Professional Masters in Information and Cyber Security (PMICS)

Computer Science and Engineering,
University of Dhaka

CSE 802

Information Security Fundamentals

- Course Instructor: Dr. Mosarrat Jahan, Associate Professor, Dept. of CSE, DU
- Course Co-instructor: Tawhidur Rahman, Chief Data Security Officer, National CIRT, Digital Data Security Agency
- Teaching Assistant: Mr. Md. Samiul Islam, Incident Associate, BGD E-GOV CIRT

Contact:

Course instructor : mosarratjahan@cse.du.ac.bd

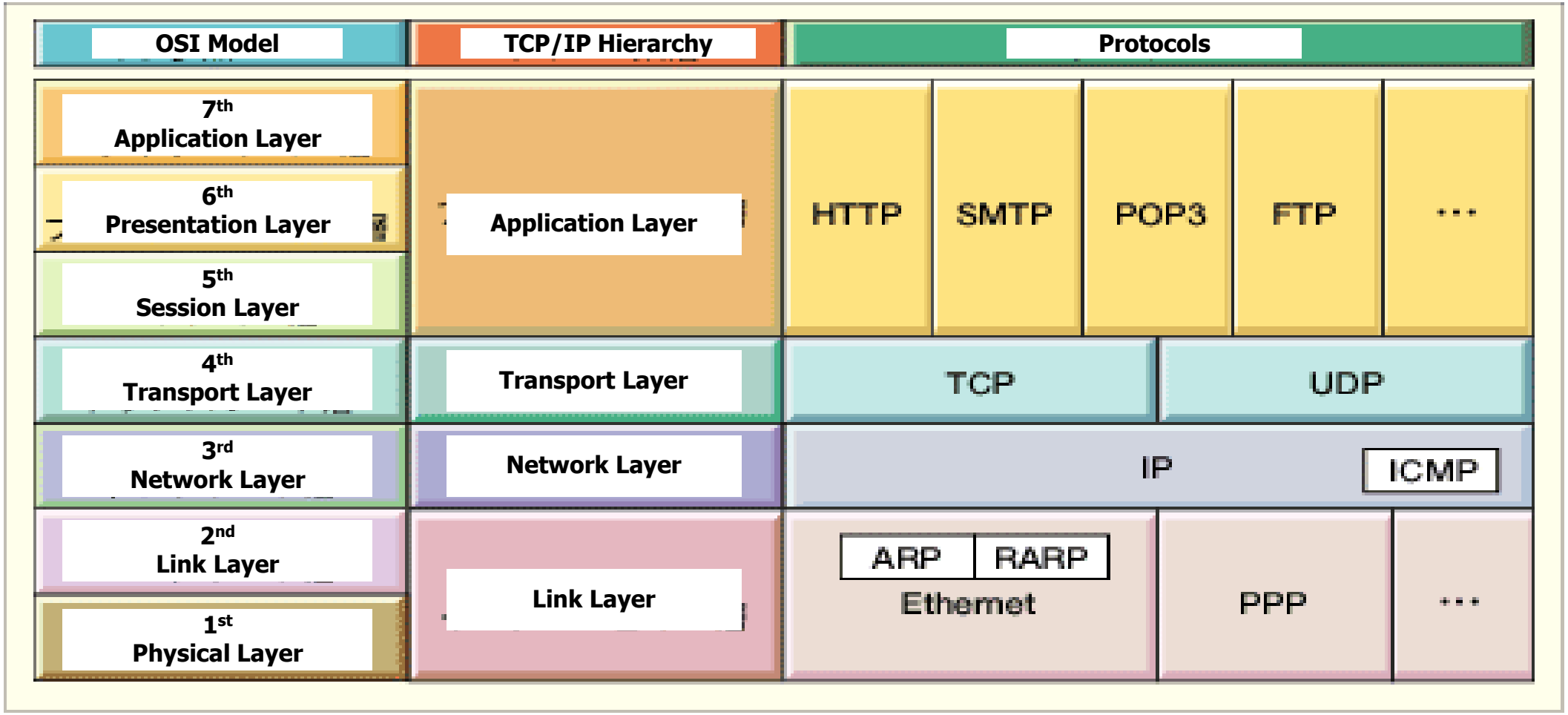
Course co-instructor: pialfg@gmail.com

Google Classroom Code: odl3lrc

Lecture Content

- Review TCP packet header and other necessary concepts
- Split-handshake attacks
- TCP congestion control and Shrew DoS attacks
- TCP SYN Flooding attacks
- IP source spoofing attack
- Prevention Mechanisms
- Trouble-shooting with Netstat utility

OSI and TCP/IP Protocol Stack



OSI (Open System Interconnect) Protocol Stack

- Application layer:

  handles the detailed specific of an application
- Presentation layer:

  translate, encode, compress and apply other transformation.
- Session layer:

  A session comprises a single request from a client or multiple two-way exchange of data.

In TCP/IP protocol stack these three layers are combined into Application layer.

OSI (Open System Interconnect) Protocol Stack

- Transport Layer:
  TCP (Transmission Control Protocol) and UDP (User Datagram Protocol)
- TCP:
  -- Ensure reliability
  --- Flow control
  -- Congestion control
- UDP:
  --- does not establish connection
  --- does not ensure in order delivery of packet
  --- does not ensure delivery of packets.

OSI (Open System Interconnect) Protocol Stack

- Network Layer:

  Handles network addressing.

- Data Link Layer:

  -- Medium Access Layer (MAC) protocol.

  -- CSMA/CA protocol

- Physical Layer:

  --- handles details signaling

  --- Ethernet (IEEE 802.3), WiFi (IEEE 802.11, 802.15, etc.)

TCP/IP protocol stack combines data link layer and physical layer into link layer protocol.

Internet Control Message Protocol (ICMP)

- Network layer protocol
- Used for

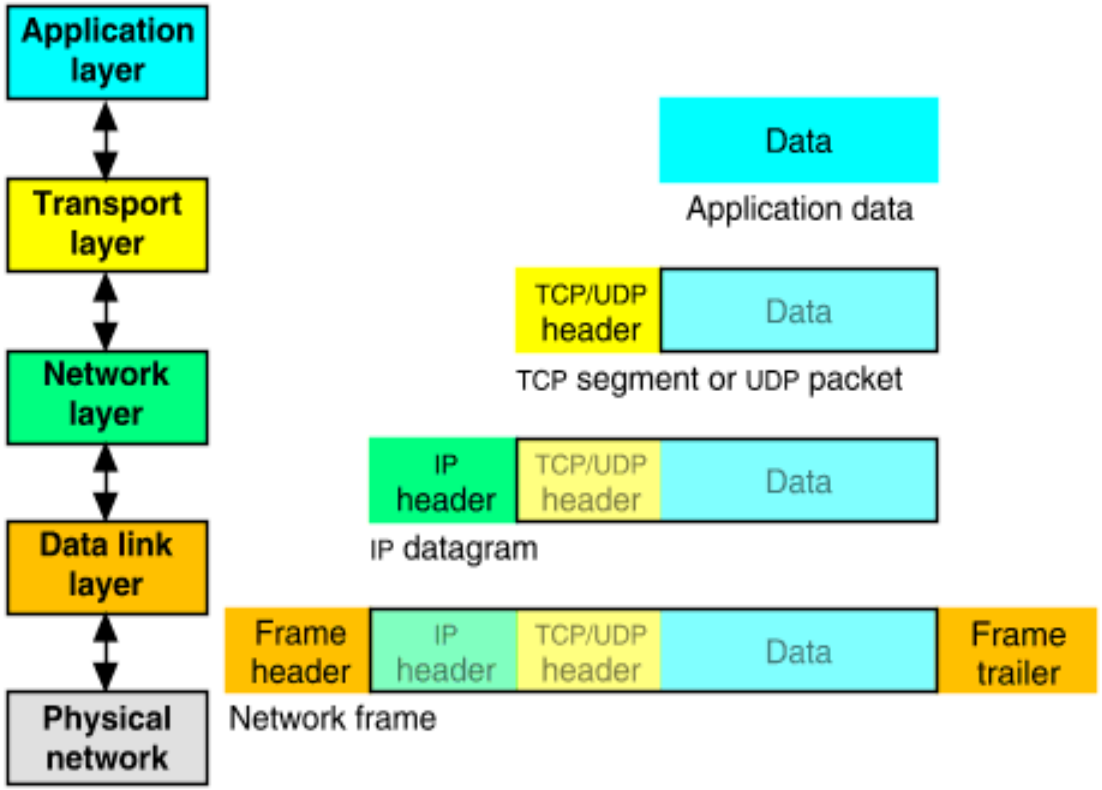
  ---Announce network error

  ---Announce network congestion (ICMP Source Quench)

  ---Assist troubleshooting (ICMP Echo)

  ---Announce timeout (ICMP Time Exceed)

Data Encapsulation



Transport Layer Protocol TCP

- TCP provides a reliable communication link between two hosts Through handshaking and acknowledgments.

- A TCP connection is full-duplex.

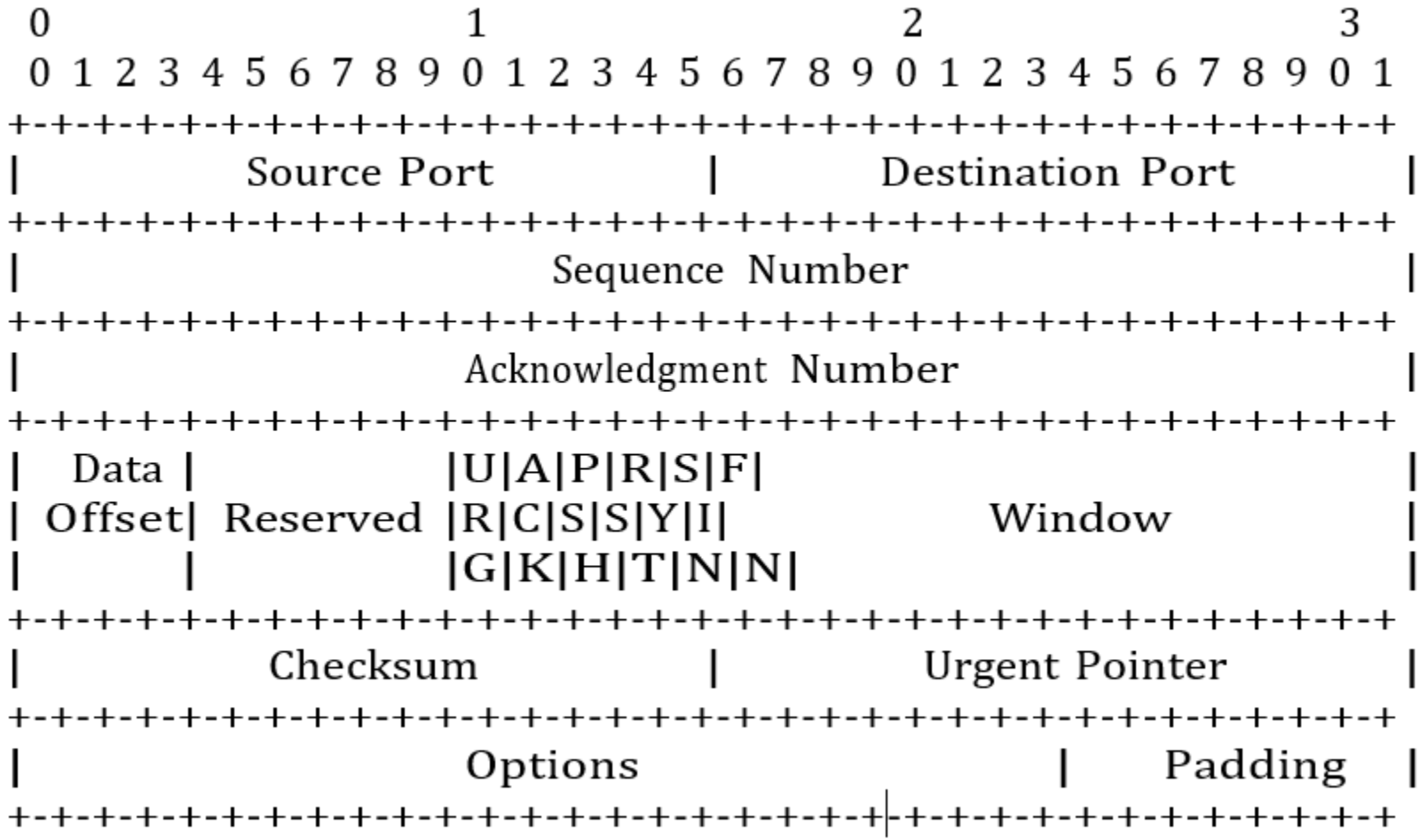
- Flow control allows the receiver's TCP to control the size of the segment dispatched by the sender's TCP.

  ---- Use the Window field of an acknowledgment packet.

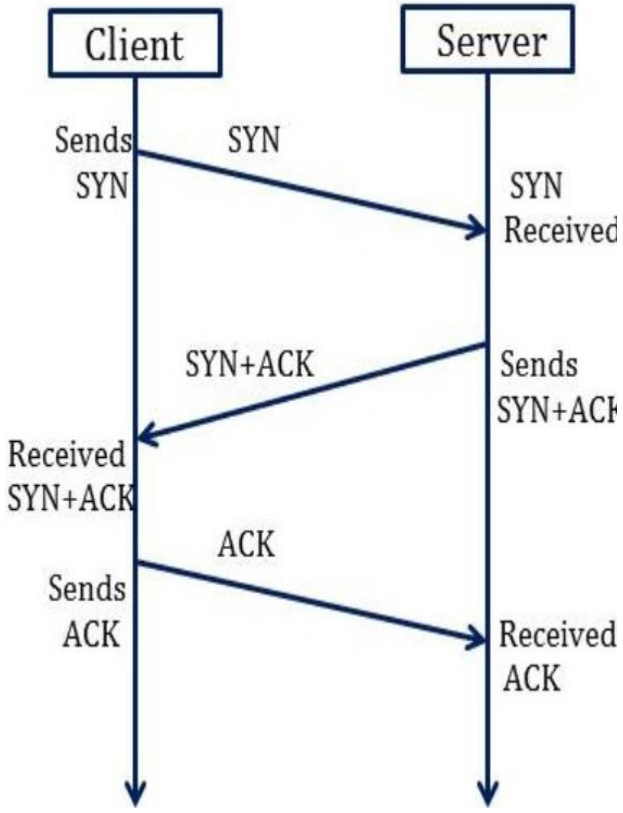
- Congestion control allows the sender's TCP to adjust the rate at which it places the packets on the wire based on the traffic congestion on the route.

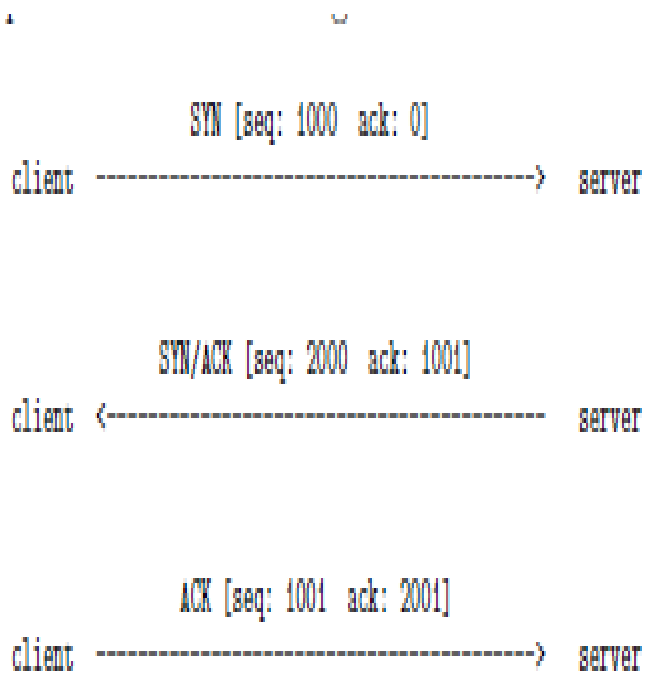
  ---Traffic congestion is detected either the non-arrival of an expected ACK packet or by the arrival of three identical ACK packets consecutively

TCP Header


0 1 2 3
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
Source Port
Destination Port
Sequence Number
Acknowledgment Number
Data Offset
Reserved
U A P R S F
R C S S Y I
G K H T N N
Window
Checksum
Urgent Pointer
Options
Padding

Three-way Handshaking





1. *C* sends to *S* a SYN packet. The Sequence Number in this TCP packet is a randomly generated number *M* .

1. S send back to *C* a SYN/ACK packet containing the expected next sequence number from *C*, the number *M* + 1, in *S*'s Acknowledgment Number field. This packet sent by *S* to *C* must also contain in its Sequence Number field another randomly generated number, *N*.

2. Now *C* must respond with an ACK packet with its Acknowledgment Number field containing *N* + 1

On-going Connection between A and B

1. Each endpoint in a TCP communication associates a byte count with the first byte of the outgoing bytes in each TCP segment.

2. This byte-count index is added to the initially sent ISN and placed in the Sequence Number field for an outgoing TCP packet.

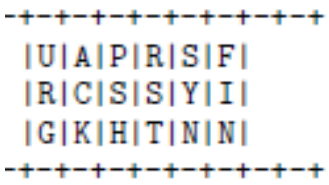
3. *B* receives these TCP segments and the Acknowledgment Number field of *B*'s ACK packets contains the index it expects to see in the Sequence Number field of the next TCP segment from *A*.

An application at *A* wants to send 100,000 bytes to an application running at *B:*

TCP breaks this up into 100 segments, each of size 1000 bytes. So *A*'s TCP sends to *B*'s TCP a packet containing the first 1000 bytes of data from the longer byte stream. The Sequence Number field of the TCP header for this outgoing packet will contain 0, which is the index of the first data byte in the outgoing segment in the 100,000 byte stream, plus the ISN used for the initiation of the connection. The Sequence Number field of the next TCP segment from *A* to *B* will be the sequence number in the first segment plus 1000, and so on.

Control Bits

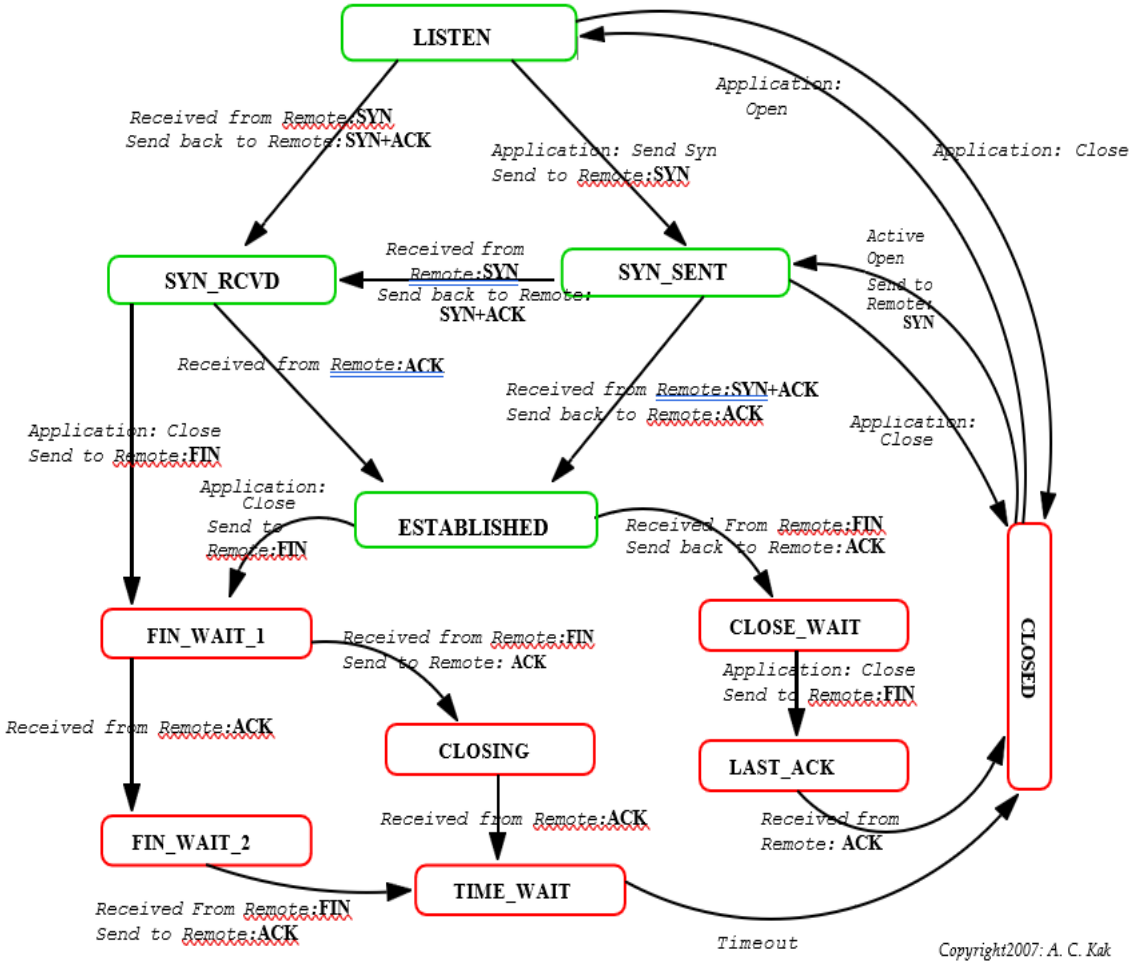
- The Control Bits field is 6 bits wide.
- 1st flag bit: URG means “URGENT” data. A packet whose URG bit is set can act like an interrupt with regard to the interaction between the sender TCP and the receiver TCP.
- 2nd flag bit: ACK means acknowledgment.
- 3rd flag bit: PSH means that the TCP segment to be put on the wire immediately
- 4th flag bit: RST means that the sender wants to reset the connection.
- 5th flag bit: SYN means synchronization of sequence numbers.
- 6th flag bit: FIN means that the sender wants to terminate the connection.



TCP flow control and the sliding window

- TCP sliding window determines the number of unacknowledged bytes that one system can send to another.
  --- The size of the send buffer.
  --- The size and available space in the receive buffer.
- It is not possible to send more bytes than that is available in the receive buffer.
- Sender TCP must wait to send more data until all bytes in the current send buffer are acknowledged by receiver TCP.
- Receiver TCP stores received data in a receive buffer. It acknowledges receipt of the data and *advertises* a new *receive window* to the sending system.
- After the receiving application retrieves data from the receive buffer, the receiving system can then advertise a receive window size that is equal to the amount of data that was read. Then, TCP on the sending system can resume sending data.
- If the receive buffer is full, then TCP advertises a receive window size of zero, and the sending system must wait to send more data.

TCP State Transition Diagram



Demonstration of Three-way Handshake

tcpdump -vvv -nn -i wlan0 -s 1500 -S -X -c 5 'src 10.185.37.87' or 'dst 10.185.37.87' and 'port 22'

ssh kak@shay.ecn.purdue.edu

Source machine IP: 10.185.37.87

Destination machine IP address (shay.ecn.purdue.edu) is 128.46.4.61.

Resource: tcpdump manpage:
https://www.tcpdump.org/manpages/tcpdump.1.html

Demonstration of Three-way Handshake

11:19:12.740733 IP (tos 0x0, ttl 64, id 37176, offset 0, flags [DF],
proto TCP (6), length 60)
10.185.37.87.47238 > 128.46.4.61.22: Flags [S], cksum 0x8849 (correct),
seq 2273331440, win 5840, options [mss 1460,sackOK,TS val 49207752 ecr 0,nop,wscale 7], length 0

```
0x0000:  4500 003c 9138 4000 4006 6661 80d3 b216    E..<.8@.@.fa....
0x0010:  802e 900a b886 0016 8780 48f0 0000 0000    ..........H.....
0x0020:  a002 16d0 8849 0000 0204 05b4 0402 080a    .....I..........
0x0030:  02ee d9c8 0000 0000 0103 0307              ............
```

11:19:12.744139 IP (tos 0x0, ttl 57, id 54821, offset 0, flags [DF],
proto TCP (6), length 64)
128.46.4.61.22 > 10.185.37.87.47238: Flags [S.], cksum 0xa52e (correct),
seq 2049315097, ack 2273331441, win 49560, options [nop,nop,TS val 549681759 ecr 49207752,mss 1428,nop,wscale 0,nop,nop,sackOK], length 0

```
0x0000:  4500 0040 d625 4000 3906 2870 802e 900a    E..@.%@.9.(p....
0x0010:  80d3 b216 0016 b886 7a26 1119 8780 48f1    ........z&    H.
0x0020:  b012 c198 a52e 0000 0101 080a 20c3 7a5f    z_
0x0030:  02ee d9c8 0204 0594 0103 0300 0101 0402    ................
```

Demonstration of Three-way Handshake

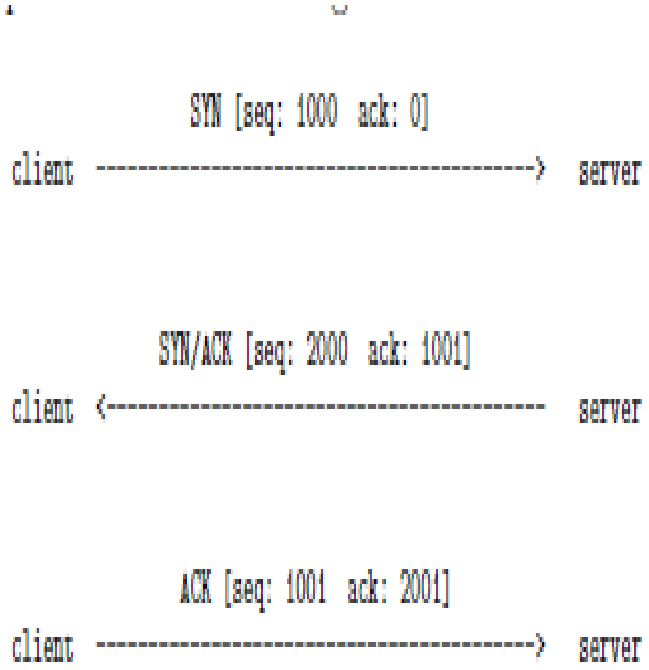
11:19:12.744188 IP (tos 0x0, ttl 64, id 37177, offset 0, flags [DF],
proto TCP (6), length 52)
10.185.37.87.47238 > 128.46.4.61.22: Flags [.], cksum 0xa744 (correct),
seq 2273331441, ack 2049315098, win 46, options [nop,nop,TS val 49207752
ecr 549681759], length 0

```
0x0000:  4500 0034 9139 4000 4006 6668 80d3 b216  E..4.9@.@.fh....
0x0010:  802e 900a b886 0016 8780 48f1 7a26 111a  H.z&..
0x0020:  8010 002e a744 0000 0101 080a 02ee d9c8  .....D..........
0x0030:  20c3 7a5f                                ..z_
```

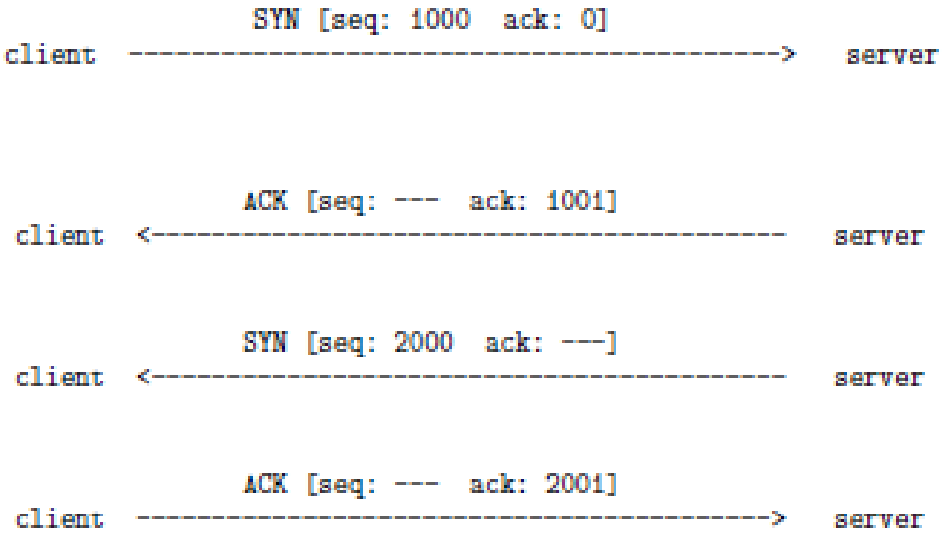
11:19:12.749205 IP (tos 0x0, ttl 57, id 54822, offset 0, flags [DF],
proto TCP (6), length 74)
128.46.4.61.22 > 10.185.37.87.47238: Flags [P.], cksum 0xf4f0 (correct),
seq 2049315098:2049315120, ack 2273331441, win 49560, options [nop,nop,TS
val 549681760 ecr 49207752], length 22

```
0x0000:  4500 004a d626 4000 3906 2865 802e 900a  E..J.&@.9.(e....
0x0010:  80d3 b216 0016 b886 7a26 111a 8780 48f1  ........z&      H.
0x0020:  8018 c198 f4f0 0000 0101 080a 20c3 7a60  .               z‘
0x0030:  02ee d9c8 5353 482d 322e 302d 5375 6e5f  ....SSH-2.0-Sun_
0x0040:  5353 485f 312e 312e 330a                 SSH_1.1.3.
```

Handshake Splitting to Establish TCP Connection

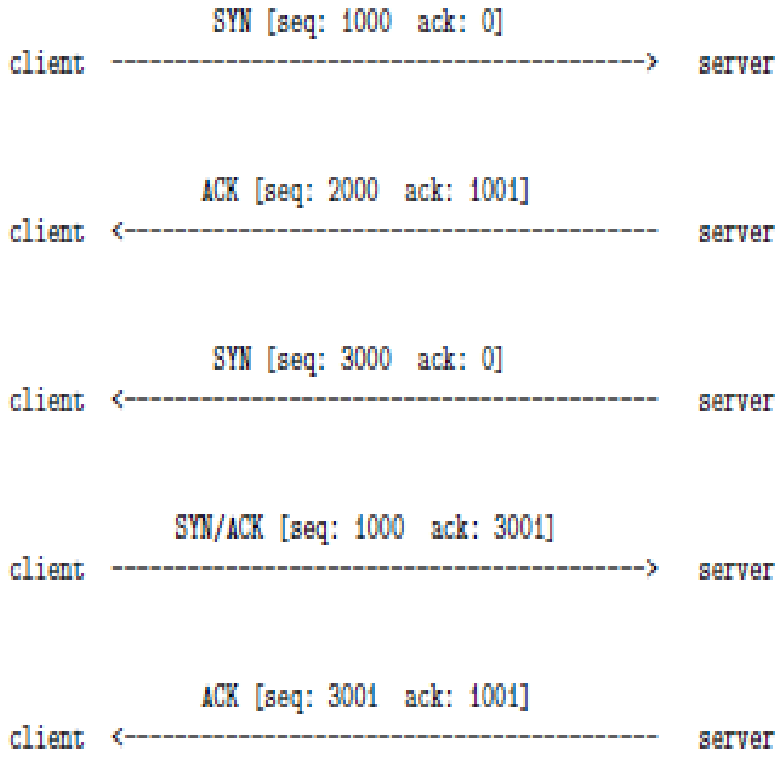


3-way handshake

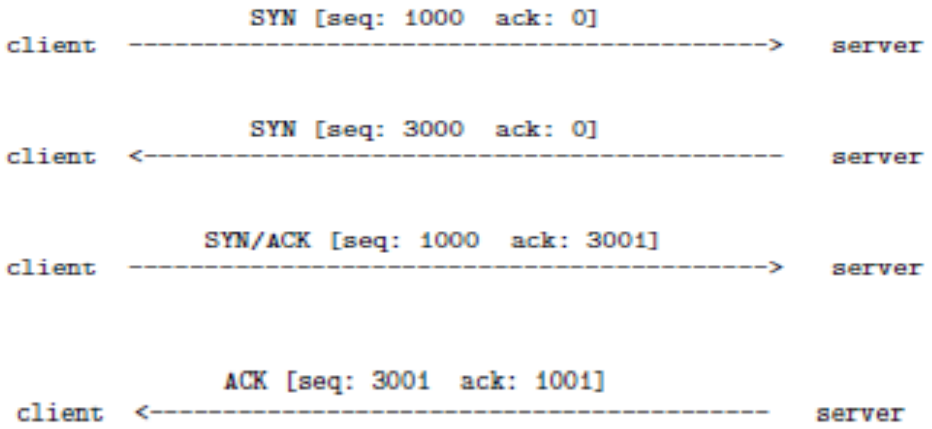


4-way handshake

Handshake Splitting to Establish TCP Connection



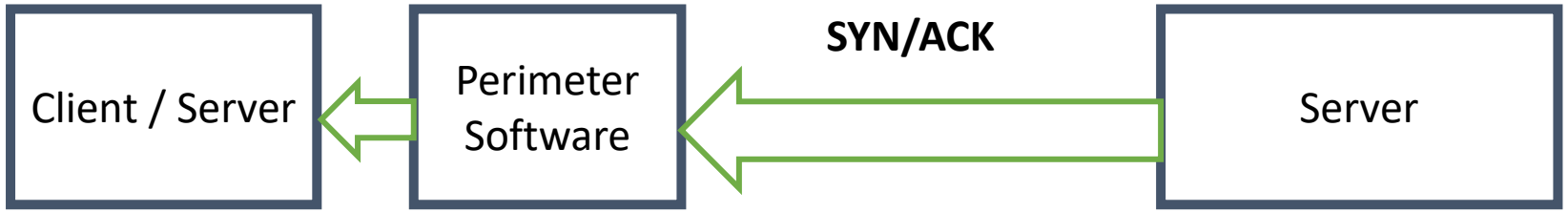
5-way handshake



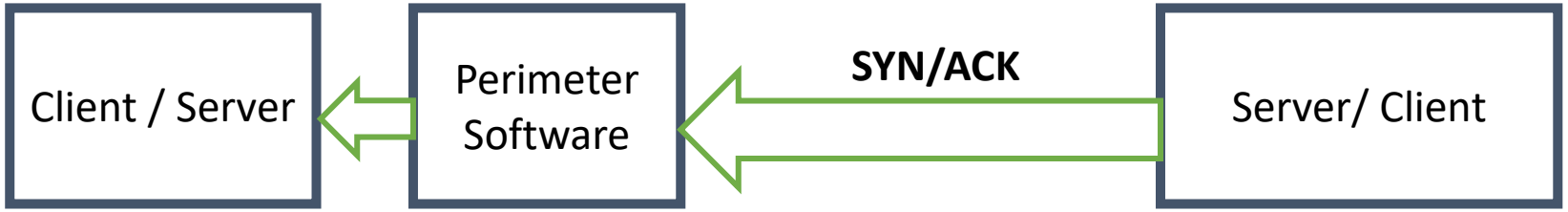
4-way handshake

Split-Handshake Attack

- 3-way handshake: Server sends SYN/ACK.
- 4-way/5-way handshake: client sends SYN/ACK



(a) Perimeter software is confirmed the direction data traffic from s server in 3-way handshaking

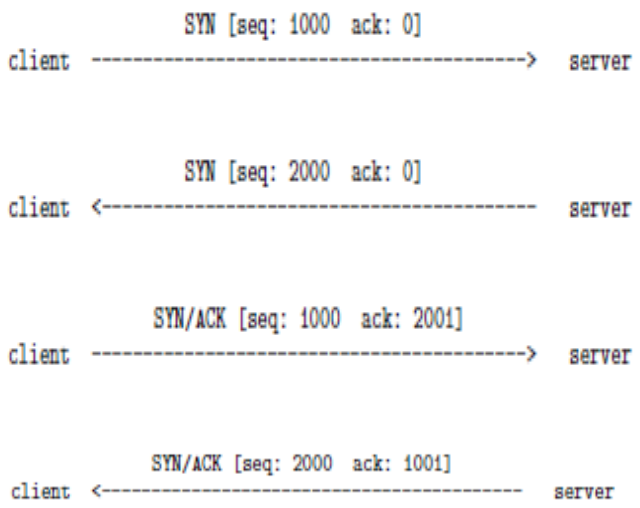


(b) Perimeter software is not confirmed the direction data traffic from a server in 4-way/5-way handshaking

Split-Handshake Attack

- Let's say that you've been "tricked" into clicking on an attachment that causes your machine to try to make a connection with a malicious server.
- Your computer will send a SYN packet to the server.
- Instead of sending back a SYN/ACK packet, the server sends back a SYN packet in order to establish a TCP connection through the split-handshake.
- Should this succeed, your intrusion prevention software and possibly even your firewall could become confused with regard to the security tests to be applied to the packets being sent over by the server.

Simultaneous Split Handshake



- Server must know the port of the client in advance.
- Split handshake attack does not occur in TCP not supporting split handshake.
- Split handshake is sometimes known as “SNEAK ACK ATTACK”

TCP Congestion Control

- Congestion is detected by

  1) an ACK is not received in a certain period of time, OR

  2) when three duplicate ACKs are received consecutively.

- The retransmission decision for a TCP segment is done on two different timescales:

  --- low traffic congestion uses RTT (Round Trip Time), order milliseconds.

  ---- high congestion uses RTO (Retransmission Timeout), order of a full second.

- CNWD:

  ---- used in congestion control.

  ---- defined in terms of SMSS (Sender Maximum Segment Size)

  ---- Initially set to 1.

  ---- a segment of size SMSS is sent per RTT.

Congestion Control based on RTT

- The sender TCP sends out a TCP segment of size= 1 SMSS (initial value of CNWD)
- If an ACK is received within an RTT, the sender TCP then sets the value of CWND field as

$$CWND = CWND + a$$

- If an ACK is *not* be received within an RTT, the value of CWND is changed to

$$CWND = CWND \times b$$

This is mechanism is known as ADDITIVE INCREASE MULTIPLICATIVE DECREASE (AIMD) algorithm.

When no ACK is received within an RTO, that indicates severe congestion.

Congestion Control based on RTO

- If an ACK is not received within the current value of RTO , the value placed in the CWND is reduced to 1 and RTO is doubled to 2 sec.
- If an ACK is not received again, the RTO is doubled, while the CWND value maintained at 1
- if an ACK is received within the current RTT, TCP switches back to the RTT timescale logic for congestion control.

Shrew DoS Attack

- Shrew DoS attack exploits the manner in which RTO is set and reset on a sender TCP.
- This attack targets the RTO at the sending TCP to set to minimum value of 1 sec.
- The attacker hits the sending TCP with a short burst of DoS packets for the duration of RTT.
- This artificially created congestion of duration RTT at the sending TCP will cause that host to reset its RTO to 1 second and the CWND value to 1 SMSS.
- In response, the sending TCP will send out one packet of length CWND and wait for the RTO of 1 sec for an ACK.
- the attacker sends another DoS burst at the end of that 1 sec and the sending TCP will double the RTO to 2 seconds while keeping CWND at 1.
- If the attacker persists in hitting the victim TCP with these short duration DoS bursts at every new value of RTO, the TCP flow emanating from the victim machine would virtually come to a halt.

Shrew DoS Attack

- It hits a targeted host with periodic brusty DoS traffic. Due to the on/off ratio of the DoS traffic, it would not be detectable by a traffic monitor that is looking for heavy traffic.

SYN Flooding Attack

- a method that a hostile client can use to conduct a denial-of-service (DoS) attack on a computer server.
- The hostile client repeatedly sends a TCP SYN to every port on the server using a fake IP address.
- The server responds to each such attempt with a SYN/ACK response from each open port and with an RST response from each closed port.
- The hostile client never sends back the expected ACK packet.
- As soon as a connection for a given port gets timed out, another SYN request arrives for the same port from the hostile client.
- The server will send SYN/ACK in response to a SYN packet upto a maximum limit. After that all SYN request will be discarded.

Defense against SYN Flooding Attack

- Victim server can set the firewall rule to block the intruder IP.
- It will be difficult to defend if multiple IPs are involved.
  - Server can limit the rate of incoming SYN packets.

- The transmission by a hostile client of SYN requests for the purpose of finding open ports is also called SYN scanning. A hostile client always knows a port is open when the server responds with a SYN/ACK.

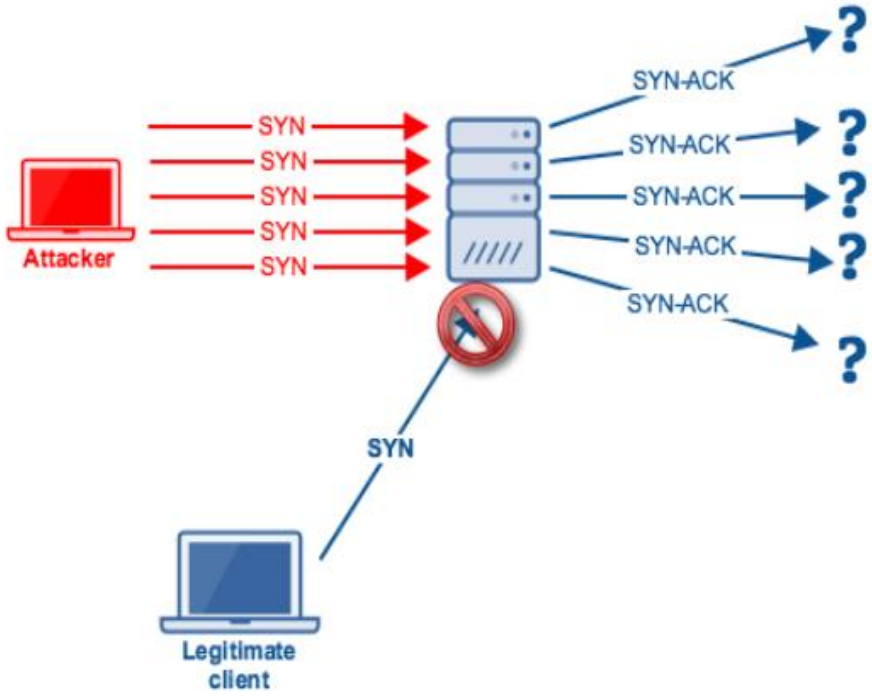
IP Source Address Spoofing for SYN FLood DoS Attacks

- refers to an intruder using one or more forged source IP addresses to launch, say, a TCP SYN flood attack on a host in another network.

- If the IP is legitimate

---- A legitimate host will be blocked to get service.

---- slow down the performance of the legitimate victim host due to the processing of huge number of SYN/ACK packets.



IP Spoofing to Establish One-way Connection

- *B* is a server executing code for a client *A*.
- *A* and *X* are in the same LAN and *X* want to establish a connection with *B*.

$$X\ (posing\ as\ A) \quad ---> \quad B \quad : \quad SYN$$
$$(sequence\ num:\ M)$$

$$B \quad ---> \quad A \quad : \quad SYN/ACK$$
$$(sequence\ num:\ N,\ \ acknowledgment\ num:\ M{+}1)$$

- *X* does not receive SYN/ACK and sends back an ACK with N+1.
- Connection is established and *X* can execute code in **B**.

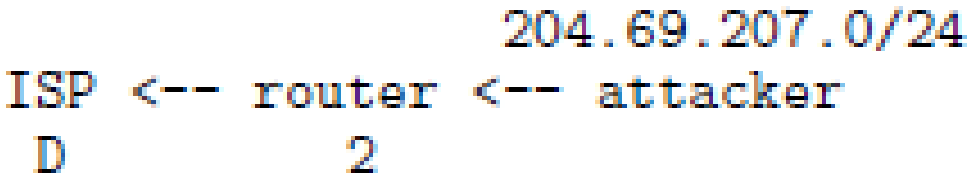
IP Spoofing to Establish One-way Connection

- *A* will receive SYN/ACK packet.
- *B* keeps busy *A* by creating SYN flooding attack.

- With the bad design of Pseudo-random number generator (PRNG) and the birthday paradox, it is possible to guess the correct sequence number for ACK.

Thwarting IP Source Address Spoofing with BCP (Best Current Practice) 38 / Ingress Filtering

- Ingress filtering check the source IP address of outgoing packet against the range that corresponds to the network address of the router.



- If the attacker use an IP address within the range 204.69.207.0/24, network admins can track that incident.

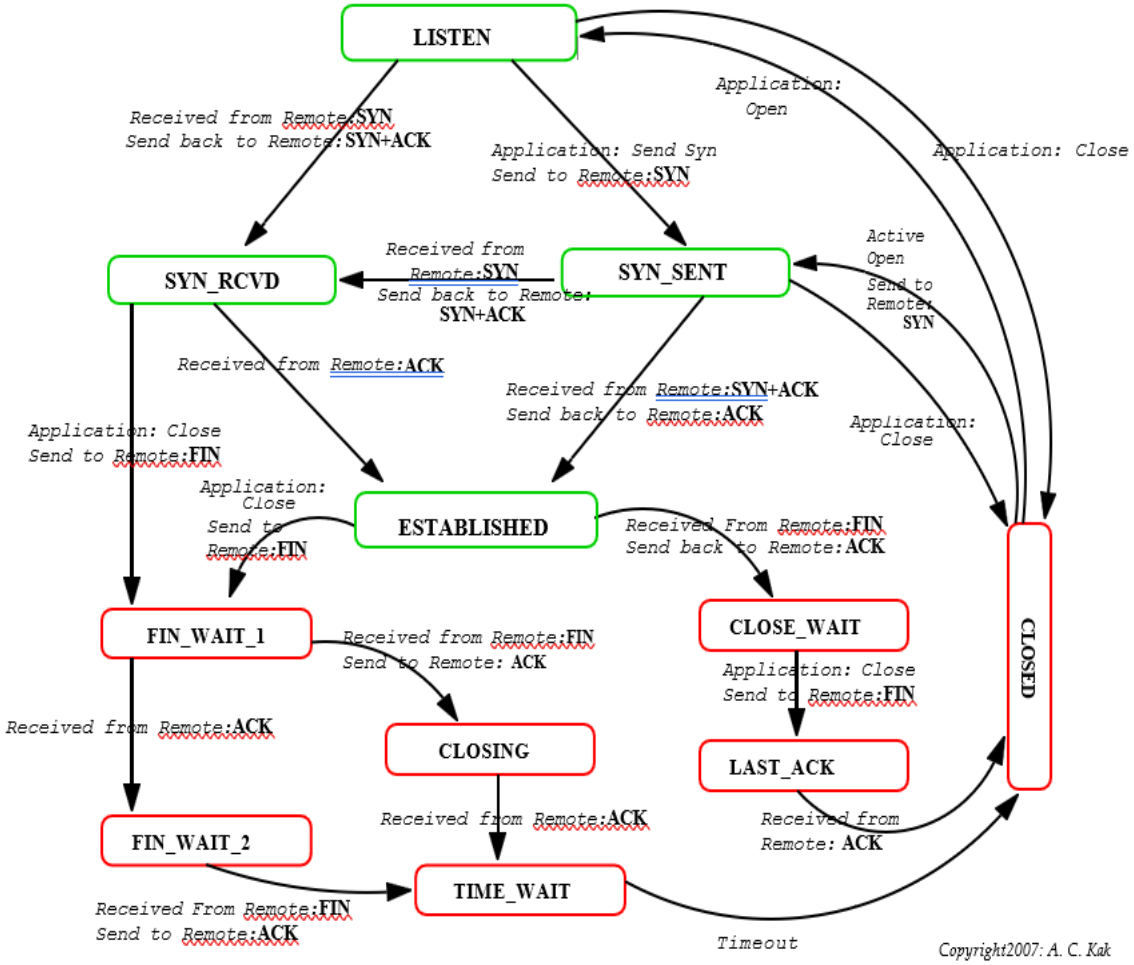
Thwarting IP Source Address Spoofing with BCP (Best Current Practice) 38 / Ingress Filtering

- If ingress filtering is not used, the source IP spoofing is difficult if the packet crosses a router.

- If the spoofed IP is invalid, then a router will detect and send ICMP host unreachable error message.
- If the spoofed IP is legitimate, the corresponding host returns RST message.

- A determined adversary, especially one who has the cooperation of an ISP and, possibly the state itself, could cause a lot of harm in a victim network.

TCP State Transition Diagram



Trouble-shooting using Netstat

- Local has no route to remote

  --- TCP got stuck in SYN_SENT and SYN_RCVD

netstat –nr

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags   MSS Window  irtt Iface
192.168.1.0     0.0.0.0         255.255.255.0   U         0 0          0 wlan0
169.254.0.0     0.0.0.0         255.255.0.0     U         0 0          0 lo
0.0.0.0         192.168.1.1     0.0.0.0         UG        0 0          0 wlan0
```

Ping the router address.

Trouble-shooting using Netstat

- **Local to remote route is down**

--- TCP got stuck in **SYN_SENT, FIN_WAIT1** and **CLOSING**

traceroute 128.46.144.10

```
192.168.1.1   (148 Creighton Road)
    -> 74.140.60.1    (a DHCP server at insightbb.com)
    -> 74.132.0.145   (another DHCP server at insightbb.com)
    -> 74.132.0.77    (another DHCP server at insightbb.com)
    -> 74.128.8.201   (some insightbb router, probably in Chicago)
    -> 4.79.74.17     (some Chicago area Level3.net router)
    -> 4.68.101.72    (another Chicago area Level3.net router)
    -> 144.232.8.113  (SprintLink router in Chicago)
    -> 144.232.20.2   (another SprintLink router in Chicago)
    -> 144.232.26.70  (another SprintLink router in Chicago)
    -> 144.228.154.166  (where?? probably Sprint's Purdue drop)
    -> 128.46.144.10  (RVL4.ecn.purdue.edu)
```

Trouble-shooting using Netstat

The local machine is under DoS attack can be checked using

netstat -n

```
anu@anu-Vostro-5471:~$ netstat -n
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 172.20.10.2:45744       34.117.65.55:443        ESTABLISHED
tcp        0      0 172.20.10.2:47578       34.149.100.209:443      ESTABLISHED
tcp        0      0 172.20.10.2:45732       34.117.65.55:443        ESTABLISHED
udp        0      0 172.20.10.2:68          172.20.10.1:67          ESTABLISHED
```

Trouble-shooting using Netstat

Network interface card statistics check using

Netstat –ni

```
anu@anu-Vostro-5471:~$ netstat -ni
Kernel Interface table
Iface             MTU    RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
enp2s0           1500        0      0      0 0             0      0      0      0 BMU
enxda1c7976069f  1500    15736      0      3 0         17609      0      0      0 BMRU
lo              65536     7819      0      0 0          7819      0      0      0 LRU
wlp3s0           1500        0      0      0 0             0      0      0      0 BMU
```

Acknowledgment

- This lecture slide is prepared with the help of lecture notes prepared by Avinash KaK for Computer and Network Security Course.

  https://engineering.purdue.edu/kak/compsec/

Lecture Material

Lecture 16 from https://engineering.purdue.edu/kak/compsec/