Midterm Review

CS 168, Fall 2014
Sylvia Ratnasamy

http://inst.eecs.berkeley.edu/~cs168/
Logistics

- Test is in this classroom starting at 4:10pm
- Closed book, closed notes, etc.
- Single two-sided “cheat sheet”, handwritten
- No calculators, electronic devices, etc.
  - Test does not require any complicated calculation
- I will have extra office hours
  - Friday, Oct 17 1-2pm in 413 Soda Hall
  - Monday, Oct 20 10-11am in 413 Soda Hall
General Guidelines (1)

- Test only assumes material covered in lecture & sections
  - Text: only to clarify details and context for the above

- The test doesn’t require you to do complicated calculations
  - Use this as a hint to whether you are on the right track

- You don’t need to memorize packet headers
  - We’ll provide the IP header for your reference on the exam sheet

- You do need to understand how things work
  - not for the sake of knowing gory details but to understand pros/cons, when a solution is applicable/useful/useless, etc.
General Guidelines (2)

- Be prepared to:
  - Weigh design options outside of the context we studied them in
    *e.g., I had a TCP connection, then BGP went nuts…*
  - Contemplate new designs we haven’t talked about
    *e.g., I introduce a new IP address format; how does this affect..”
    *e.g., I start with UDP, but want weak reliability of the form...”
  - Don’t let this daunt you. Reason from what you know about the
    pros/cons of solutions we did study
    - *e.g., TCP is inefficient when…*
General Guidelines (3)

- Exam format *(tentative!)*
  - Q1) 20 multiple-choice questions
    - *ordered (roughly) from easiest to hardest*
  - Q2) Design questions: A set of “here’s a scenario, tell me if the following is true/false”-style questions
    - *ordered (roughly) from easiest to hardest within each scenario*
  - Q3+ more traditional questions
    - *(we think) 3 < 4 < 5 < ... (< implies easier than)*
    - *sub-questions within each question ordered easiest to hardest*

- Pace yourself accordingly!
This Review

- Walk through what we expect you to know: key topics, important aspects of each

- Just because I didn’t cover it in review doesn’t mean you don’t need to know it
  - But if I covered it today, you should know it

- My plan: summarize, not explain
  - Stop me when you want to discuss something further!
Topics

- Basic concepts (lectures 2, 3)
- Architecture and principles (lecture 4)
- Network layer (lecs. 4-9)
  - Concepts: valid routing state, convergence, least-cost paths
  - Overall context (inter- and intra-domain routing)
  - Computing least-cost routes (DV, LS)
  - IP addressing
  - Inter-domain
  - Router architecture
- Transport (lecs. 9 -12)
  - Role of the transport layer
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only
Basic concepts

- You should know:
  - statistical multiplexing
  - packet vs. circuit switching
  - link characteristics
  - packet delays
How are network resources shared?

Two approaches

- Reservations
- On demand
Intuition: reservations

Link capacity = 30Mbps

Each source gets 10Mbps

Frequent overloading

Time
Intuition: on demand

No overloading

Link capacity = 30Mbps
Two approaches to sharing

- Reservations $\rightarrow$ circuit switching
- On demand $\rightarrow$ packet switching
Two approaches to sharing

- **Packet switching**
  - network resources consumed on demand per-packet
  - “admission control”: per packet

- **Circuit switching**
  - network resources reserved a priori at “connection” initiation
  - "admission control": per connection
Packet switching exploits *statistical multiplexing* better than circuit switching

- Sharing using the statistics of demand
- Good for bursty traffic (average $<<$ peak demand)
- Similar to insurance, with the same failure mode
Circuit Switching

(1) src sends a reservation request to dst
(2) Switches “establish a circuit”
(3) src starts sending data
(4) src sends a “teardown circuit” message
Packet Switching

- Data is sent as chunks of formatted bits (Packets)
- Packets consist of a “header” and “payload”
- Switches “forward” packets based on their headers
- Each packet travels independently
- No link resources are reserved in advance
Circuit Switching

‣ Pros
  - predictable performance
  - simple/fast switching (once circuit established)

‣ Cons
  - inefficient when traffic is bursty
  - complexity of circuit setup/teardown
  - circuit setup adds delay
  - switch fails $\rightarrow$ its circuit(s) fails
Packet Switching

- Pros
  - efficient use of network resources
  - simpler to implement
  - robust: can “route around trouble”

- Cons
  - requires buffer management and congestion control
  - unpredictable performance
Performance Metrics

- Delay
- Loss
- Throughput
A network link

- **Link bandwidth**
  - number of bits sent/received per unit time (bits/sec or bps)

- **Propagation delay**
  - time for one bit to move through the link (seconds)

- **Bandwidth-Delay Product (BDP)**
  - number of bits “in flight” at any time
  - \( BDP = \text{bandwidth} \times \text{propagation delay} \)
Delay

- Consists of four components
  - transmission delay
  - propagation delay
  - queuing delay
  - processing delay

  - due to link properties
  - due to traffic mix and switch internals
End-to-end delay
Packet Delay

Sending 100B packets from A to B?

- Time to transmit 800 bits = 800 x 1/10^6 s
- Time to transmit 100 Byte packet = 1/10^6 s

Time when that bit reaches B = 1/10^6 s + 1/10^3 s

The last bit reaches B at (800 x 1/10^6) + 1/10^3 s = 1.8 ms
Little’s Law (1961)

$L = A \times W$

- $A$: Average rate at which packets arrive at a queue
- $W$: Average time packets wait at the queue
- $L$: Avg. number of packets waiting in queue ($q$ length)

- Easy to compute $L$, harder to compute $W$
Topics

- Basic concepts (lectures 2, 3)
- Architecture and principles (lecture 4)
- Network layer (lecs. 4-9)
  - Concepts: valid routing state, convergence, least-cost paths
  - Overall context (inter- and intra-domain routing)
  - Computing least-cost routes (DV, LS)
  - IP addressing
  - Inter-domain
  - Router architecture
- Transport (lecs. 9-12)
  - Role of the transport layer
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only
You should know

- Layering: what/where/why
- Protocols: what/where/why
- Principles: layering, end-to-end argument, “narrow waist”
- Benefits and weaknesses/consequences of principles/choices
  - *E.g., layering is good because... but has hurt...*
Layering

- Layering is a particular form of modularization
- System is broken into a vertical hierarchy of logically distinct entities (layers)
- Service provided by one layer is based solely on the service provided by layer below
Internet Layers

Applications
...built on...
Reliable (or unreliable) transport
...built on...
Best-effort global packet delivery
...built on...
Best-effort local packet delivery
...built on...
Physical transfer of bits

Layers:
- L1: Physical
- L2: Data link
- L3: Network
- L4: Transport
- L7: Application
What gets implemented where?

- Lower three layers implemented everywhere
- Top two layers implemented only at hosts
Logical Communication

- Layers interacts with peer’s corresponding layer
Physical Communication

- Communication goes down to physical network
- Then up to relevant layer
Protocols and Layers

Communication between peer layers on different systems is defined by protocols.
There is just one network-layer protocol!
Layer Encapsulation

App
Transport
Network
Link
Physical

Alice
Router
Bob
Layers: pros and cons

Why layer?
- Reduce complexity
- Improve flexibility/innovation
  * (Each layer can evolve independently)

Why not layer?
- sub-optimal performance
- cross-layer information often useful
Some application requirements can only be correctly implemented **end-to-end**
- reliability, security, *etc.*

**End-systems**
- **Can** satisfy the requirement without network’s help
- **Will/must** do so, since they can’t rely on the network
Implications of the E2E argument

- In layered design, the E2E principle provides guidance on which layers are implemented where.

- Key argument for why IP offers only “best effort” delivery (leading to “dumb network / smart ends”):
  - Reliability implemented at the end host (TCP)
  - Often credited as key to the Internet’s success
Architectural Wisdom

● Layering
  ● reduce complexity, increase flexibility

● IP as the “narrow waist”
  ● eases interoperability

● “smart ends, dumb network” (E2E argument)
  ● No application knowledge in network → more general
  ● Minimal state in the network → more robust to failure
Topics

- Basic concepts (lectures 2,3)
- Architecture and principles (lecture 4)

Network layer (lecs. 4-9)
- Concepts: valid routing state, convergence, least-cost paths
- Overall context (inter- and intra-domain routing)
- Routing algorithms that compute least-cost routes (DV, LS)
- IP addressing
- Inter-domain
- Router architecture

Transport (lecs. 9 -12)
- Role of the transport layer
- UDP vs. TCP
- TCP details: reliability and flow control
- TCP congestion control: general concepts only
Forwarding vs. Routing

- **Forwarding:** “data plane”
  - *Directing one data packet*
  - *Each router using local forwarding table*

- **Routing:** “control plane”
  - *Computing the forwarding tables that guide packets*
  - *Jointly computed by routers using a distributed algorithm*
Routing: basic concepts

- Valid routing state
- Convergence
- Least-cost paths
“Valid” Routing State

- Global forwarding state is “valid” if it produces forwarding decisions that always deliver packets to their destinations

- Global routing state is valid \textit{if and only if}:
  - There are no dead ends (other than destination)
  - There are no loops
Convergence Delay

• Time to achieve convergence
  – E.g., all nodes have the same link-state database

• Sources of convergence delay?
  – time to detect failure
  – time to flood link-state information
  – time to re-compute forwarding tables

• Performance during convergence period?
  – lost packets due to blackholes
  – looping packets
  – out-of-order packets reaching the destination
Least-cost path routing

- Given: router graph & link costs
- Goal: find least-cost path from each source router to each destination router
- Distance-Vector and Link-State are examples
“Least Cost” Routes

• “Least cost” routes an easy way to avoid loops
  – No sensible cost metric is minimized by traversing a loop

• Least cost routes are destination-based
  – i.e., do not depend on the source

• Least-cost paths form a spanning tree
Topics

• Basic concepts (lectures 2,3)

• Architecture and principles (lecture 4)

• Network layer (lecs. 4-9)
  – Concepts: valid routing state, convergence, least-cost paths
  – Overall context (inter- and intra-domain routing)
  – Routing algorithms that compute least-cost routes (DV, LS)
  – IP addressing
  – Inter-domain
  – Router architecture

• Transport (lecs. 9 -12)
  – Role of the transport layer
  – UDP vs. TCP
  – TCP details: reliability and flow control
  – TCP congestion control: general concepts only
“Autonomous System (AS)” or “Domain”
Region of a network under a single administrative entity

“Border Routers”

An “end-to-end” route

“Interior Routers”
Internet Routing

- Internet Routing works at two levels

- Each AS runs an *intra-domain* routing protocol that establishes routes within its domain
  - Intra-domain routes are “least cost”
  - e.g., Link State (OSPF) and Distance Vector (RIP)

- ASes participate in an *inter-domain* routing protocol that establishes routes between domains
  - Inter-domain routes determined by policy (need not be least-cost)
  - e.g., Path Vector (BGP)
Link State Routing

• Every router knows its local “link state”
• A router floods its link state to all other routers
• Every router learns the entire network graph
• Every router locally runs Dijkstra’s to compute its forwarding table

Will not test your ability to solve Dijkstra’s under pressure

• But you should know the high level properties of LS
  – every node maintains complete topology
  – link updates flooded everywhere
  – may have loops while nodes have inconsistent topology information
Distance-vector routing

• Distributed algorithm (Bellman-Ford)
• All routers run it “together”
  - each router runs its own instance
  - neighbors exchange and react to each other’s messages
Distance Vector Routing

- Each router knows the links to its neighbors.

- Each router has provisional “least cost” estimate to every other router — its distance vector (DV)
  - *E.g.: Router A: “A can get to B with cost 11”*

- Routers exchange this DV with their neighbors.

- Routers look over the set of options offered by their neighbors and select the best one.

- Iterative process converges to set of shortest paths.
\[ d_x(z) = \min_n \{ \text{cost}(x,n) + d_n(z) \} \]

for all neighbors \( n \)

Bellman-Ford equation
DV: You **should understand**

- How DV works
  - what’s in a DV; how nodes process and update DVs
- The counting to infinity problem
  - why it occurs
- Poison Reverse
  - when it does/doesn’t fix counting-to-infinity
Counting-to-Infinity

- **Cause**
  - z routes through y, y routes through x (to reach $dst$)
  - y loses connectivity to x
  - y decides to route through z (to reach $dst$)
- **Can take a very long time to resolve**
Poisoned Reverse

- **How:**
  - If z routes to $dst$ through y, z advertises to y that its cost to $dst$ is infinite
  - y never decides to route to $dst$ through z
- Often avoids the count-to-infinity problem
Topics

- Basic concepts (lectures 2,3)
- Architecture and principles (lecture 4)
- **Network layer (lecs. 4-9)**
  - Concepts: valid routing state, convergence, least-cost paths
  - Overall context (inter- and intra-domain routing)
  - Routing algorithms that compute least-cost routes (DV, LS)
  - IP addressing
  - Inter-domain
  - Router architecture

- **Transport (lecs. 9 -12)**
  - Role of the transport layer
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only
Addressing Goal: **Scalable** Routing

- **State**: Small forwarding tables at routers
- **Churn**: Limited rate of change in routing tables

Ability to aggregate addresses is crucial for both (one entry to *summarize* many addresses)
Hierarchy in IP Addressing

- 32 bits are partitioned into a prefix and suffix components
- Prefix is the network component; suffix is host component

Interdomain routing operates on the network prefix

“slash” notation: 12.34.158.0/23 → network with a 23 bit prefix and $2^9$ host addresses
IP addressing → scalable routing?

- Hierarchical address allocation helps routing scalability if allocation matches topological hierarchy.

- Problem: may not be able to aggregate addresses for “multi-homed” networks.
- UCB is “multi-homed” to AT&T and ESNet
  - Multi-homed domain $\rightarrow$ domain has 2 (or more) providers

- ESNet must maintain routing entries for both a.*.*.* and a.c.*.*.
IP addressing → scalable routing?

- Hierarchical address allocation helps routing scalability if allocation matches topological hierarchy.

- Problem: may not be able to aggregate addresses for “multi-homed” networks.

- Two competing forces in scalable routing:
  - Aggregation reduces number of routing entries.
  - Multi-homing increases number of entries.
BGP and Inter-Domain Routing

- Destinations are IP prefixes (12.0.0.0/8)
- Nodes are Autonomous Systems (ASes)
- Links represent both physical connections and business relationships
  - customer-provider or peer-to-peer
- BGP is the protocol for inter-domain routing
Topology and policy is shaped by the business relationships between ASes

- Three basic kinds of relationships between ASes
  - AS A can be AS B’s customer
  - AS A can be AS B’s provider
  - AS A can be AS B’s peer

- Business implications
  - Customer pays provider
  - Peers don’t pay each other
BGP extends DV

- With some important differences
  - routes selected based on policy, not just shortest path
  - path vector (useful to avoid loops)
  - Selective route advertisement
  - may aggregate routes (aggregating prefixes)
Policy imposed in how routes are selected and exported

- **Selection**: Which path to use?
  - controls whether/how traffic leaves the network
- **Export**: Which path to advertise?
  - controls whether/how traffic enters the network

Can reach 128.3/16 blah blah
## Typical Export Policy

<table>
<thead>
<tr>
<th>Destination prefix advertised by...</th>
<th>Export route to...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer</td>
<td>Everyone (providers, peers, other customers)</td>
</tr>
<tr>
<td>Peer</td>
<td>Customers</td>
</tr>
<tr>
<td>Provider</td>
<td>Customers</td>
</tr>
</tbody>
</table>

We’ll refer to these as the “Gao-Rexford” rules (capture common but not required! practice!) You must know this!
Typical Selection Policy

- In decreasing order of priority
  - make/save money (send to customer > peer > provider)
  - maximize performance (smallest AS path length)
  - minimize use of my network bandwidth ("hot potato")
  - …

- BGP uses route attributes to implement the above
  - AS PATH, LOCAL_PREF, MED, …

You should know the general idea/goal for each attribute; we won’t quiz you on the detailed implementation
Policy Dictates Route Selection

A

B

C

D

E

F

Pr → Cu
Peer
Peer

traffic allowed

traffic not allowed
Topics

- Basic concepts (lectures 2,3)
- Architecture and principles (lecture 4)
- Network layer (lecs. 4-9)
  - Concepts: valid routing state, convergence, least-cost paths
  - Overall context (inter- and intra-domain routing)
  - Routing algorithms that compute least-cost routes (DV, LS)
  - IP addressing
  - Inter-domain
  - Router architecture
- Transport (lecs. 9-12)
  - Role of the transport layer
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only
We’ll give you the header format but you should know what each field is and its use/misuse

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4-bit Version</td>
<td>Version number</td>
</tr>
<tr>
<td>4-bit Header Length</td>
<td>Header length</td>
</tr>
<tr>
<td>8-bit Type of Service</td>
<td>Type of service</td>
</tr>
<tr>
<td>16-bit Total Length (Bytes)</td>
<td>Total length in bytes</td>
</tr>
<tr>
<td>16-bit Identification</td>
<td>Identification</td>
</tr>
<tr>
<td>3-bit Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>13-bit Fragment Offset</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>8-bit Time to Live (TTL)</td>
<td>Time to live</td>
</tr>
<tr>
<td>8-bit Protocol</td>
<td>Protocol number</td>
</tr>
<tr>
<td>16-bit Header Checksum</td>
<td>Header checksum</td>
</tr>
<tr>
<td>32-bit Source IP Address</td>
<td>Source IP address</td>
</tr>
<tr>
<td>32-bit Destination IP Address</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>Options (if any)</td>
<td>Options</td>
</tr>
<tr>
<td>Payload</td>
<td>Payload</td>
</tr>
</tbody>
</table>
# IPv4 and IPv6 Header Comparison

<table>
<thead>
<tr>
<th>Field name kept from IPv4 to IPv6</th>
<th>Fields not kept in IPv6</th>
<th>Name &amp; position changed in IPv6</th>
<th>New field in IPv6</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPv4</td>
<td>IPv6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Version</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IHL</td>
<td>Traffic Class</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type of Service</td>
<td>Flow Label</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Length</td>
<td>Payload Length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td>Next Header</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td>Hop Limit</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fragment Offset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time to Live</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Address</td>
<td>Source Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Address</td>
<td>Destination Address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td>Padding</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Padding</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What’s inside a router?

Linecards (input)

- Processes packets on their way in
- 1
- 2
- \(\cdots\)
- \(N\)

Interconnect (Switching) Fabric

Route/Control Processor

- Processes packets before they leave
- Transfers packets from input to output ports

Linecards (output)

Input and Output for the same port are on one physical linecard
What’s inside a router?

(1) Implement IGP and BGP protocols; compute routing tables

(2) Push forwarding tables to the line cards
What's inside a router?

Linecards (input) → Linecards (output)

1. Interconnect Fabric
2. Route/Control Processor

- Constitutes the data plane
- Constitutes the control plane
Challenges in Router Design

- @ Line cards: destination lookups at high speed
  - e.g., find the longest prefix match (LPM) in the table that matches the packet destination address

- @ Switch fabric: head-of-line blocking, scheduling the switch fabric at high speed

- @ Route processor: complexity/correctness more a problem than performance

You should understand why these challenges arise but we don’t expect you to know how to fix them
- e.g., specifics of scheduling algorithms or LPM lookups
Topics

- Basic concepts (lectures 2,3)
- Architecture and principles (lecture 4)
- Network layer (lecs. 4-9)
  - Concepts: valid routing state, convergence, least-cost paths
  - Overall context (inter- and intra-domain routing)
  - Routing algorithms that compute least-cost routes (DV, LS)
  - IP addressing
  - Inter-domain
  - Router architecture
- Transport (lecs. 9 -12)
  - Role of the transport layer
  - UDP vs. TCP
  - TCP details: reliability and flow control
  - TCP congestion control: general concepts only
Role of the Transport Layer

(1) Communication between application processes
   - Mux and demux from/to application processes
   - Implemented using *ports*

(2) Provide common end-to-end services for app layer
   - Reliable, in-order data delivery
   - Well-paced data delivery
Both UDP and TCP provide mux/demux-ing via ports

<table>
<thead>
<tr>
<th>Data abstraction</th>
<th>UDP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packets (datagrams)</td>
<td>Stream of bytes of arbitrary length</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Service</th>
<th>UDP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best-effort (same as IP)</td>
<td>• Reliability&lt;br&gt;• In-order delivery&lt;br&gt;• Congestion control&lt;br&gt;• Flow control</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Applications</th>
<th>UDP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Video, audio streaming</td>
<td>File transfer, chat</td>
<td></td>
</tr>
</tbody>
</table>
Reliable Transport: General Concepts

- Checksums (for error detection)
- Timers (for loss detection)
- Acknowledgments (feedback from receiver)
  - cumulative: “received everything up to X”
  - selective: “received X”
- Sequence numbers (detect duplicates, accounting)
- Sliding Windows (for efficiency)

You should know:
- what these concepts are
- why they exist
- how TCP uses them
Things to know about TCP

- How TCP achieves reliability
- RTT estimation
- Connection establishment/teardown
- Flow Control
- Congestion Control (concepts only)

- For each, know how the functionality is implemented and why it is needed
E.g., RTT Estimation

- Why? TCP uses timeouts to retransmit packets
- But RTT may vary (significantly!) for different reasons and on different timescales
  - due to temporary congestion
  - due to long-lived congestion
  - due to a change in routing paths
- An incorrect RTT estimate might introduce spurious retransmissions or overly long delays
- RTT estimators should react to change but not too quickly
  - proposed solutions use EWMA, incorporate deviations
E.g., Reliability

- Why? IP is best-effort but many apps. need reliable delivery
  - Having TCP take care of it simplifies application development
- How
  - checksums and timers (for error and loss detection)
  - fast retransmit (for faster-than-timeout loss detection)
  - cumulative ACKs (feedback from receiver -- what’s lost/what’s not)
  - sliding windows (for efficiency)
  - buffers at sender (to hold packets while waiting for ACKs)
  - buffers at receiver (to reorder packets before delivery to app.)
E.g., Connection Establishment

● Why?
  ● TCP is a stateful protocol (CWND, buffer space, ISN, etc.)
  ● Need to initialize connection state at both ends
  ● Exchange initial sequence numbers

● How? Three-way handshake
  ● Host A sends a **SYN** to host B
  ● Host B returns a SYN acknowledgment (**SYN ACK**)
  ● Host A sends an **ACK** (+ data) to acknowledge the SYN ACK
  ● Hosts exchange proposed Initial Sequence Numbers at each step
E.g., Flow Control

- Why?
  - TCP offers a reliable in-order byte stream abstraction
  - Hence, TCP at the receiver must buffer a packet until all packets before it (in byte-order) have arrived and the receiving application has consumed available bytes
  - Hence receiver advances its window when the receiving application consumes data
  - But sender advances its window when new data ACK’d
  - Hence, risk the sender might overrun the receiver’s buffers

- How? “Advertised Window” field in TCP header
  - Receiver advertises the “right hand edge” of its window to sender
  - Sender agrees not to exceed this amount
E.g., Congestion Control

- Why?
  - Because a sender shouldn’t overload the network itself
  - But yet, should make efficient use of available network capacity
  - While sharing available capacity fairly with other flows
  - And adapting to changes in available capacity

- How?
  - Dynamically adapts the size of the sending window
    (don’t worry about the exact algorithms used to do the adaptation)
Final Questions?

- Good luck!