NOTES

- *structure of applications:*
    - *client-server-> server: always-on host*
    - *peer-to-peer (P2P) -> no always-on server*
- *client process: process that initiates communication*
- *server process: process that waits to be contacted*
- identifier includes both IP address and port numbers
- protocols :
    - *open protocols -> defined in RFCs*
    - *allows for interoperability e.g., HTTP, SMTP*
    - *proprietary protocols: e.g., Skype*
- *IP/UDP datagrams with same dest. port #, but different source IP addresses and/or source port numbers will be directed to same socket at receiving host*
- host uses *IP addresses & port numbers* to direct segment to appropriate socket
- Multiplexing, demultiplexing: based on segment, datagram header field values
- **UDP:** demultiplexing using destination port number (only)

- **TCP:** demultiplexing using 4-tuple: source and destination IP addresses, and port numbers
- Multiplexing/demultiplexing happen at *all* layers
- receiver cannot know if its last ACK/NAK received OK at sender
- TCP uses this approach to be NAK-free
- TCP uses cumulative ACK
- UDP size 8 byte , TCP size 20 byte
- *timeout(n):* retransmit packet n and all higher seq # packets in window
- Flow control is a mechanism to the calamity of a receiver being over-run by a sender that is sending too fast – it allows the RECEIVER to explictly control the SENDER so sender won't overflow receiver's buffer by transmitting too much, too fast

Congestion:

informally: "too many sources sending too much data too fast for *network* to handle"

manifestations:

long delays (queueing in router buffers)

packet loss (buffer overflow at routers)

congestion control: too many senders, sending too fast	flow control: one sender too fast for one receiver

- TCP RENO -> بزود بالضعف
- TCP Tahoe -> بزود بمقدار واحد ال segmant

- two control-plane approaches:
  - *traditional routing algorithms:* implemented in routers
  - *software-defined networking (SDN)*: implemented in (remote) servers
- *Data plane: use forwarding function (how datagram arriving on router input port is forwarded to router output port)*
- Control plane : how datagram is routed among routers along end-end path from source host to destination host
- large IP datagram divided (“fragmented”) within net

- one datagram becomes several datagrams
- “reassembled” only at final destination

- *interface:* connection between host/router and physical link
- subnet part -> high order bits
- host part -> low order bits
- *what’s a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other *without intervening router*
- How does a *host* get IP address? hard-coded by system admin in a file
  Or DHCP (plug-and-play)
- *DHCP overview:*
  - host broadcasts “DHCP discover” msg [optional]
  - DHCP server responds with “DHCP offer” msg [optional]
  - host requests IP address: “DHCP request” msg
  - DHCP server sends address: “DHCP ack” msg
    - لو الرسمه من غير switch زي slide هيبقى كله broadcast

unicast هيبقوا (offer , ack) ال switch لكن لو فيه

- DHCP can return
  - IP
  - address of first-hop router for client
  - name and IP address of DNS sever
  - network mask
- DHCP use UDP
- *network* get **subnet part of IP** ? gets allocated portion of its provider ISP's address space
- how does an ISP get block of addresses? Using ICANN
- ICANN :
  - allocates addresses
  - manages DNS
  - assigns domain names, resolves disputes
- باختصار ISP بتستخدم ICANN علشان تجيب Range of IPs وبتديه ل DHCP يوزعه
- NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) to (NAT IP address, new port #)
  remote clients/servers will respond using (NAT IP address, new port #) as destination addr
- *remember (**in NAT translation table**)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) with corresponding (source IP address, port #) stored in **NAT table**

NAT translation table

WAN side addr	LAN side addr

- NAT use in network layer only
- IP types :
  - Private
  - Public
  - APIPA
- *IPv6 datagram format:*
  - fixed-length 40 byte header

    - no fragmentation allowed
- IPv6 -> *checksum* is removed
- IPv6 -> ICMPv6 is added in options (multicast)
- *tunneling:* IPv6 datagram carried as *payload* in IPv4 datagram among IPv4 routers
- how do **forwarding tables** (destination-based forwarding) or **flow tables** (generalized forwarding) computed?   by the control plane
- Logically centralized control plane

A distinct (typically remote) controller interacts with local control agents (CAs) in routers to compute forwarding tables

- Per-router control plane

    Individual routing algorithm components *in each and every router* interact with each other in control plane to compute forwarding tables
- Routing -> control plane (software)
- Forwarding -> data plane (hardware)
- *destination-based forwarding:* forward based only on destination IP address (traditional)

- *generalized forwarding:* forward based on any set of header field values
- *longest prefix matching :* when looking for forwarding table entry for given destination address, use *longest* address prefix that matches destination address.
- switching rate: rate at which packets can be transfer from inputs to outputs
    - often measured as multiple of input/output line rate
- three types of switching fabrics : memory , bus , crossbar
- Switching via memory speed limited by memory bandwidth
- *bus contention:* switching speed limited by bus bandwidth
- Switching via interconnection network (crossbar) fragmenting datagram into fixed length cells, switch cells through the fabric.
- *queueing delay and loss due to input/output port buffer overflow*
- fabric slower than input ports combined
- *buffering* required when datagrams arrive from fabric faster than the transmission rate
- *scheduling discipline* chooses among queued datagrams
- Datagram (packets) can be lost due to congestion, lack of buffers
- Priority scheduling – who gets best performance, network neutrality

- *discard policy:* if packet arrives to full queue: who to discard?
  - *tail drop:* drop arriving packet
  - *priority:* drop/remove on priority basis
  - *random:* drop/remove randomly
- Scheduling policies:
  - Priority : send highest priority queued packet
  - *Round Robin (RR) : cyclically scan class queues, sending one complete packet from each class*
  - *Weighted Fair Queuing (WFQ): each class gets weighted amount of service in each cycle*

routing protocols

- path selection
- RIP, OSPF, BGP

IP protocol

- addressing conventions
- datagram format
- packet handling conventions

ICMP protocol

- error reporting
- router "signaling"

- *routing algorithm:* algorithm that finds that least cost path
- *Dijkstra's algorithm*

computes least cost paths from one node ('source") to all other nodes

(gives *forwarding table* for that node)

- forwarding table configured by both intra- and inter-AS routing algorithm
    - intra-AS routing determine entries for destinations within AS
    - inter-AS & intra-AS determine entries for external destinations
- OSPF "advanced" features
    - *security:* all OSPF messages authenticated (to prevent malicious intrusion)
    - multiple same-cost paths allowed (only one path in RIP)
        - for each link, multiple cost metrics for different TOS (e.g., satellite link cost set low for best effort ToS; high for real-time ToS)
    - integrated uni- and multi-cast support:
        - Multicast OSPF (MOSPF) uses same topology data base as OSPF
    - hierarchical OSPF in large domains.
- OSPF -> metrics
- BGP -> *policy*
- BGP is a "path vector" protocol
- OSPF -> link-state

- *Policy-based routing:*
  - gateway receiving route advertisement uses *import policy* to accept/decline path (e.g., never route through AS Y).
  - AS policy also determines whether to *advertise* path to other other neighboring ASes

- BCP MESSAGES
  - OPEN: opens TCP connection to remote BGP peer and authenticates sending BGP peer
  - UPDATE: advertises new path (or withdraws old)
  - KEEPALIVE: keeps connection alive in absence of UPDATES; also ACKs OPEN request
  - NOTIFICATION
- ICMP message: type, code plus first **8 bytes** of IP datagram causing error
- SNMP -> بيعرف كل كبيره و صغيره عن راوتر و هكر بيستخدموه
- Link Layer Services
  - *flow control:*

pacing between adjacent sending and receiving nodes

- *error detection*:
    - errors caused by signal attenuation, noise.
    - receiver detects presence of errors:
        - signals sender for retransmission or drops frame
- *error correction:*

  receiver identifies *and corrects* bit error(s) without resorting to retransmission
- *half-duplex and full-duplex*

  with half duplex, nodes at both ends of link can transmit, but not at same time

- Link Layer (reliable as TCP):-
    - in each and every host
    - link layer implemented in “adaptor” (aka *network interface card* NIC) or on a chip
- 48 bit MAC address (for most LANs) burned in NIC ROM, also sometimes software settable
- IP used for layer 3 (network layer) forwarding
- each adapter on LAN has unique LAN address

- address depends on IP subnet to which node is attached
- MAC address allocation administered by IEEE
- *ARP table:* each IP node (host, router) on LAN has table
  - IP/MAC address mappings for some LAN nodes:
  - < IP address; MAC address; TTL>
  - TTL (Time To Live): time after which address mapping will be forgotten (typically 20 min)
- ARP is “plug-and-play”:
  - nodes create their ARP tables *without intervention from net administrator*
- Ethernet (unreliable, connectionless) frame structure
  - *preamble:* 7 bytes , used to synchronize receiver, sender clock rates
  - *addresses:* 6 byte source, destination MAC addresses
    - if adapter receives frame with matching destination address, or with broadcast address (e.g. ARP packet), it passes data in frame to network layer protocol
    - otherwise, adapter discards frame

- *type:* indicates higher layer protocol (mostly IP but others possible, e.g., Novell IPX, AppleTalk)
- *CRC:* cyclic redundancy check at receiver
  - error detected: frame is dropped

- Ethernet's MAC protocol: unslotted *CSMA/CD with binary backoff*
- In Ethernet , data in dropped frames recovered only if initial sender uses higher layer rdt (e.g., TCP), otherwise dropped data lost
- Ethernet Switch :
  - link-layer device: takes an *active* role
    - store, forward Ethernet frames
    - examine incoming frame's MAC address, selectively forward frame to one-or-more outgoing links when frame is to be forwarded on segment, uses CSMA/CD to access segment
  - *transparent*
    - hosts are unaware of presence of switches
  - *plug-and-play, self-learning*
    - switches do not need to be configured

- Ethernet protocol used on *each* incoming link, but no collisions; full duplex
  - each link is its own collision domain
- *switching* can transmit simultaneously, without collisions
- Flood -> frame destination location unknown
- destination A location known -> selectively send on just one link
- *routers:* compute tables using routing algorithms, IP addresses
- *switches:* learn forwarding table using flooding, learning, MAC addresses
- Application , Network , Transport Layers & flow control , switch use buffer
- forwarding between VLANS: done via routing (just as with separate switches)
- *trunk port:* carries frames between VLANS defined over multiple physical switches
- 802.1q protocol adds/removed additional header fields for frames forwarded between trunk ports
- 802.1Q VLAN frame format
  - Tag Control Information (12 bit VLAN ID field, 3 bit priority field like IP TOS)
  - 2-byte Tag Protocol Identifier

MAC addr	interface	TTL

Switch table

(initially empty)

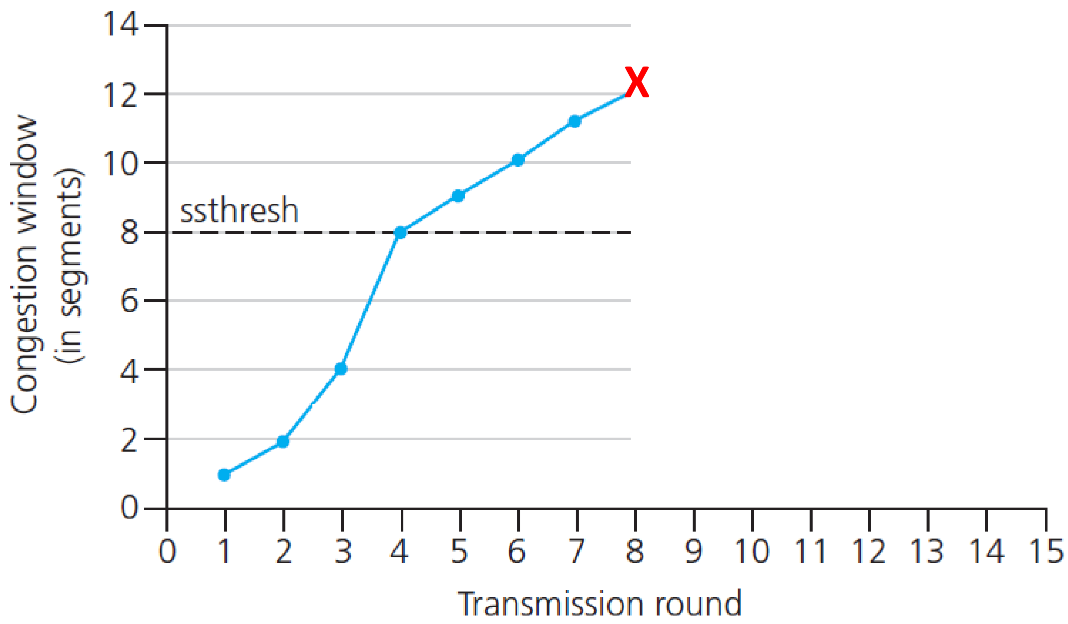
- switch is faster than Router
- forwarding is faster than Routing
- Multiple wireless senders and receivers create additional problems
    - *Hidden terminal problem*
    - *Signal attenuation*
- symmetric key crypto: It only requires a single key for both encryption and decryption.
- **Asymmetric Key Encryption:** It requires two keys, a public key and a private key, one to encrypt and the other one to decrypt

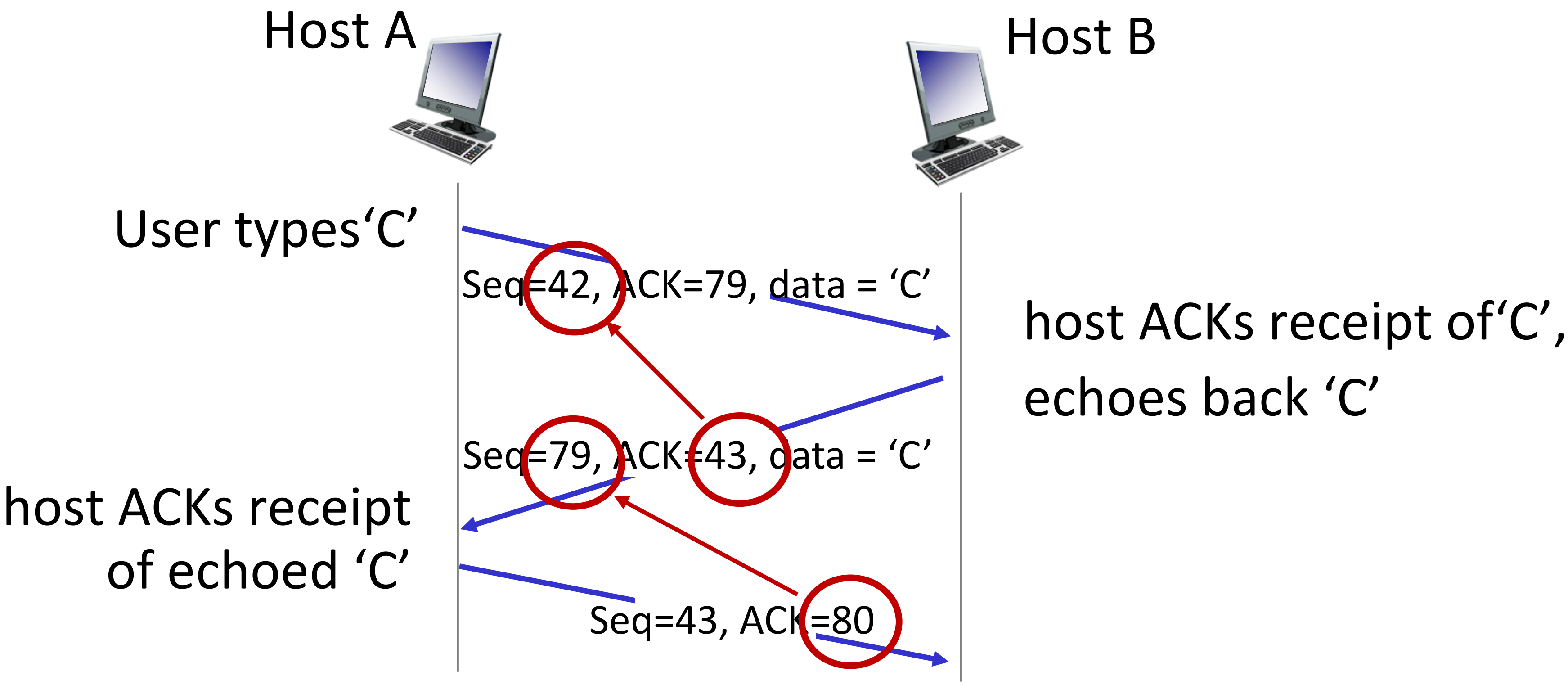
Q: when should the exponential increase switch to linear?

A: when **cwnd** gets to 1/2 of its value before timeout.

Implementation:

- variable **ssthresh**
- on loss event, **ssthresh** is set to 1/2 of **cwnd** just before loss event





simple telnet scenario