Network and Port Scanning

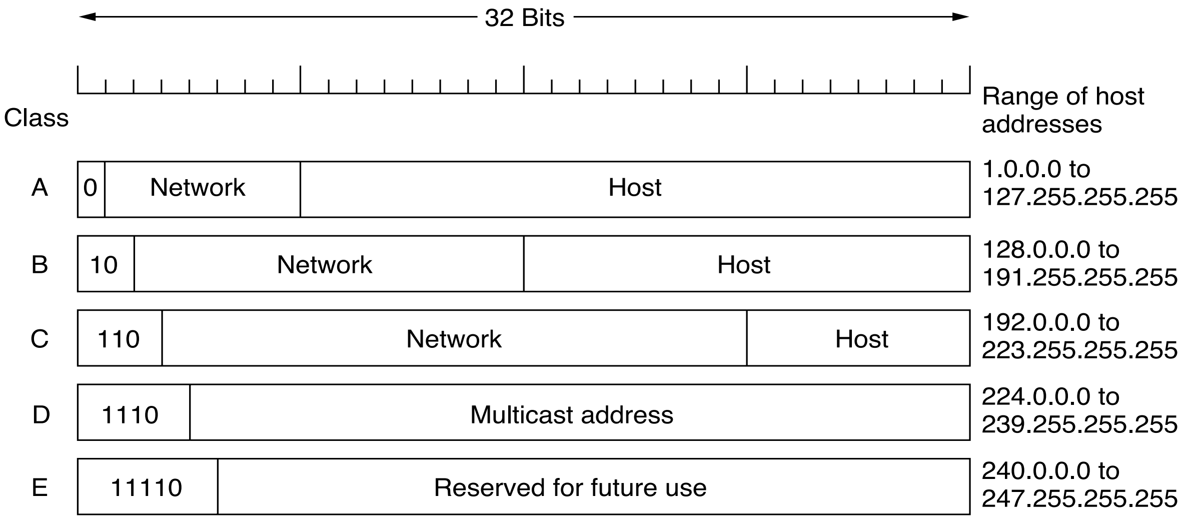
Chien-Chung Shen

cshen@cis.udel.edu

Host Discovery

- One of very first steps in **network reconnaissance mission** to reduce a (sometimes huge) set of IP ranges into a list of active or interesting hosts
  - administrator uses an ICMP ping to locate hosts on internal network
  - external penetration uses a diverse set of "probes" in an attempt to evade firewall restrictions
- Aka "ping" scan, but goes beyond `ICMP echo request` packets

IP Address



32 Bits

Class			Range of host addresses
A	0	Network / Host	1.0.0.0 to 127.255.255.255
B	10	Network / Host	128.0.0.0 to 191.255.255.255
C	110	Network / Host	192.0.0.0 to 223.255.255.255
D	1110	Multicast address	224.0.0.0 to 239.255.255.255
E	11110	Reserved for future use	240.0.0.0 to 247.255.255.255

- `$ nslookup stimpy.cis.udel.edu`
- `128.4.31.17` is a **class B** address
- `strauss.udel.edu 128.175.13.74`
- `$ nmap -sL 128.4.0.0/16 > a`
- `Locate 128.4.21.33`

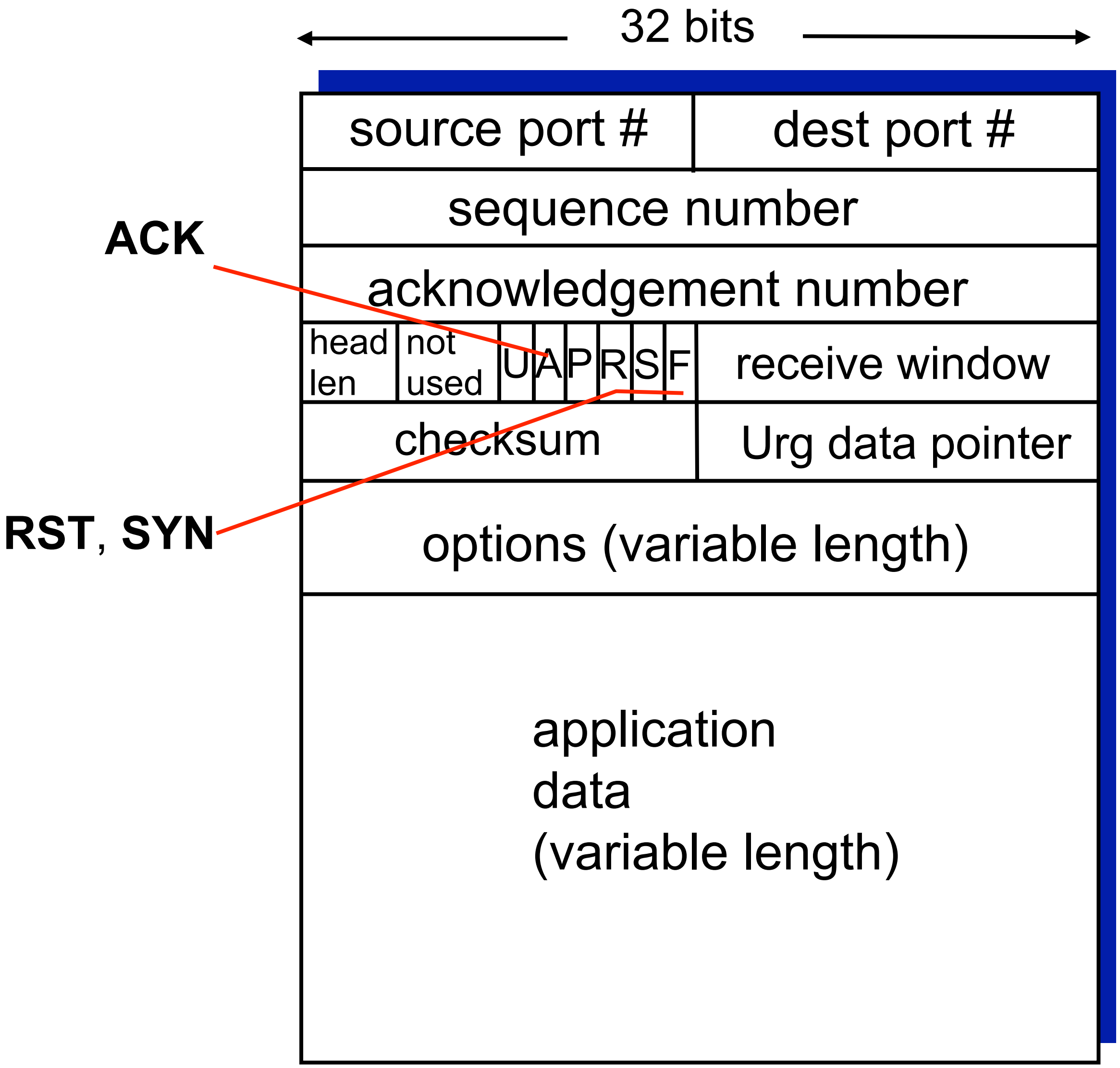
Port Scanning

- In TCP/IP, every (network) service on a machine is assigned a **port (number)**
- On Unix machine, ports assigned to standard services are listed in **`/etc/services`**
  - a (Unix) process listens on the port for incoming connection requests
  - what is the port # of **`ssh`**?
- **Goal** of port scanning: find out which ports are **open**, **closed**, or **filtered**
  - *e.g.*, **find out if a remote host is providing a service that is vulnerable to buffer overflow attack**
  - port scanning may involve all 65,535 ports or only the ports that are well-known to provide services vulnerable to security-related exploits

Port Scanning

- A port is **open** on a machine if there is a running (server) process on the machine and the port is assigned to this process
  - if a port on a remote host is open for incoming connection requests and you send it a **`SYN`** packet, the remote host will respond back with a **`SYN+ACK`** packet
- A port is **filtered** if packets passing through that port are subject to **filtering rules** of a **firewall**
  - if a port is filtered with something like an **`iptables`** based packet filter and you send it a **`SYN`** packet or an ICMP **`ping`** packet, you may not get back anything at all
- If a port on a remote host is **closed** and you send it a **`SYN`** packet, the remote host will respond back with a **`RST`** packet

TCP Segment



32 bits

source port #

dest port #

sequence number

acknowledgement number

head len

not used

U A P R S F

receive window

checksum

Urg data pointer

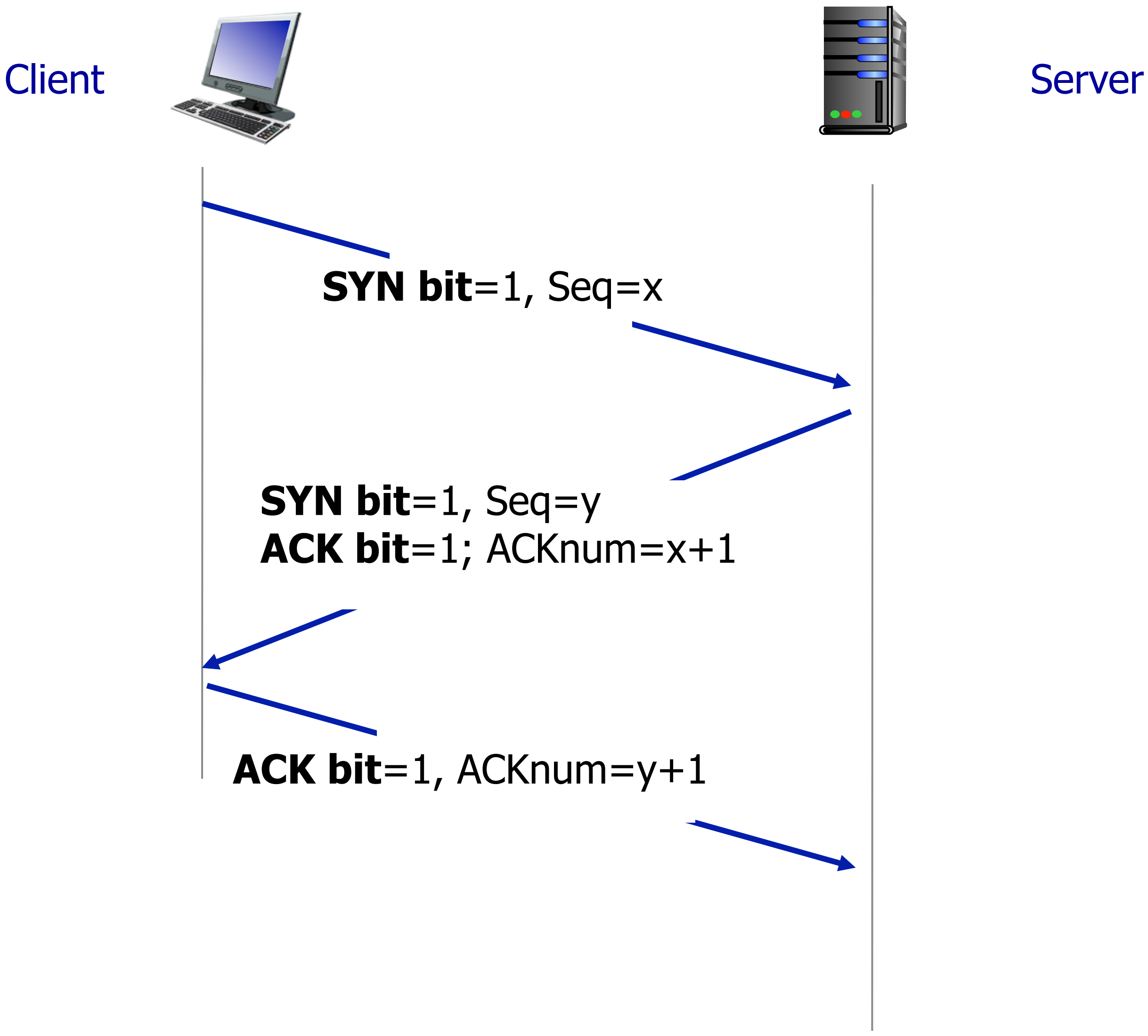
options (variable length)

application
data
(variable length)

ACK

RST, SYN

TCP 3-Way Handshake



Client

Server

SYN bit=1, Seq=x

SYN bit=1, Seq=y
ACK bit=1; ACKnum=x+1

ACK bit=1, ACKnum=y+1

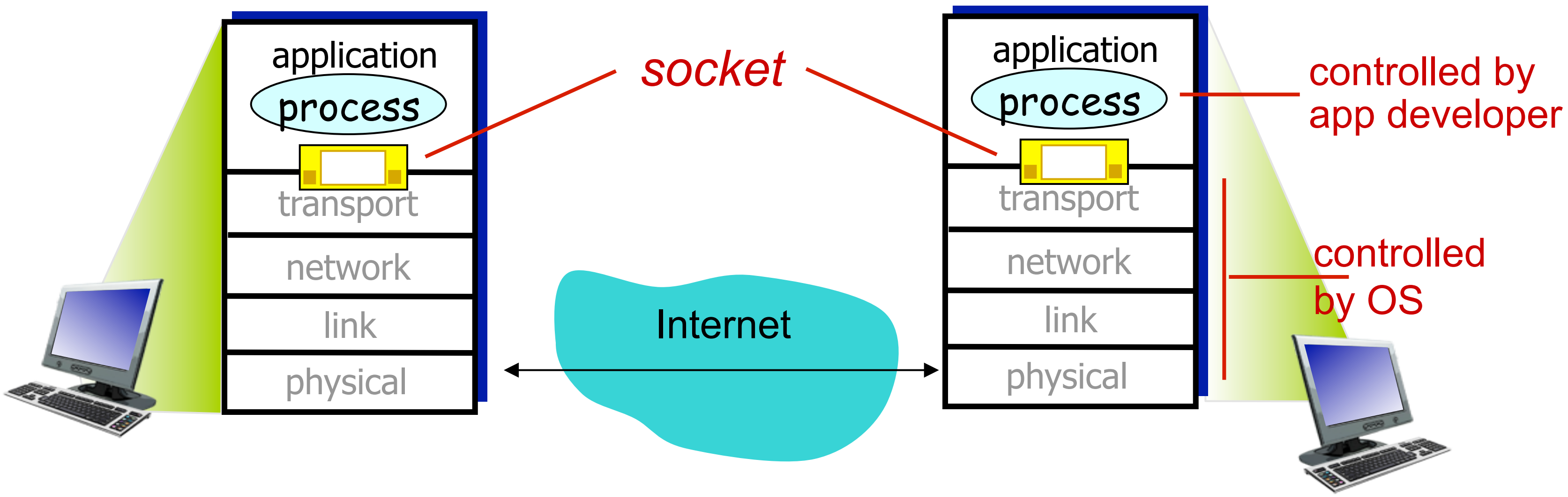
connect (TCP) Scan

- Check out **man** page of **connect()**

```
#include <sys/socket.h>
int connect(int socketfd,  // file descriptor from socket()
        const struct sockaddr *address, // server IP address
        socklen_t address_len);
```

- *A* call to **connect()** if successful completes a **three-way handshake** for a TCP connection with a server
- In a typical use of **connect()** for port scanning, if the connection succeeds, the port scanner immediately closes (via **close()**) the connection (having ascertained that the port is open) to avoid DoS attack

Socket



- "door" between application process and TCP transport protocol

Port Scanner in Python

- http://www.pythonforbeginners.com/code-snippets-source-code/port-scanner-in-python
- Using built-in **socket** module
- **`$ python port-scanner.py`**
- Nmap module/library in Python
  - http://xael.org
  - https://pypi.python.org/pypi/python-libnmap/0.6.1

Port Scanning with (TCP) **SYN** (1)

- Most popular form of port scanning
- Open TCP connection via three-way handshake
  - **SYN -> SYN+ACK -> ACK**
- In port scanning with **SYN** packets, scanner sends out **SYN** packets to different ports of a remote machine. When scanner receives **SYN+ACK** packet in return for a given port, scanner can be sure that the port on remote machine is open
  - it is the "duty" of a good port-scanner to **immediately** send back **RST** packet in response to received **SYN+ACK** packet so that the **half-open** TCP connection at remote machine is closed immediately

Port Scanning with TCP **SYN** (2)

- When a target machine receives a **SYN** packet for a **closed** port, it sends back an **RST** packet back to the sender
- When a target machine is protected by a packet-level firewall, it is the firewall rules that decide what the machine's response will be to a received **SYN** packet

`connect()` vs. `SYN`

- SYN
  - port scanner generates **raw** IP packets itself, and monitors for responses
  - aka "half-open scanning", because it never actually opens a full TCP connection
  - SYN scan has advantage that individual services never actually receive a connection (less intrusive?)
- **`connect()`**
  - use operating system's network functions
  - full TCP connection established

UDP Scan (1)

- **SYN** packet is a TCP concept
- In a UDP scan, if a UDP packet is sent to a port that is **not open**, the remote machine will respond with an ICMP **port-unreachable** message. So the **absence** of a returned message can be inferred *as a sign* of an **open** UDP port
- A packet filtering firewall at a remote machine may prevent the machine from responding with an ICMP error message **even when a port is closed**

UDP Scan (2)

- Send application-specific UDP packets, hoping to generate application layer response
  - *e.g.*, sending DNS query to port 53 will result in a response, if DNS server is present
- limited to scanning ports for which an application specific probe packet is available

nmap Network Mapper

- Open-source **nmap** stands for "network mapper" ([**http://nmap.org**](http://nmap.org))
- **nmap** is more than just a port scanner
  - listing open ports on a network
  - trying to construct an inventory of all services running in a network
  - trying to detect as to which operating system is running on each machine
- **nmap** can carry out TCP SYN scan, TCP **connect()** scans, UDP scans, ICMP scans, *etc.*

`nmap`

- As listed in manpage, **`nmap`** comes with a large number of options for carrying out different security scans of a network
- **`-sT`**: carries out a TCP **`connect()`** scan
- **`-sU`**: sends a dataless UDP header to every port (state of the port is inferred from the ICMP response packet [if there is such a response at all])

nmap

- **-sP**: "ping scanning" to determine which machines are up in a network
  - **nmap** sends out ICMP echo request packets to every IP address in a network. Hosts that respond are up
  - But this does not always work since many sites now block echo request packets. To get around this, **nmap** can also send a TCP **ACK** packet to (by default) port 80. If the remote machine responds with an **RST** back, then that machine is up
  - Another possibility is to send the remote machine a **SYN** packet and waiting for an **RST** or a **SYN/ACK**. For root users, **nmap** uses both ICMP and ACK techniques in parallel. For non-root users, only the TCP **connect()** is used
- **-sV**: "version detection"
  - After **nmap** figures out which TCP and/or UDP ports are open, it next tries to figure out what service is actually running those ports
  - In addition to determine the service protocol (http, ftp, ssh, telnet, *etc.*), nmap also tries to determine the application name (such as Apache httpd, ISC bind, Solaris telnetd, *etc.*), version number, *etc.*

Port Scan Examples

- `(sudo) nmap -sS localhost`
  - **SYN** scan
- `nmap -sS stimpy.cis.udel.edu`
- `nmap -sS -A stimpy.cis.udel.edu`
  - **aggressive or advanced**
- If the target machine has the DenyHosts shield running and you repeatedly scan that machine with '`-A`' turned on, your IP address may **become quarantined** on the target machine (assuming that port 22 is included in the range of the ports scanned). When that happens, you will **not** be able to SSH into the target machine

nmap

- By default, **nmap** first **pings** a remote host in a network before scanning the host. The idea is that if the machine is down, why waste time by scanning all its ports
- Since many sites now block/filter ping echo request packets, this strategy may bypass machines that may otherwise be up in a network
- To change this behavior, the following **nmap** may produce richer results
  - `nmap -sS -A -P0 <host>`
  - `-P0`: skip pinging

nmap

- **nmap** can make good guess of the OS running on the target machine by using **TCP/IP stack fingerprinting**
- It sends out a series of TCP and UDP packets to the target machine and examines content of returned packets for values in various header fields, including sequence number, initial window size, *etc*. Based on these values, **nmap** then constructs an OS "signature" of the target machine and sends it to a database of such signatures to make a guess about the OS running on the target machine