** | Demultiplexing | N | Linear growth of the congestion window |
| **Z** | Forwarding | O | Protocol used between Autonomous Systems (AS) |
| **M** | Routing | P | Exponential growth in the congestion window |
| **W** | Packet loss on input ports | Q | When input links and fabric run faster than output links |
| **Q** | Packet loss on output ports | R | Class D address space contains __________________ |
| **F** | RIP | S | Resends only those packets that were lost |
| **L** | OSPF | T | Congestion window drops to 1 |
| **O** | BGP | U | The slowest link on a path traversed by a packet |
| **R** | Multicast Addresses | V | A service that dynamically allocates IP addresses |
| **Y** | Group Shared Tree | W | When switching fabric is slower than input links |
| **K** | Source based tree | X | Accomplished by checking for triple Acks and Timeouts |
| **V** | DHCP | Y | A tree not rooted at a multicast source |
| **D** | NAT | Z | The process of moving a packet from an input port to an output port. |



| No. | Time | Source | Destination | Protocol | Info |
|---|---|---|---|---|---|
| 49 | 1.477075 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [SYN] Seq=0 [TCP CHECKSUM INCORRECT] Len=0 MSS=1460 WS=2 |
| 50 | 1.477304 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [SYN, ACK] Seq=0 Ack=1 Win=5840 Len=0 MSS=1460 WS=0 |
| 51 | 1.477342 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [ACK] Seq=1 Ack=1 Win=65700 [TCP CHECKSUM INCORRECT] Len=0 |
| 52 | 1.477816 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [PSH, ACK] Seq=1 Ack=1 Win=65700 [TCP CHECKSUM INCORRECT] Len= |
| 53 | 1.478166 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [ACK] Seq=1 Ack=546 Win=6540 Len=0 |
| 54 | 1.478840 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [PSH, ACK] Seq=1 Ack=546 Win=6540 Len=640 |
| 55 | 1.479205 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [PSH, ACK] Seq=546 Ack=641 Win=65060 [TCP CHECKSUM INCORRECT] |
| 56 | 1.480604 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [ACK] Seq=641 Ack=1092 Win=7644 Len=1460 |
| 57 | 1.480705 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [ACK] Seq=2101 Ack=1092 Win=7644 Len=1460 |
| 58 | 1.480721 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [ACK] Seq=1092 Ack=3561 Win=65700 [TCP CHECKSUM INCORRECT] Ler |
| 59 | 1.481170 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [ACK] Seq=3561 Ack=1092 Win=7644 Len=1460 |
| 60 | 1.481234 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [PSH, ACK] Seq=5021 Ack=1092 Win=7644 Len=1213 |
| 61 | 1.481251 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [ACK] Seq=1092 Ack=6234 Win=65700 [TCP CHECKSUM INCORRECT] Ler |
| 62 | 1.494299 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [PSH, ACK] Seq=1092 Ack=6234 Win=65700 [TCP CHECKSUM INCORREC |
| 63 | 1.495378 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [ACK] Seq=6234 Ack=1421 Win=8736 Len=1460 |
| 64 | 1.495459 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [PSH, ACK] Seq=7694 Ack=1421 Win=8736 Len=1270 |
| 65 | 1.495476 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [ACK] Seq=1421 Ack=8964 Win=65700 [TCP CHECKSUM INCORRECT] Ler |
| 83 | 1.690717 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [PSH, ACK] Seq=1421 Ack=8964 Win=65700 [TCP CHECKSUM INCORREC |
| 84 | 1.692210 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [ACK] Seq=8964 Ack=1681 Win=8736 Len=1460 |
| 85 | 1.692215 | 216.249.119.5 | 216.249.119.54 | TCP | http > 50856 [PSH, ACK] Seq=10424 Ack=1681 Win=8736 Len=544 |
| 86 | 1.692253 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [ACK] Seq=1681 Ack=10968 Win=65700 [TCP CHECKSUM INCORRECT] Le |
| 87 | 1.692933 | 216.249.119.54 | 216.249.119.5 | TCP | 50856 > http [RST, ACK] Seq=1681 Ack=10968 Win=0 [TCP CHECKSUM INCORRECT] l |

Figure 3

**Figure 3 gives a capture form wireshark. Use it to fill in the following information:**

| | |
|---|---|
| Client IP Address | 216.249.119.54 |
| Server IP | 216.249.119.5 |
| Client Port | 50856 |
| Server Port | 80 |
| Using the No. column, Identify the packets that setup the connection | 49-51 |
| How many errors are shown in this Figure? | 10 |
| Is this connection closed? If so which packet completes the close? | No [it is reset] |
| What application layer protocol is being used? | HTTP |
| Identify the flags being used (e.g. ACK) | SYN, ACK, PSH, RST |

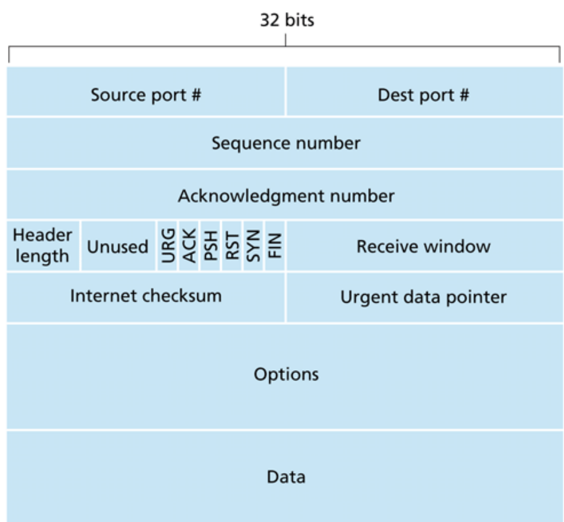



Figure 4

**Using** Figure 4 **Identify the following:**

| | |
|---|---|
| Identifies the application sending the packet | Source Port |
| Identifies how much data has been received | Sequence Number |
| Advertises information used for flow control | Receive Windows |
| Set to start a connection | SYN Flag |
| Set to end a connection | FIN Flag |
| Set on every segment except the first one | ACK Flag |
| Supplies error checking information | Internet Checksum |

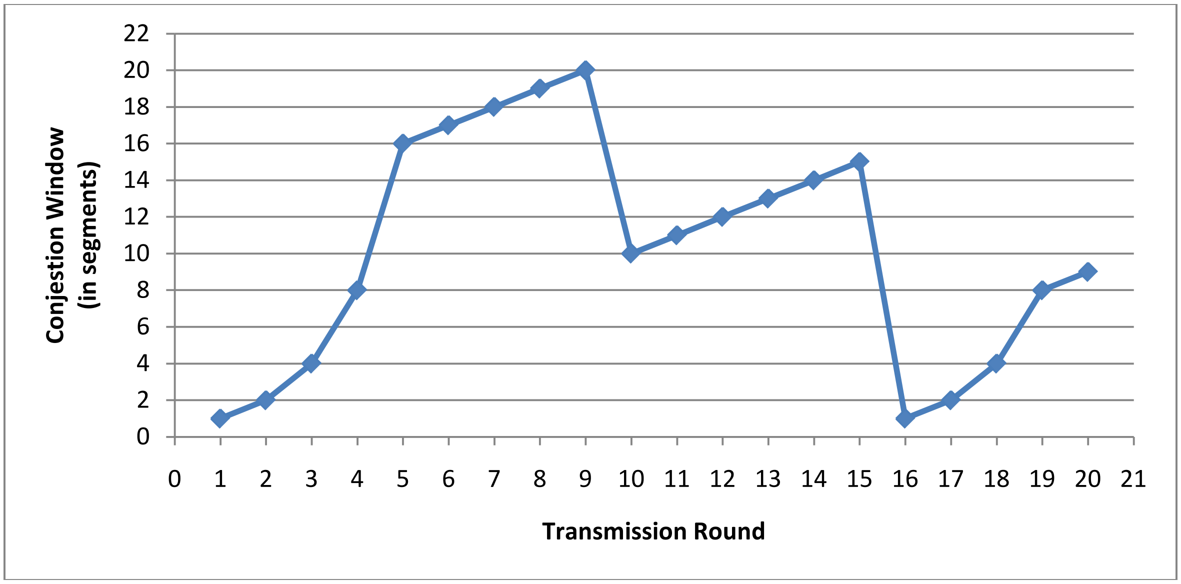



Figure 5: TCP Reno Graph

Identify the phases/events in Figure 5:

Slow Start 1-5, 16-19

Congestion Avoidance 5-9, 10-15, 19-20

Triple ACK 9

Time Out 15

The distance vector algorithm is similar to P26 on your homework, which I gave you the answer to. Take a look at for review and see if you can do this one too.

The link-state algorithm is basically finding the shortest path. Therefore creating the table is as simple as following the rules on page 378.

| Step | N' | D(B), p(B) | D(C), p(C) | D(D), p(D) | D(E), p(E) |
|---|---|---|---|---|---|
| 0 | A | 2, A | 10, A | 10, A | 10, E |
| 1 | AB | | 5, B | 10, A | 10, E |
| 2 | ABC | | | 9, C | 10, E |
| 3 | ABCD | | | | 10,E |
| **4** | **ABCD** | **2,A** | **5, B** | **9, C** | **10,E** |