CS 54001-1: Large-Scale Networked Systems

Professor: Ian Foster

TAs: Xuehai Zhang, Yong Zhao

Winter Quarter

www.classes.cs.uchicago.edu/classes/archive/2003/winter/54001-1

CS 54001-1 Course Goals

- Yes
  - Gain understanding of fundamental issues that effect design, construction, and operation of large-scale networked systems
  - Gain understanding of some significant future trends in network design and use
- No
  - Learn how to write network applications

Remember

- I ask you to:
  - Read Peterson and Davies Ch 1 and 2
  - Read "End to End Arguments in System Design"
  - Use traceroute to determine paths to following locations & build map of network
    - ANL, IIT, NWU, UIC, Loyola, UIUC, Purdue, Indiana

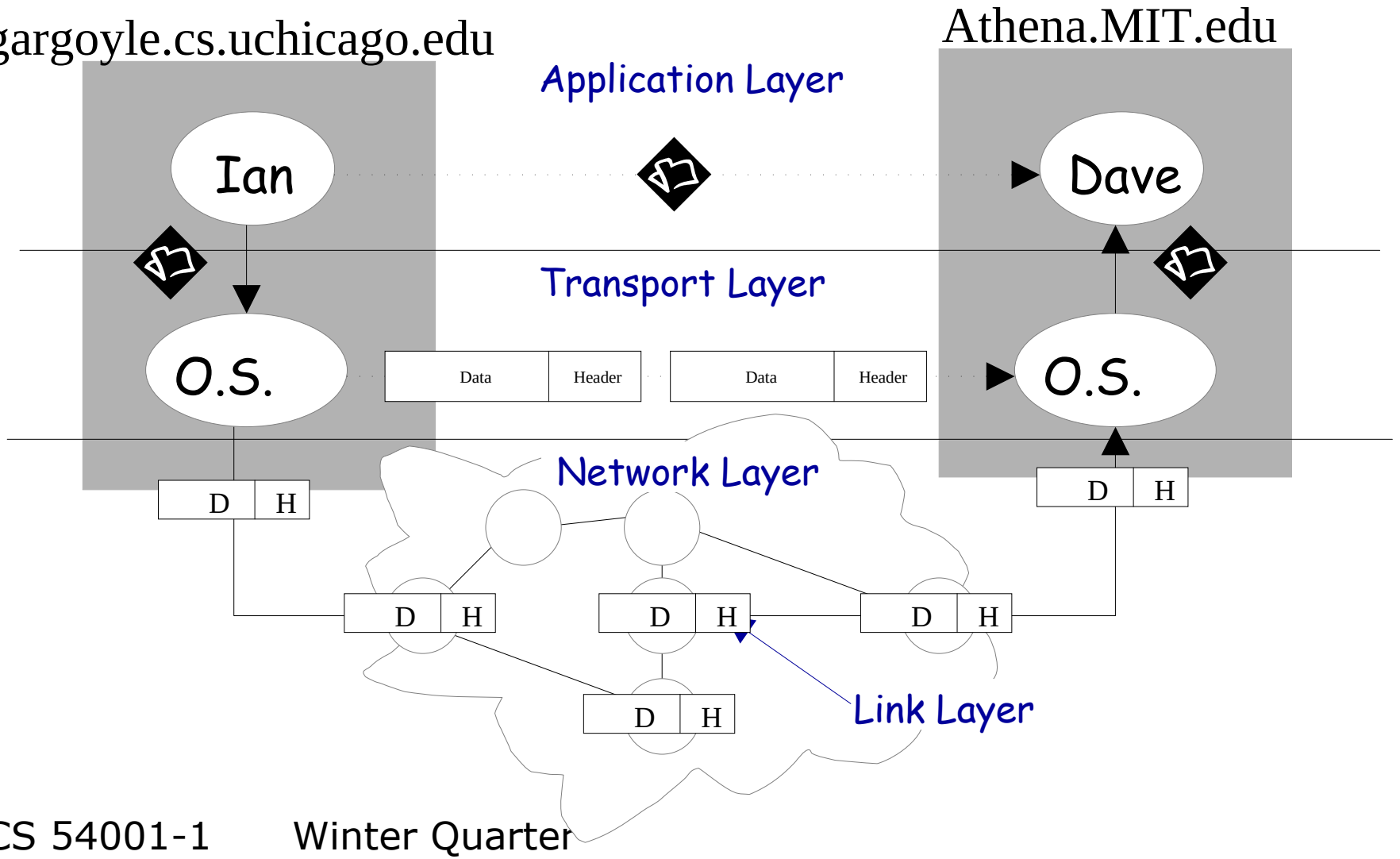
Last Week: Internet Design Principles & Protocols

- An introduction to the mail system
- An introduction to the Internet
- Internet design principles and layering
- Brief history of the Internet
- Packet switching and circuit switching
- Protocols
- Addressing and routing
- Performance metrics
- A detailed FTP example

This Week: Routing and Transport

- Routing techniques
  - Flooding
  - Distributed Bellman Ford Algorithm
  - Dijkstra's Shortest Path First Algorithm
- Routing in the Internet
  - Hierarchy and Autonomous Systems
  - Interior Routing Protocols: RIP, OSPF
  - Exterior Routing Protocol: BGP
- Transport: achieving reliability
- Transport: achieving fair sharing of links

Recap:
An Introduction to the Internet



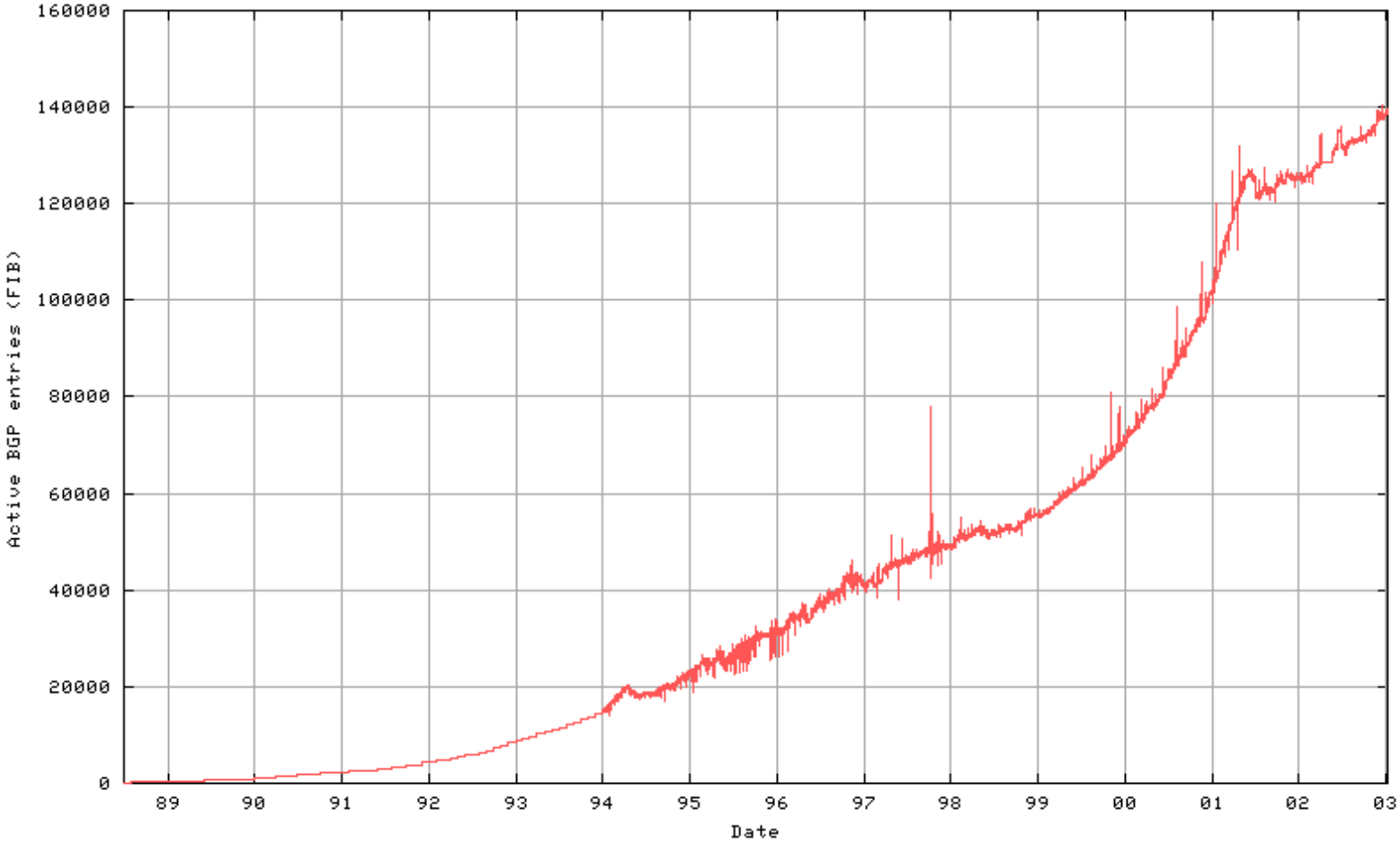
Characteristics of the Internet

- Each packet is individually routed
- No time guarantee for delivery
- No guarantee of delivery in sequence
- No guarantee of delivery at all!
  - Things get lost
  - Acknowledgements
  - Retransmission
    - How to determine when to retransmit? Timeout?
    - Need local copies of contents of each packet.
    - How long to keep each copy?
    - What if an acknowledgement is lost?

Characteristics of the Internet (2)

- No guarantee of integrity of data.
- Packets can be fragmented.
- Packets may be duplicated.

Size of the Routing Table at the core of the Internet

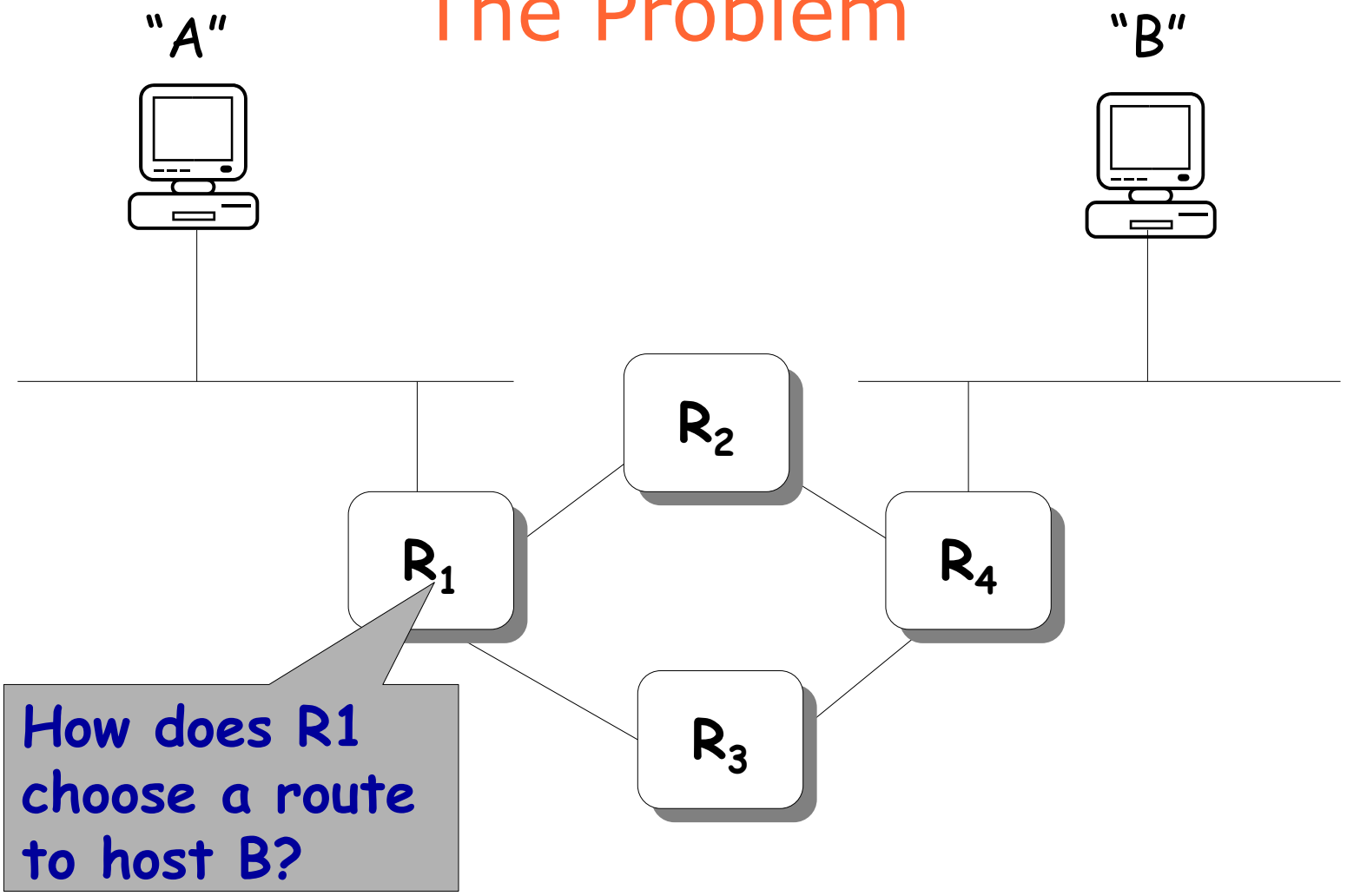


Source: http://www.telstra.net/ops/bgptable.html

This Week: Routing and Transport

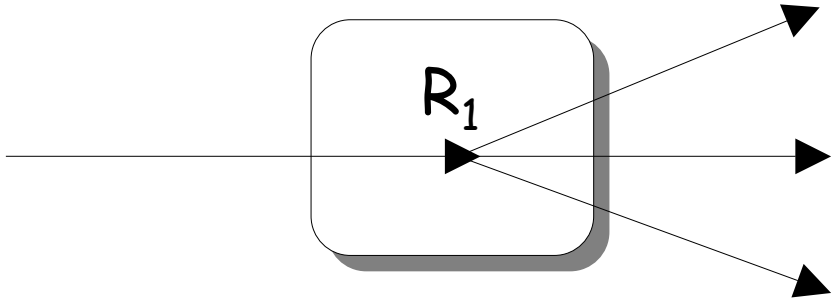
- **Routing techniques**
  - **Flooding**
  - **Distributed Bellman Ford Algorithm**
  - **Dijkstra's Shortest Path First Algorithm**
- Routing in the Internet
  - Hierarchy and Autonomous Systems
  - Interior Routing Protocols: RIP, OSPF
  - Exterior Routing Protocol: BGP
- Transport: achieving reliability
- Transport: achieving fair sharing of links

The Problem



Technique 1: Flooding

Routers forward packets to all ports except the ingress port.



Advantages:

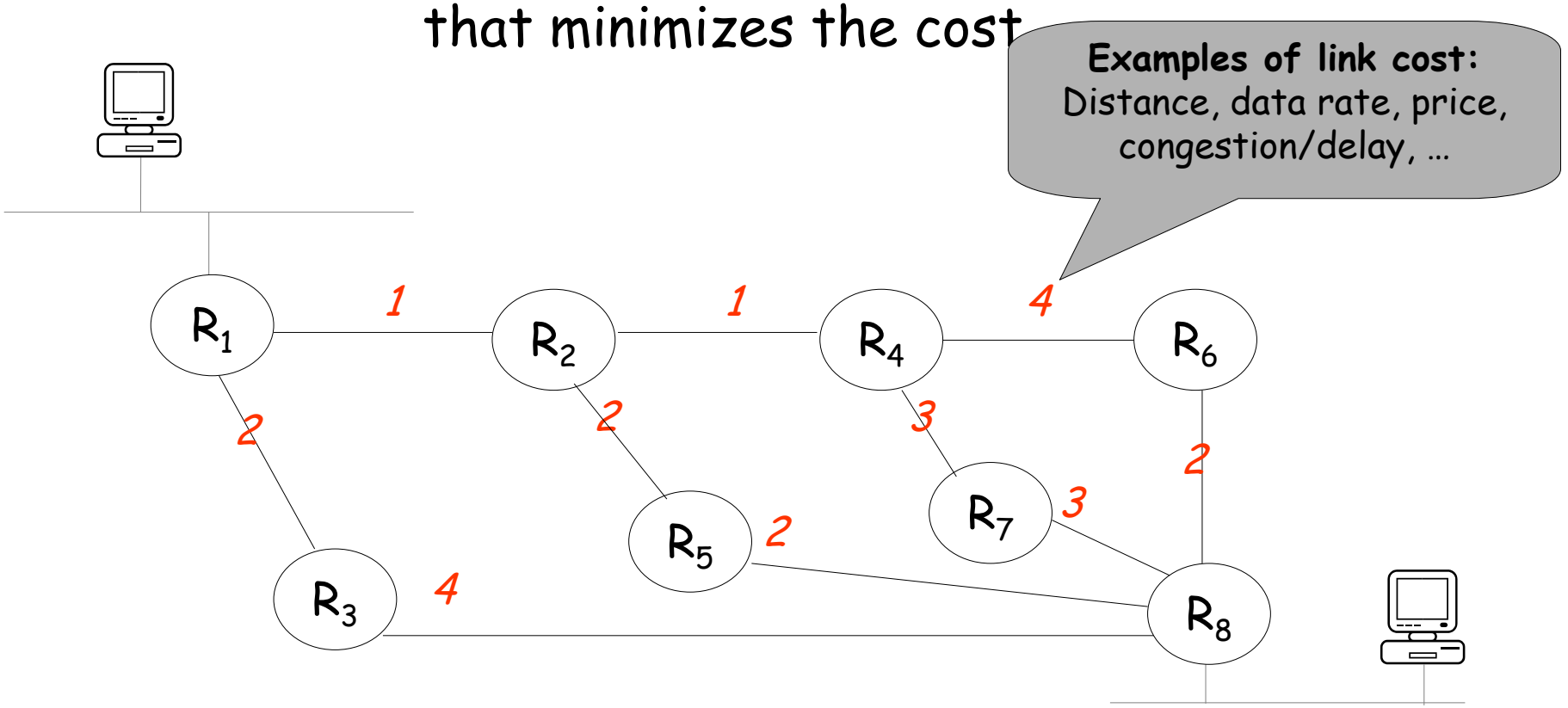
- Every destination in the network is reachable.
- Useful when network topology is unknown.

Disadvantages:

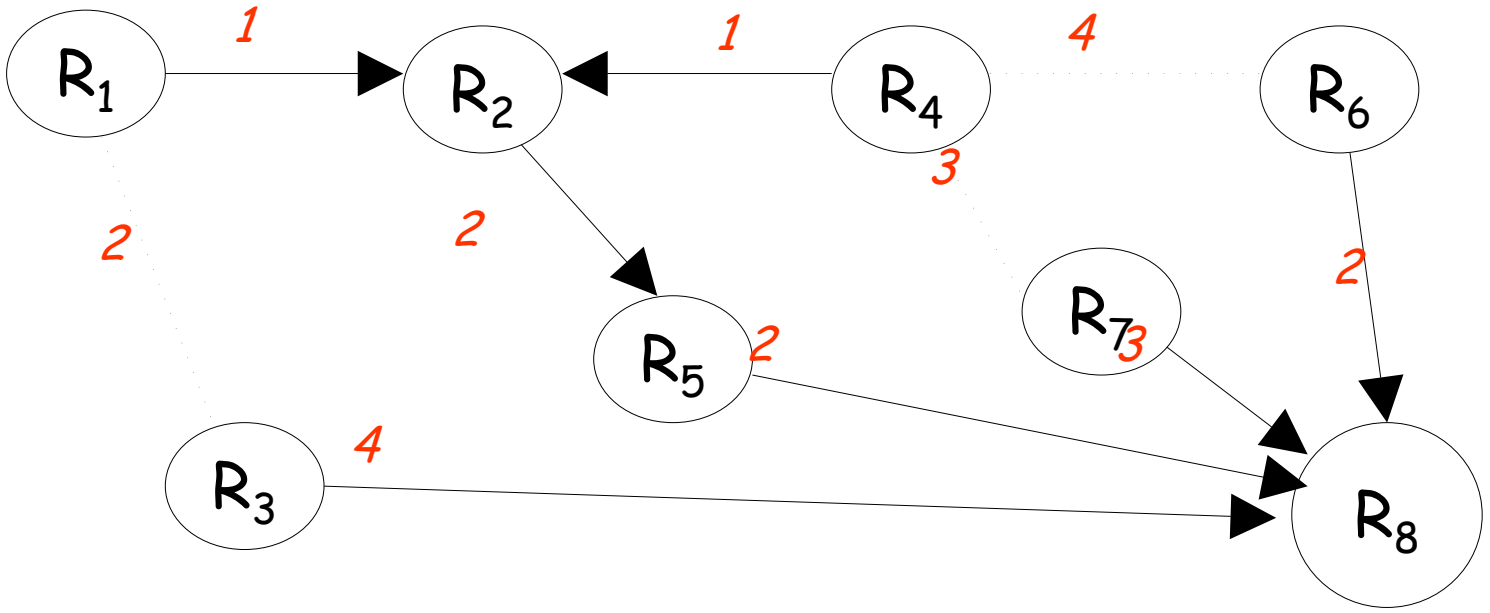
- Some routers receive packet multiple times.
- Packets can go round in loops forever.

Technique 2: Bellman-Ford Algorithm

Objective: Determine the route from (R_1, …, R_7) to R_8 that minimizes the cost



Solution is simple by inspection... (in this case)

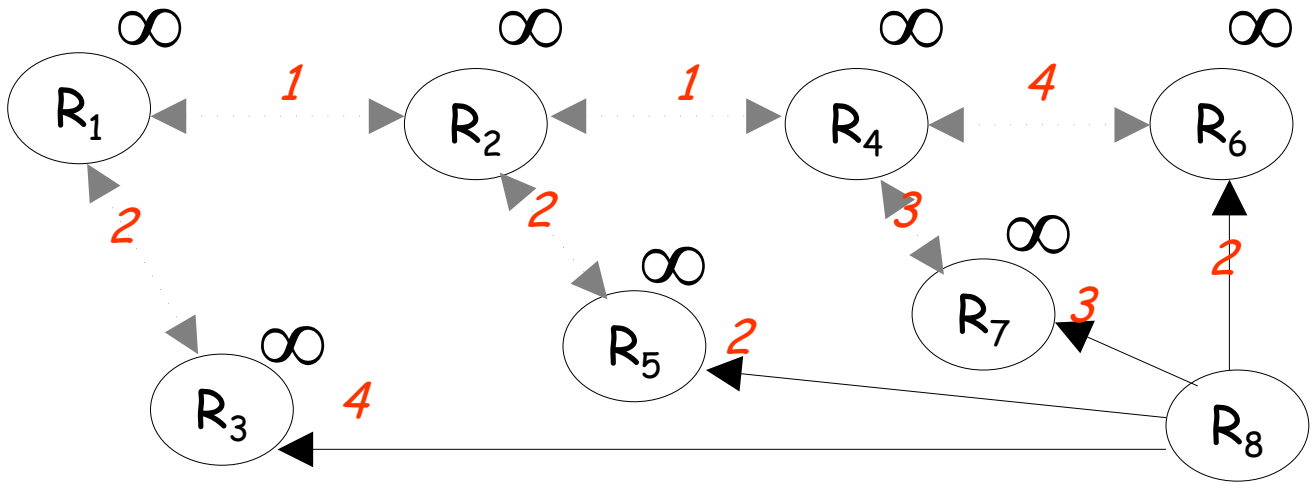


- The solution is a spanning tree with R_8 as the root of the tree.
- The Bellman-Ford Algorithm finds the spanning tree automatically.

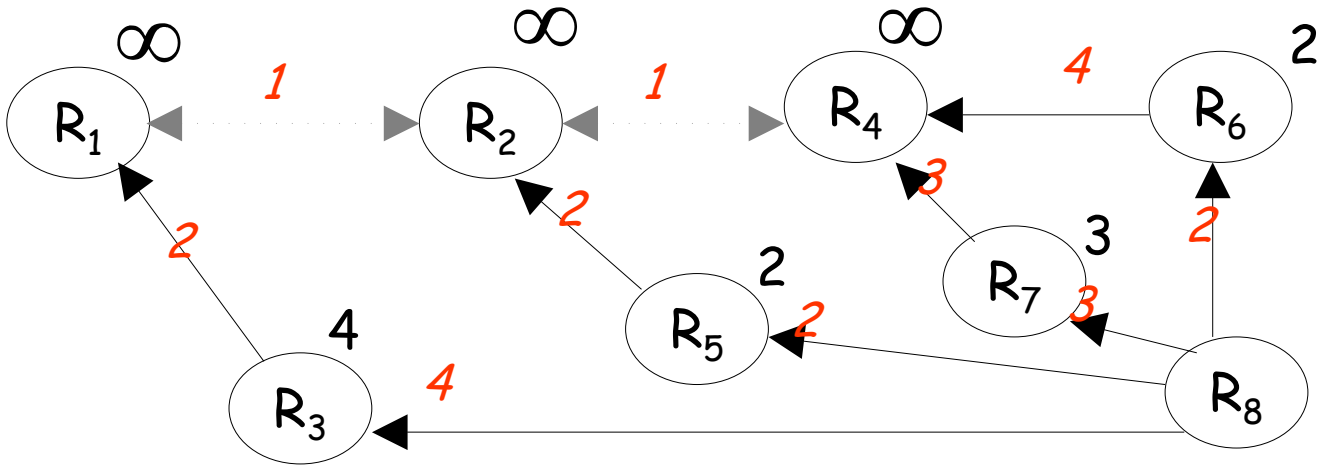
The Distributed Bellman-Ford Algorithm

1. Let $\underline{X}_n = (C_1, C_2, ..., C_7)$ where: C_i = cost from R_i to R_8.
2. Set $\underline{X}_0 = (\infty, \infty, \infty, ..., \infty)$.
3. Every T seconds, router i sends C_i to its neighbors.
4. If router i is told of a lower cost path to R_8, it updates C_i. Hence, $\underline{X}_{n+1} = f(\underline{X}_n)$ where $f(.)$ determines the next step improvement.
5. If $\underline{X}_{n+1} \neq \underline{X}_n$ then goto step (3).
6. Stop.

Bellman-Ford Algorithm: Example

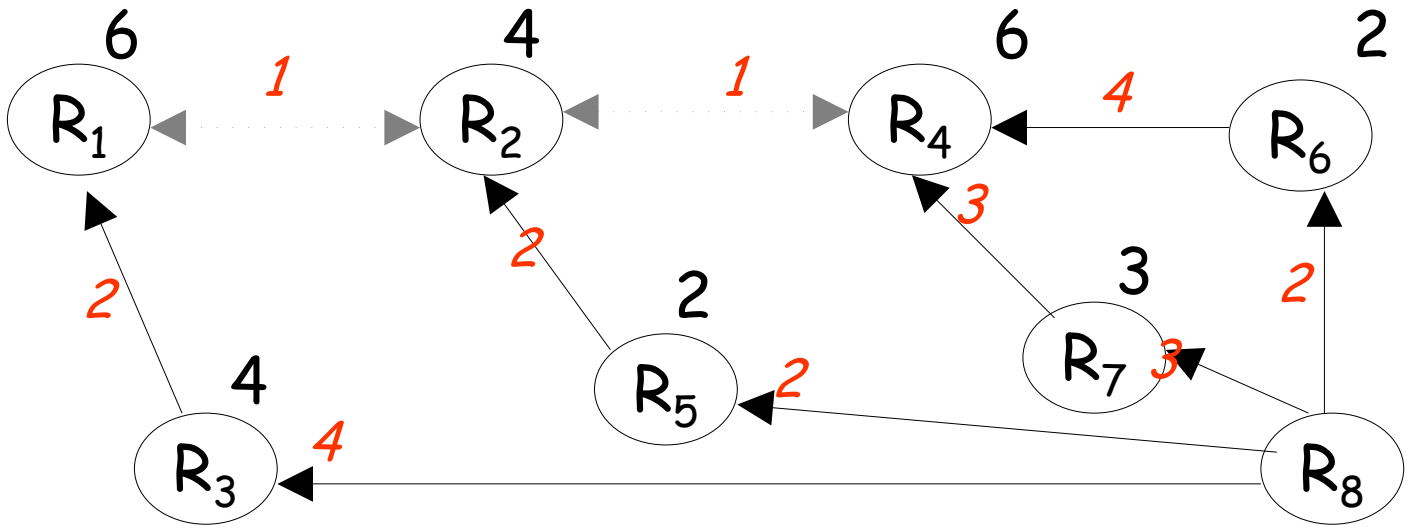


R_1	Inf
R_2	Inf
R_3	4, R_8
R_4	Inf
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



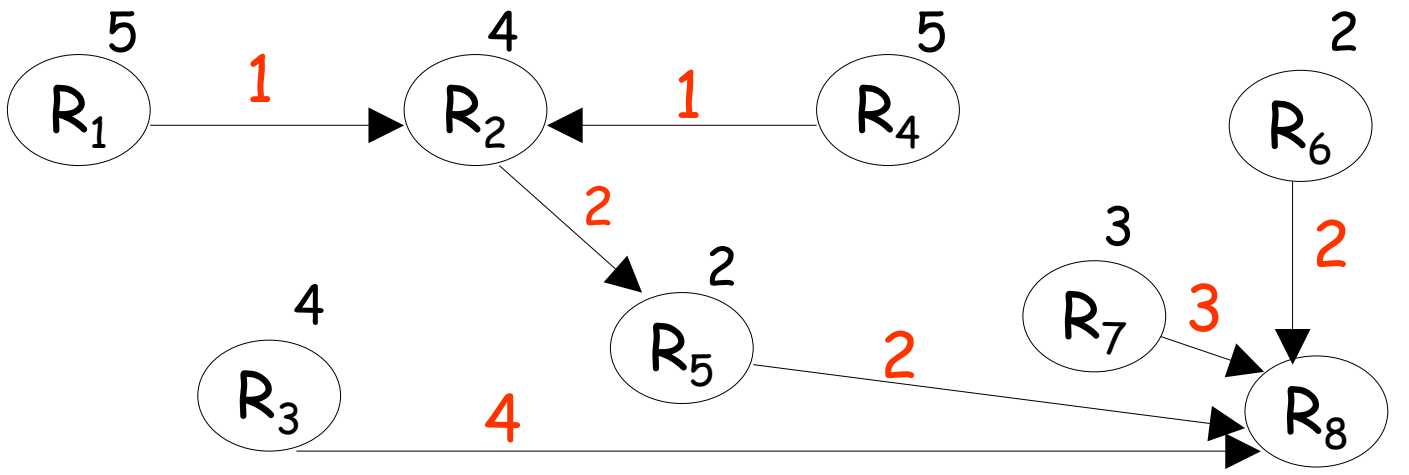
Bellman-Ford Algorithm

R_1	6, R_3
R_2	4, R_5
R_3	4, R_8
R_4	6, R_7
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



Solution

R_1	5, R_2
R_2	4, R_5
R_3	4, R_8
R_4	5, R_2
R_5	2, R_8
R_6	2, R_8
R_7	3, R_8



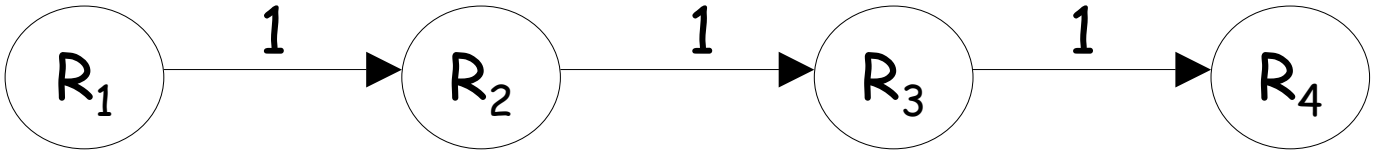
Bellman-Ford Algorithm

Questions:

1. How long can the algorithm take to run?
2. How do we know that the algorithm always converges?
3. What happens when link costs change, or when routers/links fail?

A Problem with Bellman-Ford

"Bad news travels slowly"



Consider the calculation of distances to R_4:

Time	R_1	R_2	R_3
0	$3,R_2$	$2,R_3$	$1, R_4$
1	$3,R_2$	$2,R_3$	$3,R_2$
2	$3,R_2$	$4,R_3$	$3,R_2$
3	$5,R_2$	$4,R_3$	$5,R_2$
...	"Counting to infinity"		...

$R_3 \rightarrow R_4$ fails

Counting to Infinity Problem Solutions

1. Set infinity = "some small integer" (e.g., 16) Stop when count = 16
2. Split Horizon: Because R2 received lowest cost path from R3, it does not advertise cost to R3
3. Split-horizon with poison reverse: R2 advertises infinity to R3

Technique 3: Dijkstra's Shortest Path First Algorithm

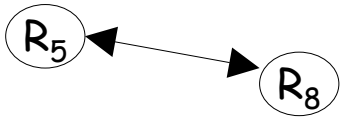
- Routers send out update messages whenever the state of a link changes. Hence the name: "Link State" algorithm
- Each router calculates lowest cost path to all others, starting from itself
- At each step of the algorithm, router adds the next shortest (i.e., lowest-cost) path to the tree
- Finds spanning tree routed on source router

Dijkstra's Shortest Path First Algorithm: Example

Step 1: Shortest path set, $S = \{R_8\}$. Candidate set, $C = \{R_3, R_5, R_7, R_6\}$

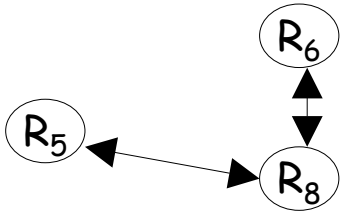
Step 2: $S = \{R_8, R_5\}$,

$C = \{R_3, R_7, R_6, R_2\}$.



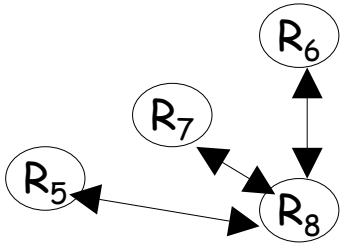
Step 3: $S = \{R_8, R_5, R_6\}$,

$C = \{R_3, R_7, R_2, R_4\}$.



Step 4: $S = \{R_8, R_5, R_6, R_7\}$,

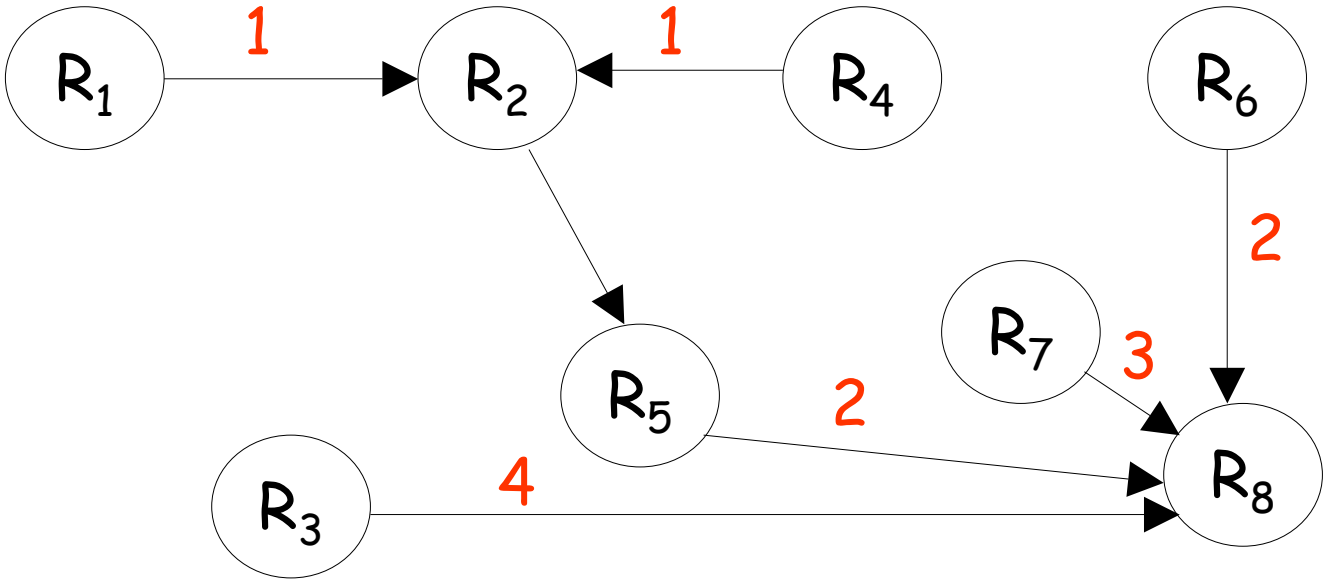
$C = \{R_3, R_2, R_4\}$.



Dijkstra's SPF Algorithm

Step 8 : $S = \{R_8, R_5, R_6, R_7, R_2, R_1, R_4\}$,

$C = \{\}$.



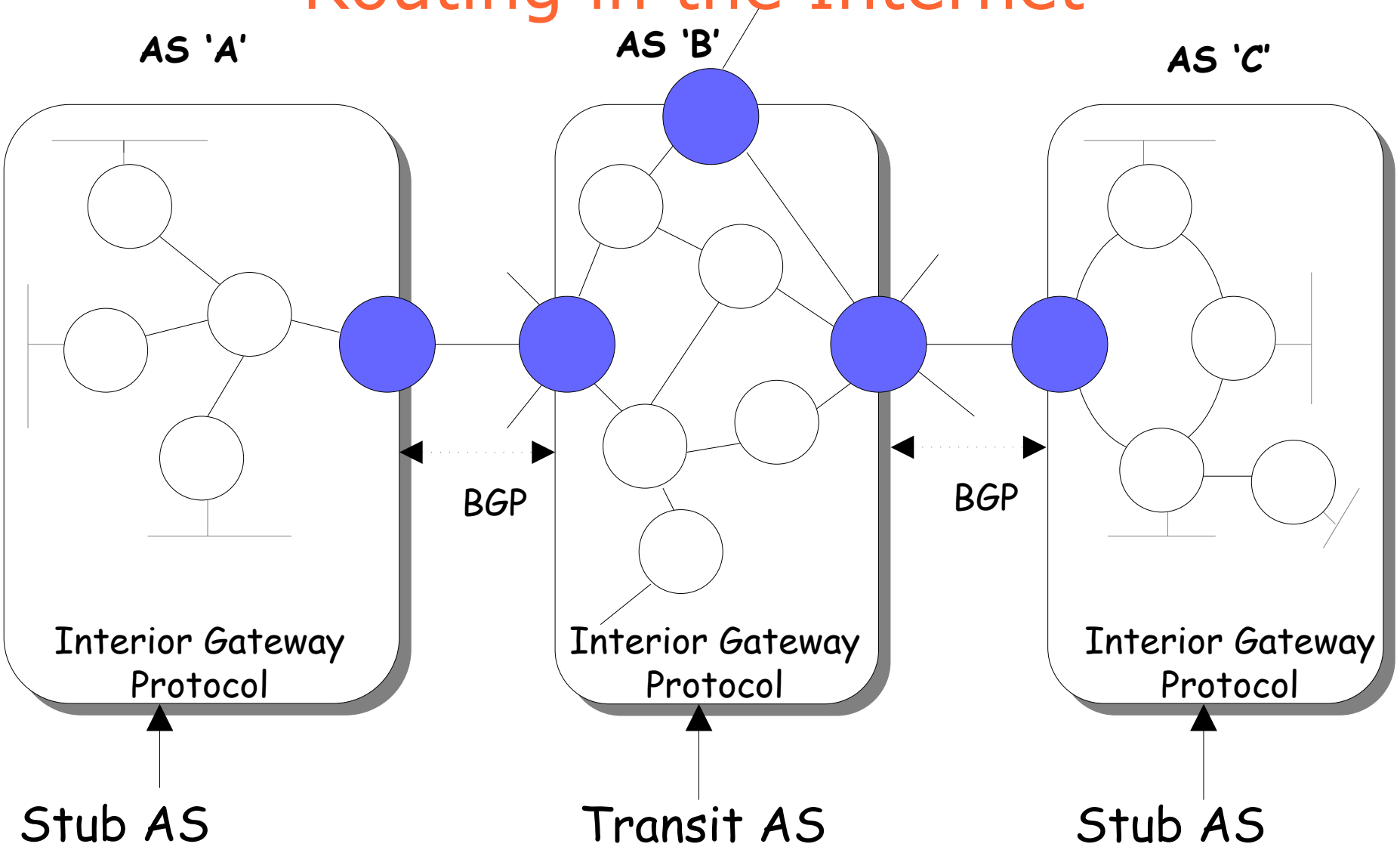
This Week: Routing and Transport

- Routing techniques
  - Flooding
  - Distributed Bellman Ford Algorithm
  - Dijkstra's Shortest Path First Algorithm
- **Routing in the Internet**
  - **Hierarchy and Autonomous Systems**
  - **Interior Routing Protocols: RIP, OSPF**
  - **Exterior Routing Protocol: BGP**
- Transport: achieving reliability
- Transport: achieving fair sharing of links

Routing in the Internet

- The Internet uses hierarchical routing
- Internet is split into Autonomous Systems (ASs)
  - Examples of ASs: Stanford (32), HP (71), MCI Worldcom (17373)
  - Try: whois –h whois.arin.net ASN "MCI Worldcom"
- Within an AS, the administrator chooses an Interior Gateway Protocol (IGP)
  - Examples of IGPs: RIP (rfc 1058), OSPF (rfc 1247).
- Between ASs, the Internet uses an Exterior Gateway Protocol
  - ASs today use the Border Gateway Protocol, BGP-4 (rfc 1771)

Routing in the Internet



Routing within a Stub AS

- There is only one exit point, so routers within the AS can use default routing
  - Each router knows all Network IDs within AS
  - Packets destined to another AS are sent to the default router
  - Default router is the border gateway to the next AS
- Routing tables in Stub ASs tend to be small

Interior Routing Protocols

- RIP (Routing Information Protocol)
  - Uses distributed Bellman-Ford algorithm
  - Updates sent every 30 seconds
  - No authentication
  - Originally in BSD UNIX
- OSPF (Open Shortest Path First)
  - Link-state updates sent (using flooding) as and when required
  - Every router runs Dijkstra's algorithm
  - Authenticated updates
  - Autonomous system may be partitioned into "areas"

Exterior Routing Protocols

Problems:

- Topology: The Internet is a complex mesh of different ASs with very little structure
- Autonomy of ASs: Each AS defines link costs in different ways, so not possible to find lowest cost paths
- Trust: Some ASs can't trust others to advertise good routes (e.g., two competing backbone providers), or to protect the privacy of their traffic (e.g., two warring nations)
- Policies: Different ASs have different objectives (e.g., route over fewest hops; use one provider rather than another)

Border Gateway Protocol (BGP-4)

- BGP is not a link-state or distance-vector routing protocol
- BGP advertises complete paths (a list of ASs)
- Example of path advertisement:
  - "The network 171.64/16 can be reached via the path {AS1, AS5, AS13}".
- Paths with loops are detected locally and ignored
- Local policies pick the preferred path among options
- When link/router fails, the path is "withdrawn"

This Week: Routing and Transport

- Routing techniques
  - Flooding
  - Distributed Bellman Ford Algorithm
  - Dijkstra's Shortest Path First Algorithm
- Routing in the Internet
  - Hierarchy and Autonomous Systems
  - Interior Routing Protocols: RIP, OSPF
  - Exterior Routing Protocol: BGP
- **Transport: achieving reliability**
- Transport: achieving fair sharing of links

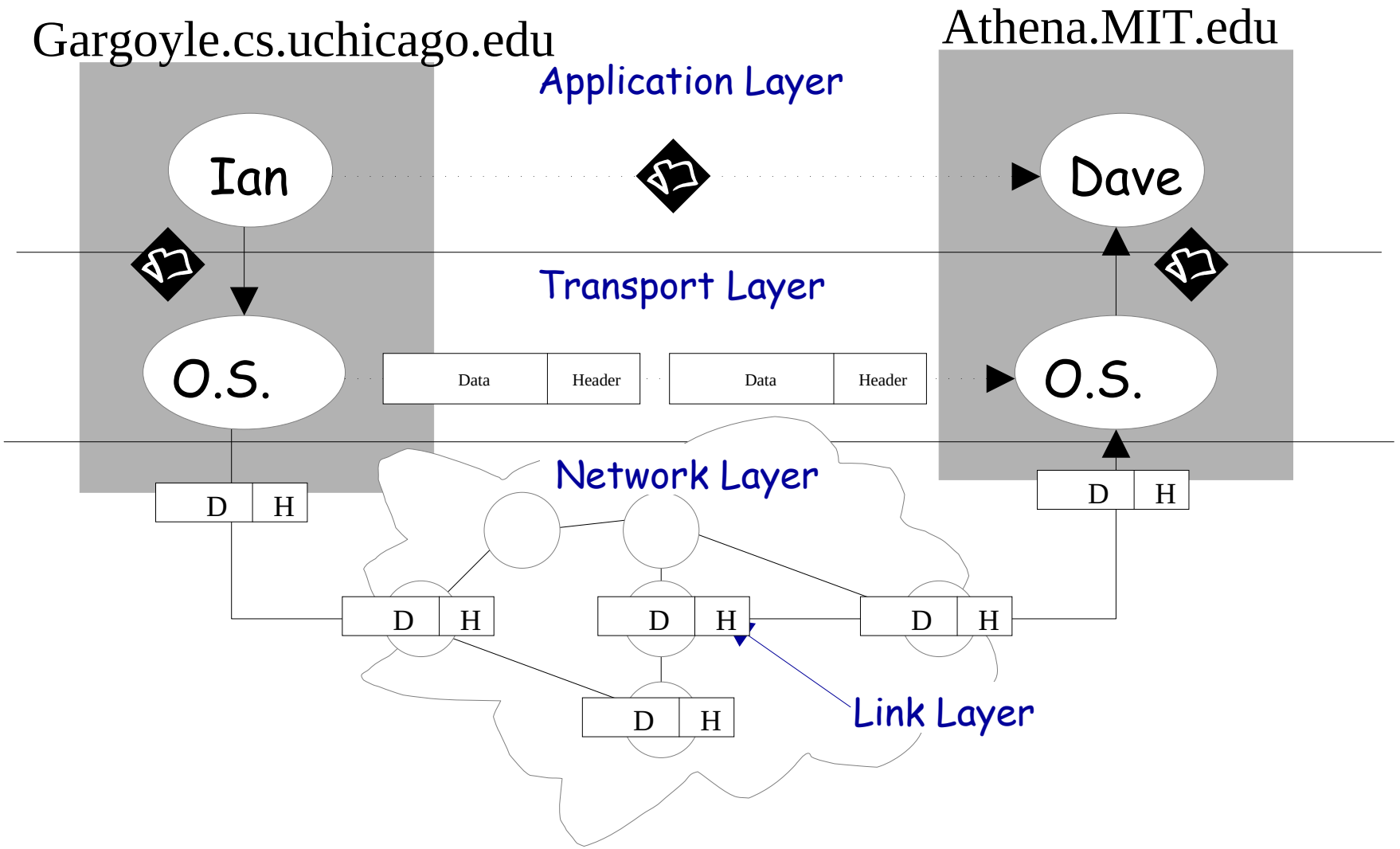
Outline

- The Transport Layer
- The TCP Protocol
  - TCP Characteristics
  - TCP Connection setup
  - TCP Segments
  - TCP Sequence Numbers
  - TCP Sliding Window
  - Timeouts and Retransmission
  - (Congestion Control and Avoidance)
- The UDP Protocol

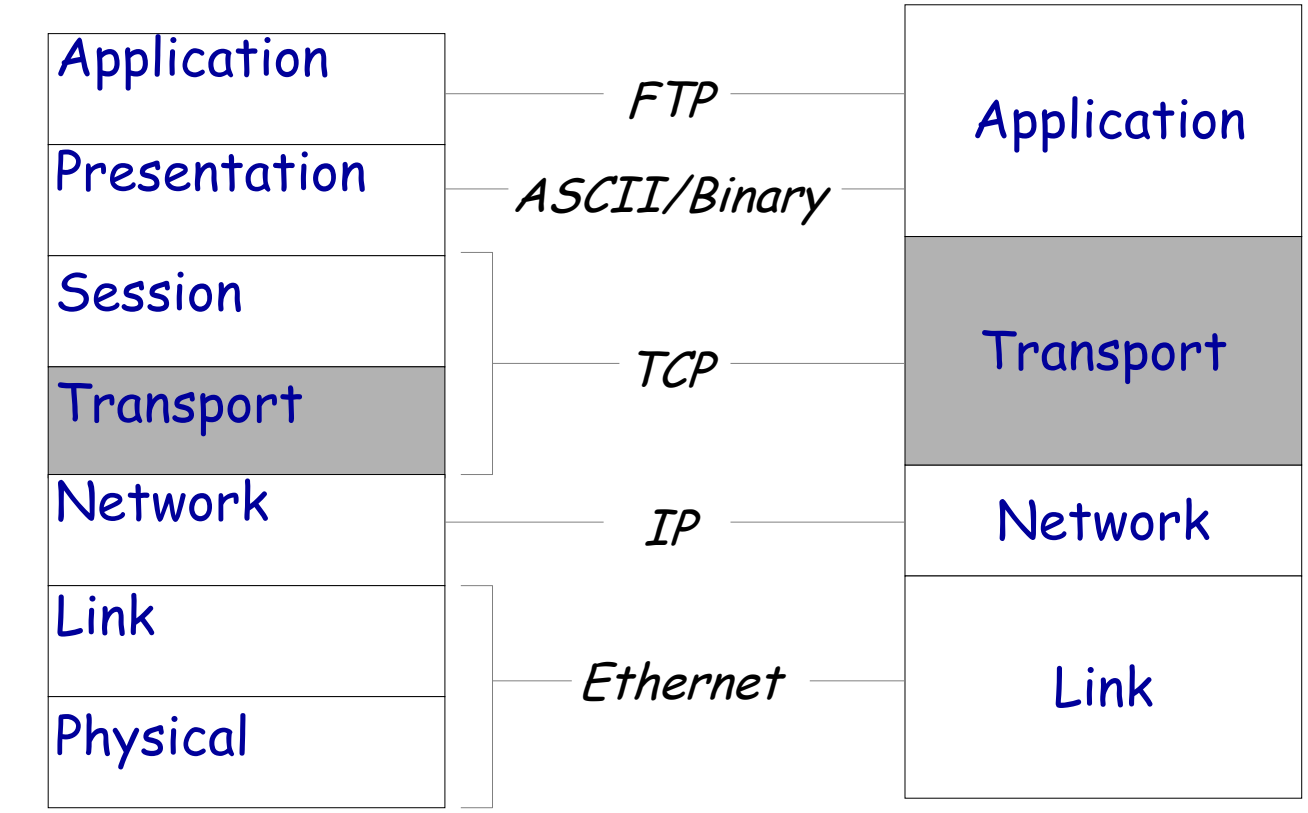
The Transport Layer

- What is the transport layer for?
- What characteristics might it have?
  - Reliable delivery
  - Flow control
  - …

Review of the Transport Layer



Layering: FTP Example



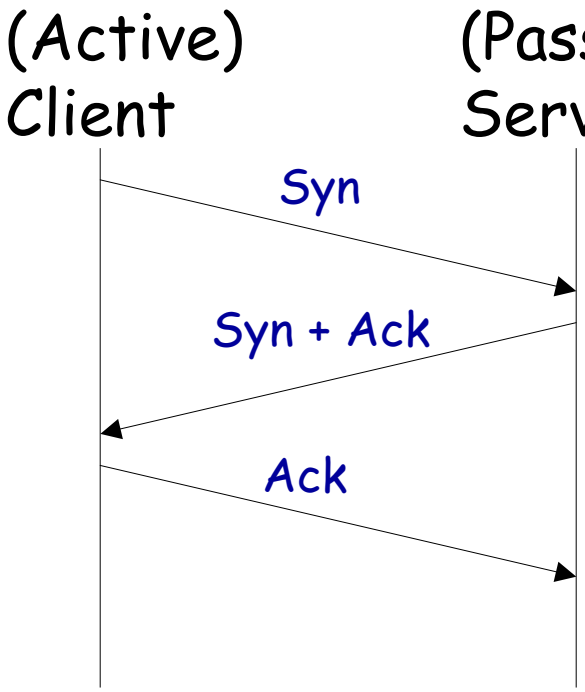
The 7-layer OSI Model

The 4-layer Internet model

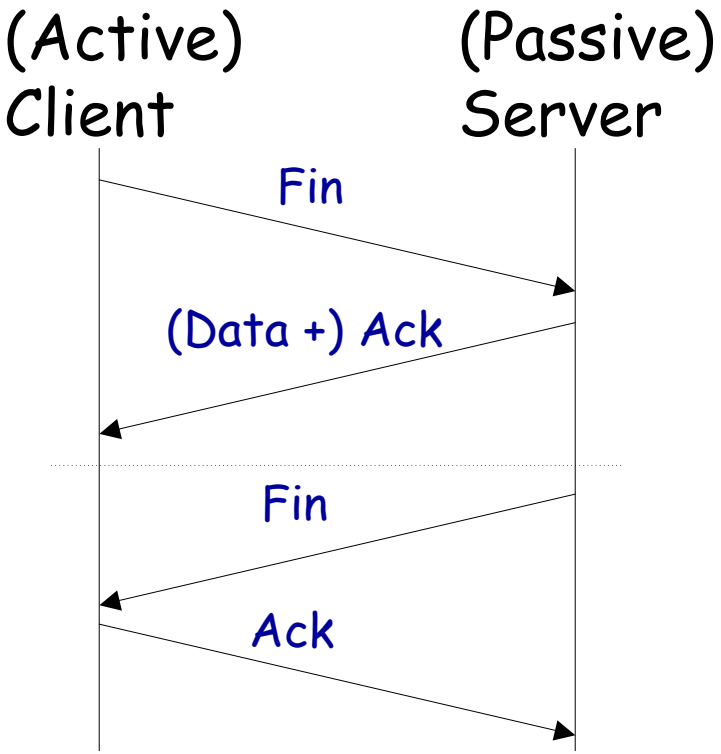
TCP Characteristics

- TCP is connection-oriented
  - 3-way handshake used for connection setup
- TCP provides a stream-of-bytes service
- TCP is reliable:
  - Acknowledgements indicate delivery of data
  - Checksums are used to detect corrupted data
  - Sequence numbers detect missing, or mis-sequenced data
  - Corrupted data is retransmitted after a timeout
  - Mis-sequenced data is re-sequenced
  - (Window-based) Flow control prevents over-run of receiver
- TCP uses congestion control to share network capacity among users

TCP is connection-oriented



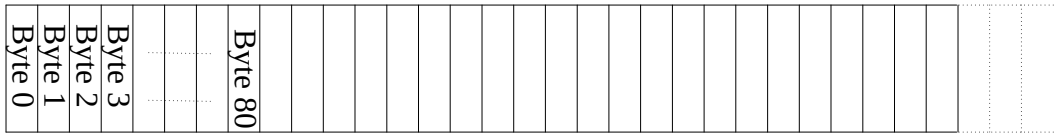
Connection Setup
3-way handshake



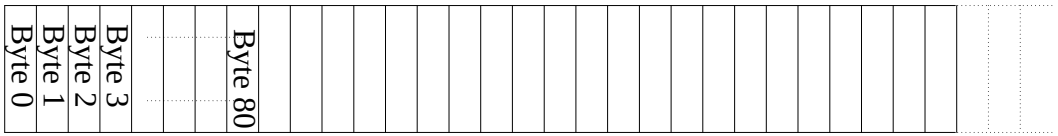
Connection Close/Teardown
2 x 2-way handshake

TCP supports a "stream of bytes" service

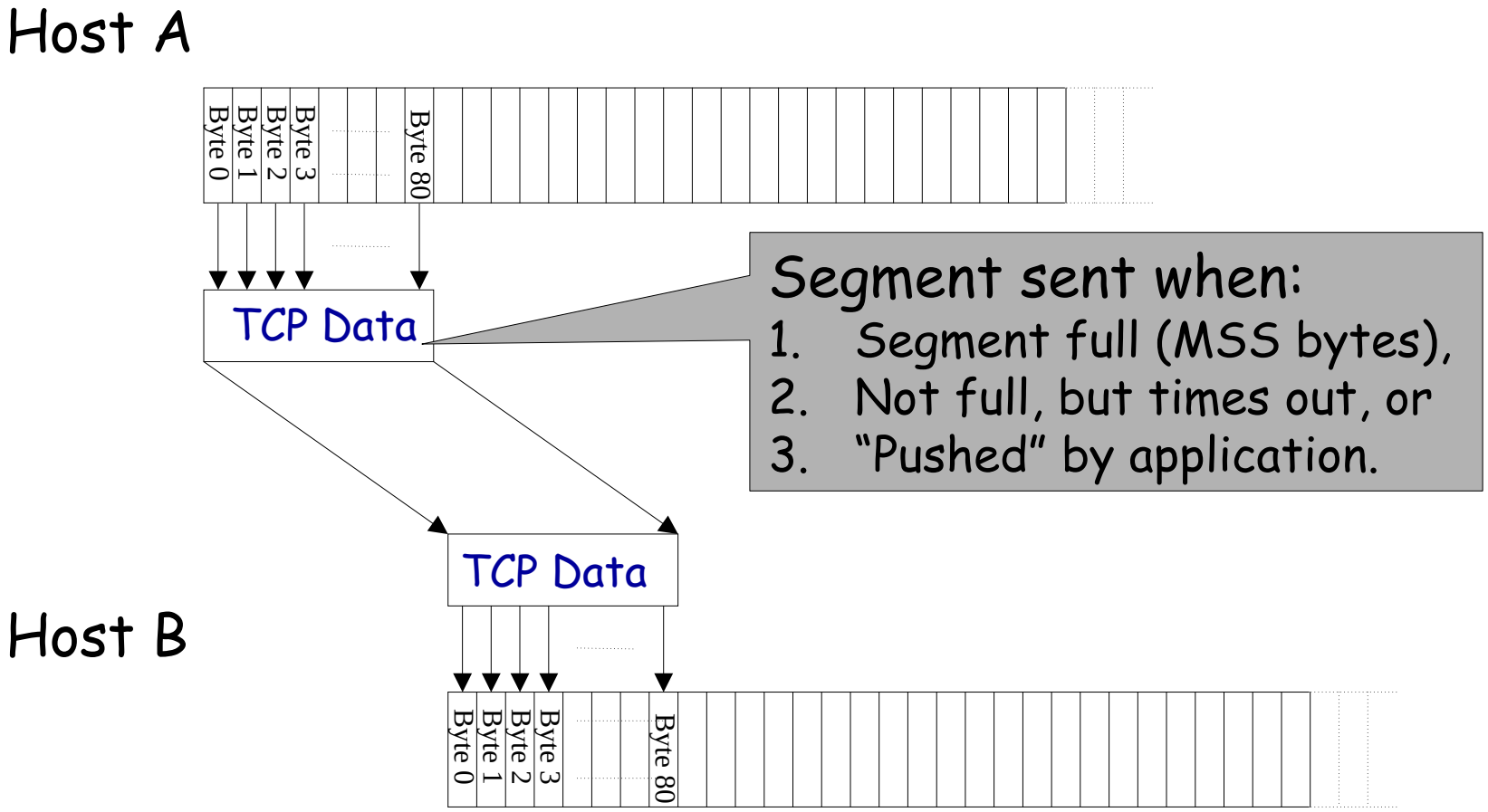
Host A



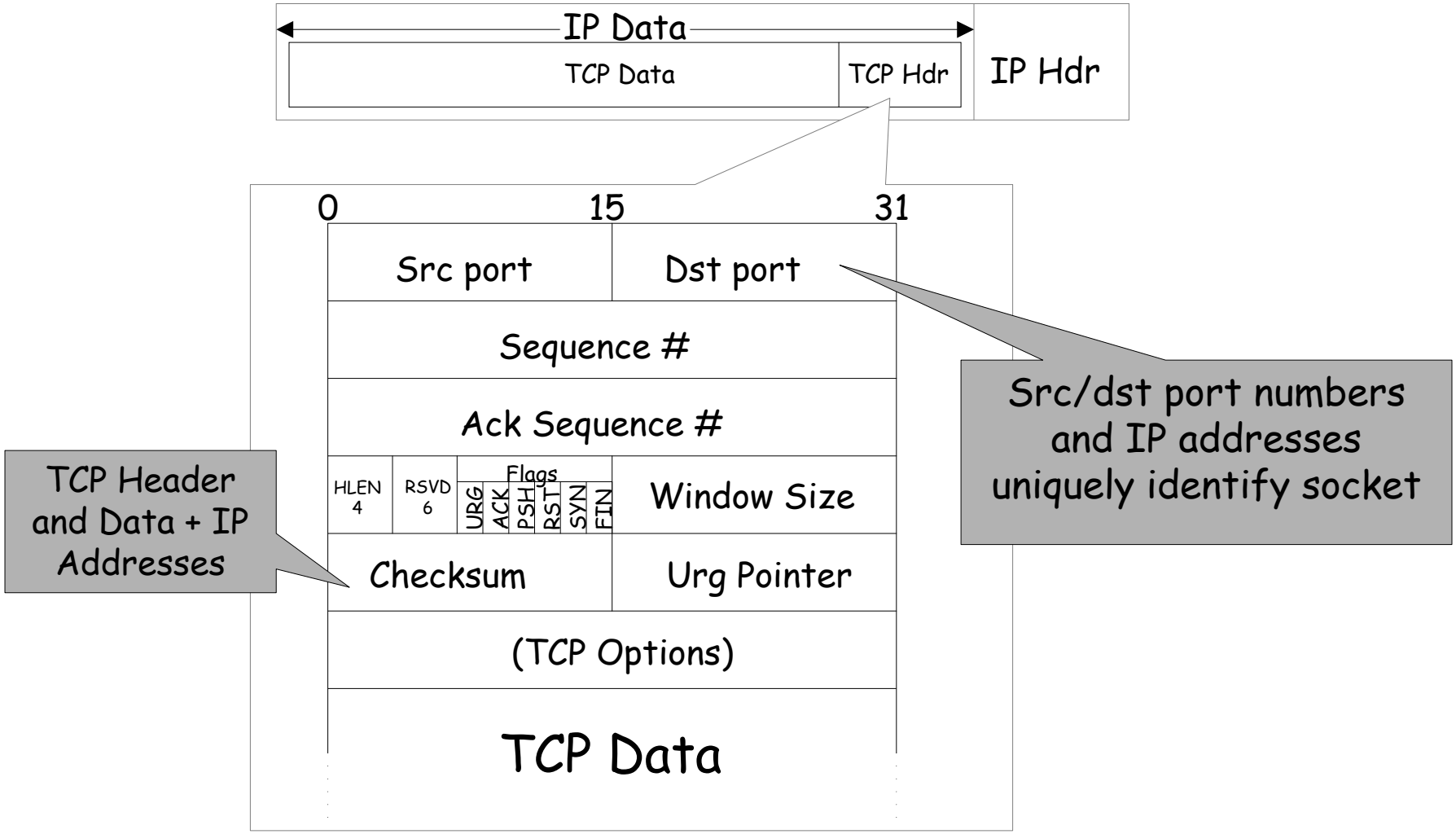
Host B



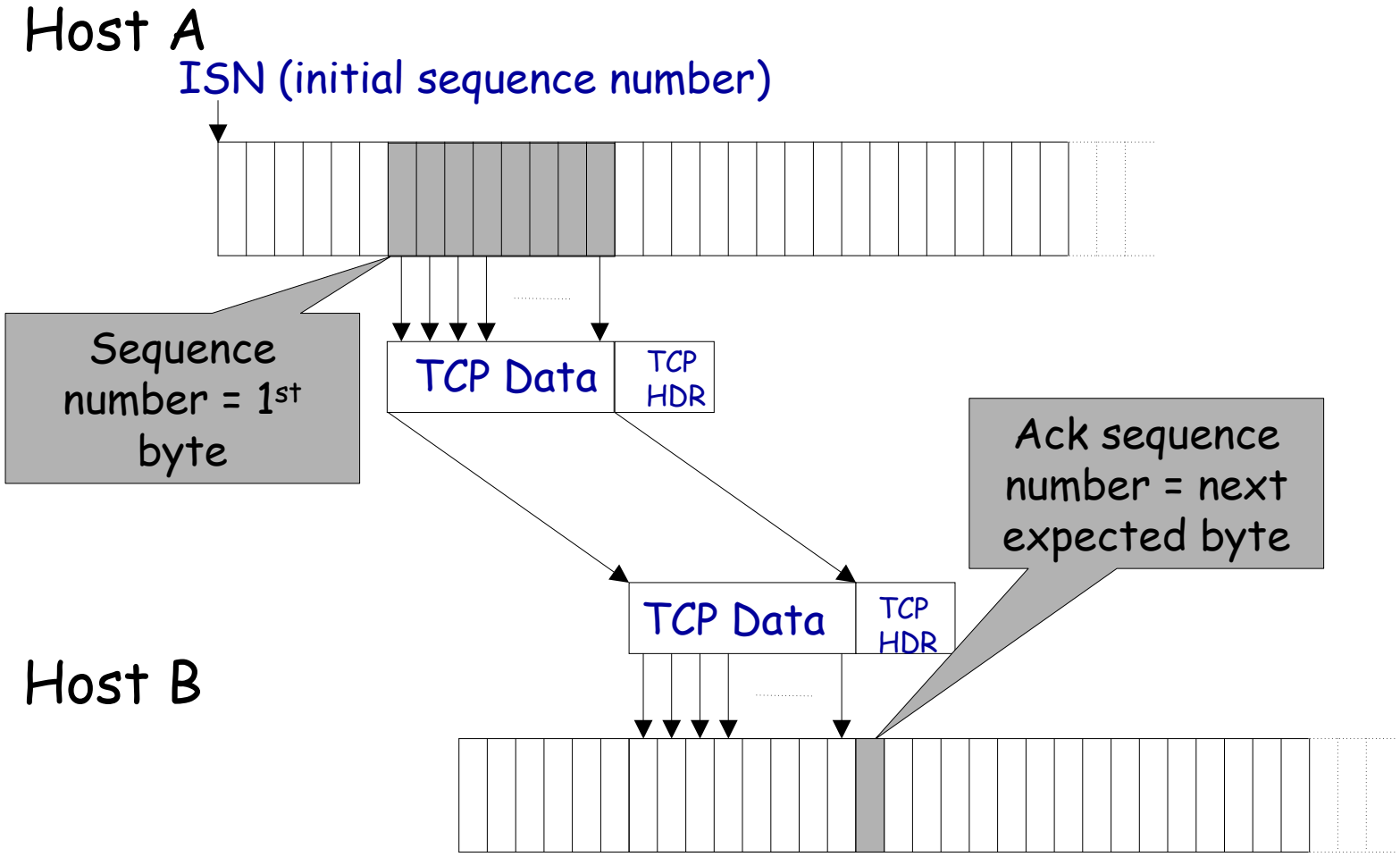
…which is emulated using TCP "segments"



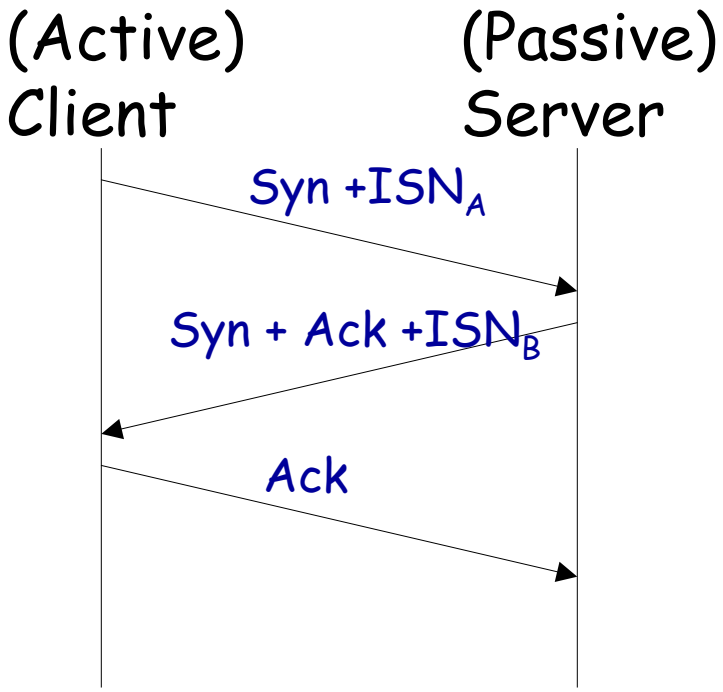
The TCP Segment Format



Sequence Numbers



Initial Sequence Numbers

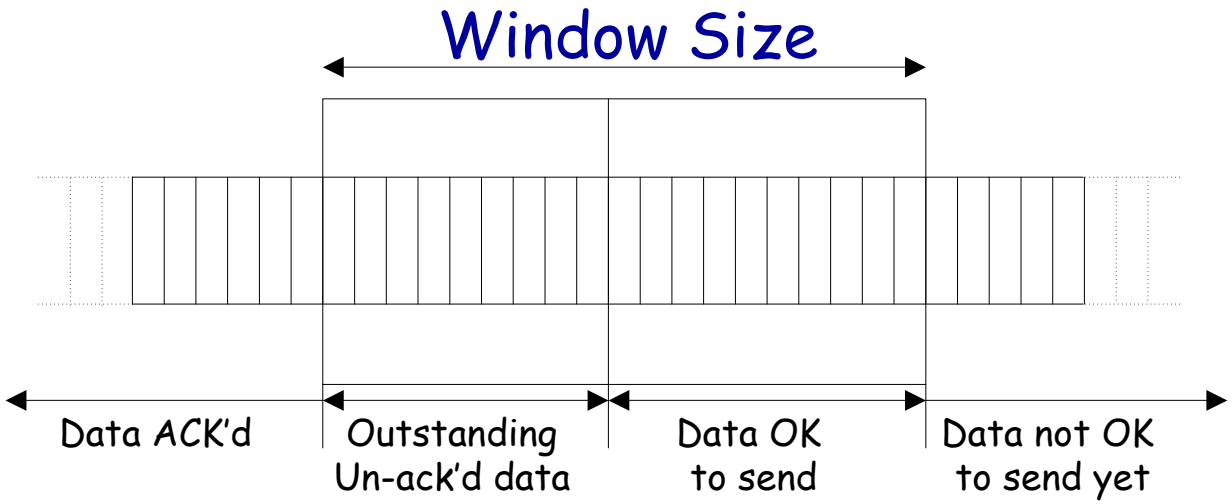


Connection Setup
3-way handshake

TCP Sliding Window

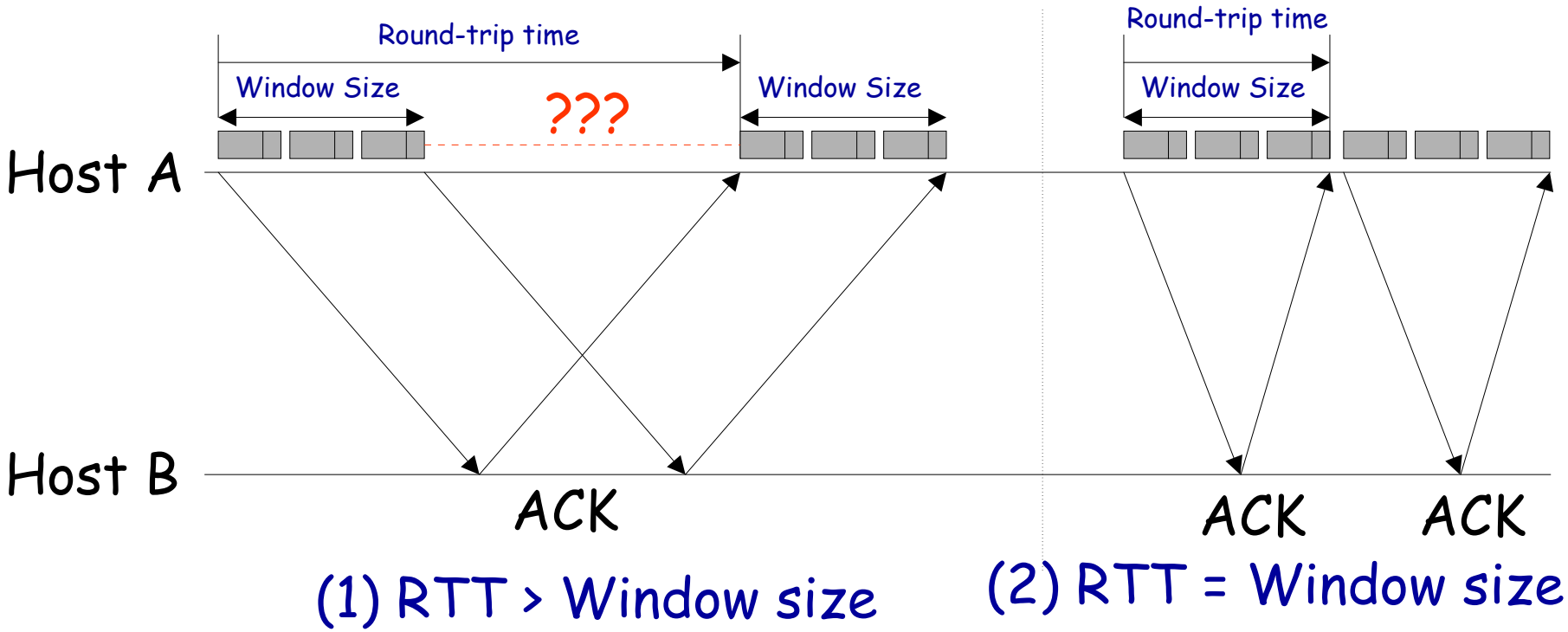
- How much data can a TCP sender have outstanding in the network?
- How much data should TCP retransmit when an error occurs? Just selectively repeat the missing data?
- How does the TCP sender avoid over-running the receiver's buffers?

TCP Sliding Window

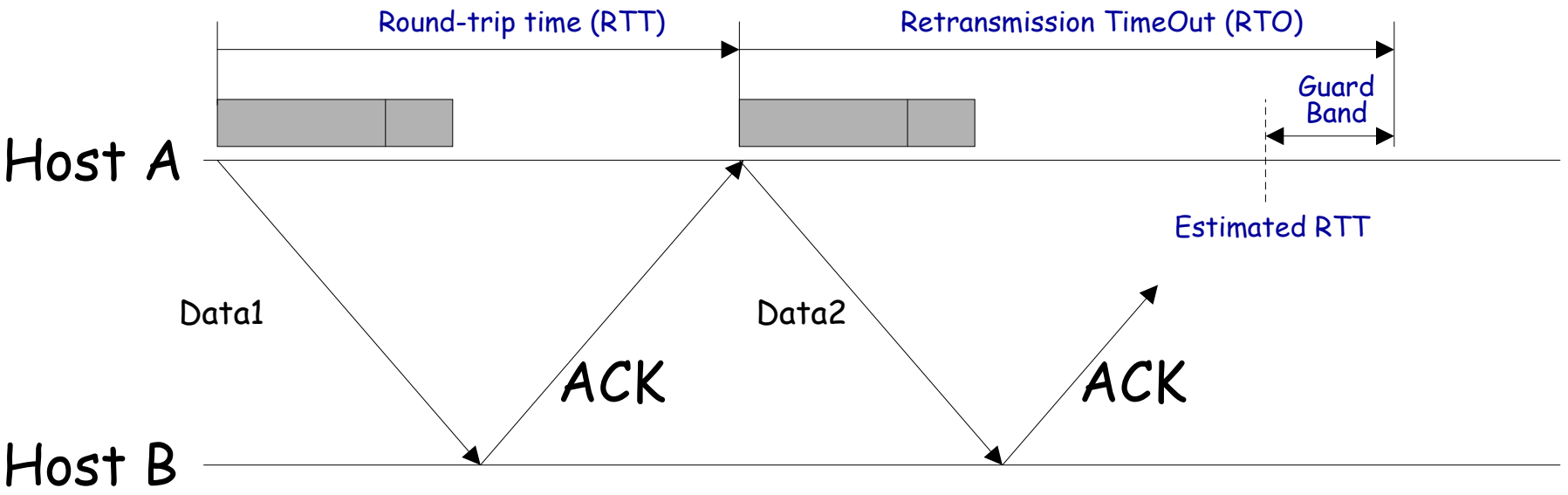


- Retransmission policy is "Go Back N"
- Current window size is "advertised" by receiver (usually 4k – 8k Bytes when connection set-up)

TCP Sliding Window



TCP: Retransmission and Timeouts



TCP uses an adaptive retransmission timeout value:

Congestion, Changes in Routing } RTT changes frequently

TCP: Retransmission and Timeouts

Picking the RTO is important:

- Pick a values that's too big and it will wait too long to retransmit a packet,
- Pick a value too small, and it will unnecessarily retransmit packets.

The original algorithm for picking RTO:

1. EstimatedRTT = α EstimatedRTT + (1 - α) SampleRTT
2. RTO = 2 * EstimatedRTT

Characteristics of the original algorithm:

- Variance is assumed to be fixed.
- But in practice, variance increases as congestion increases.

TCP: Retransmission and Timeouts

Newer Algorithm includes estimate of variance in RTT:

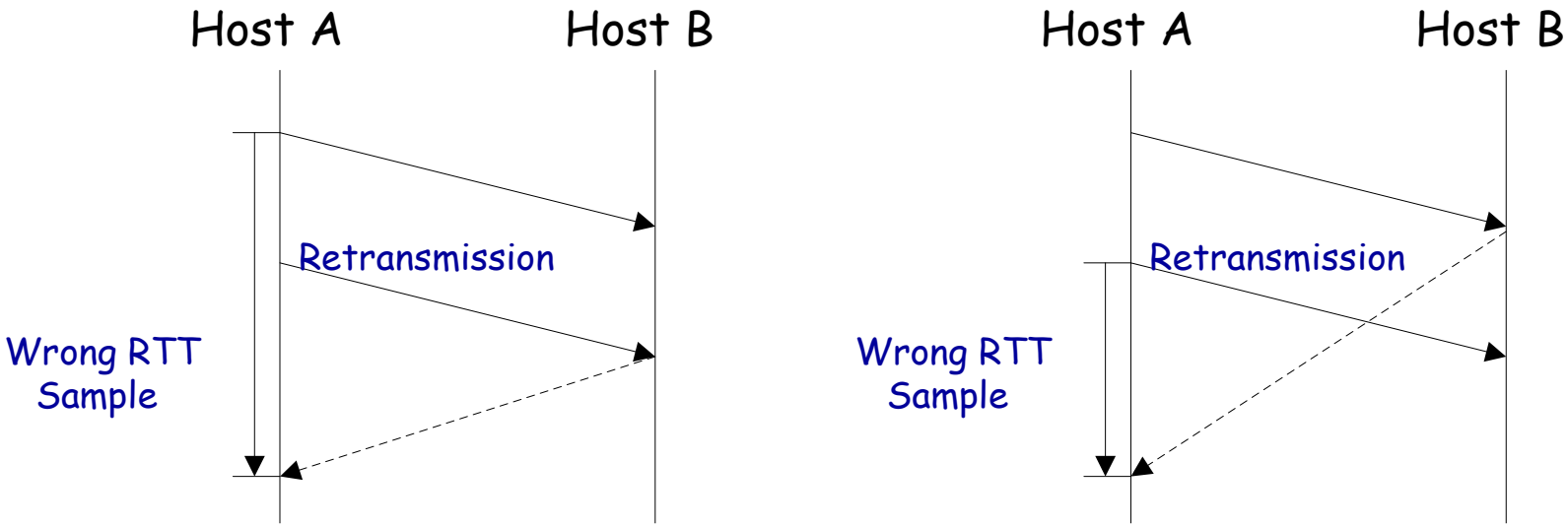
- Difference = SampleRTT - EstimatedRTT
- EstimatedRTT = EstimatedRTT + (δ*Difference)
- Deviation = Deviation + δ*(|Difference| - Deviation)

- RTO = μ * EstimatedRTT + ϕ * Deviation

  $\mu \approx 1$

  $\phi \approx 4$

TCP: Retransmission and Timeouts Karn's Algorithm

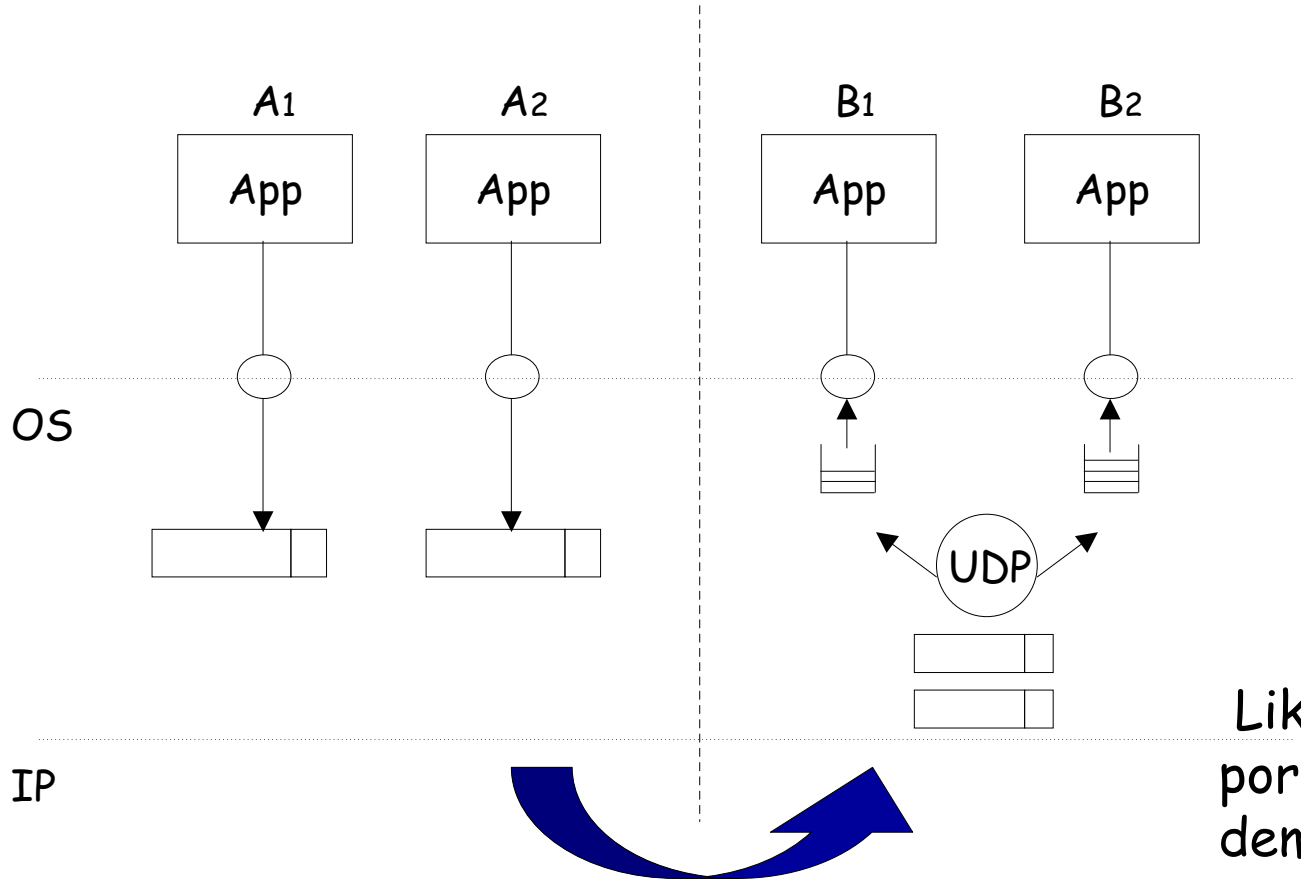


Problem:
How can we estimate RTT when packets are retransmitted?
Solution:
On retransmission, don't update estimated RTT (and double RTO)

User Datagram Protocol (UDP) Characteristics

- UDP is a connectionless datagram service
  - There is no connection establishment: packets may show up at any time
- UDP packets are self-contained
- UDP is unreliable:
  - No acknowledgements to indicate delivery of data
  - Checksums cover the header, and only optionally cover the data
  - Contains no mechanism to detect missing or mis-sequenced packets
  - No mechanism for automatic retransmission
  - No mechanism for flow control, and so can over-run the receiver

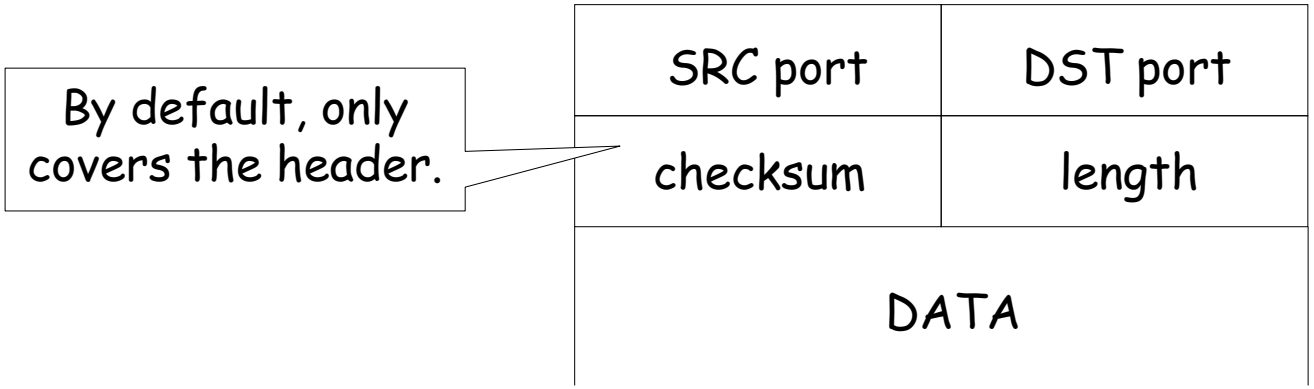
User-Datagram Protocol (UDP)



Like TCP, UDP uses port number to demultiplex packets

User-Datagram Protocol (UDP)

Packet format



- Why do we have UDP?
  - It is used by applications that don't need reliable delivery, or
  - Applications that have their own special needs, such as streaming of real-time audio/video.

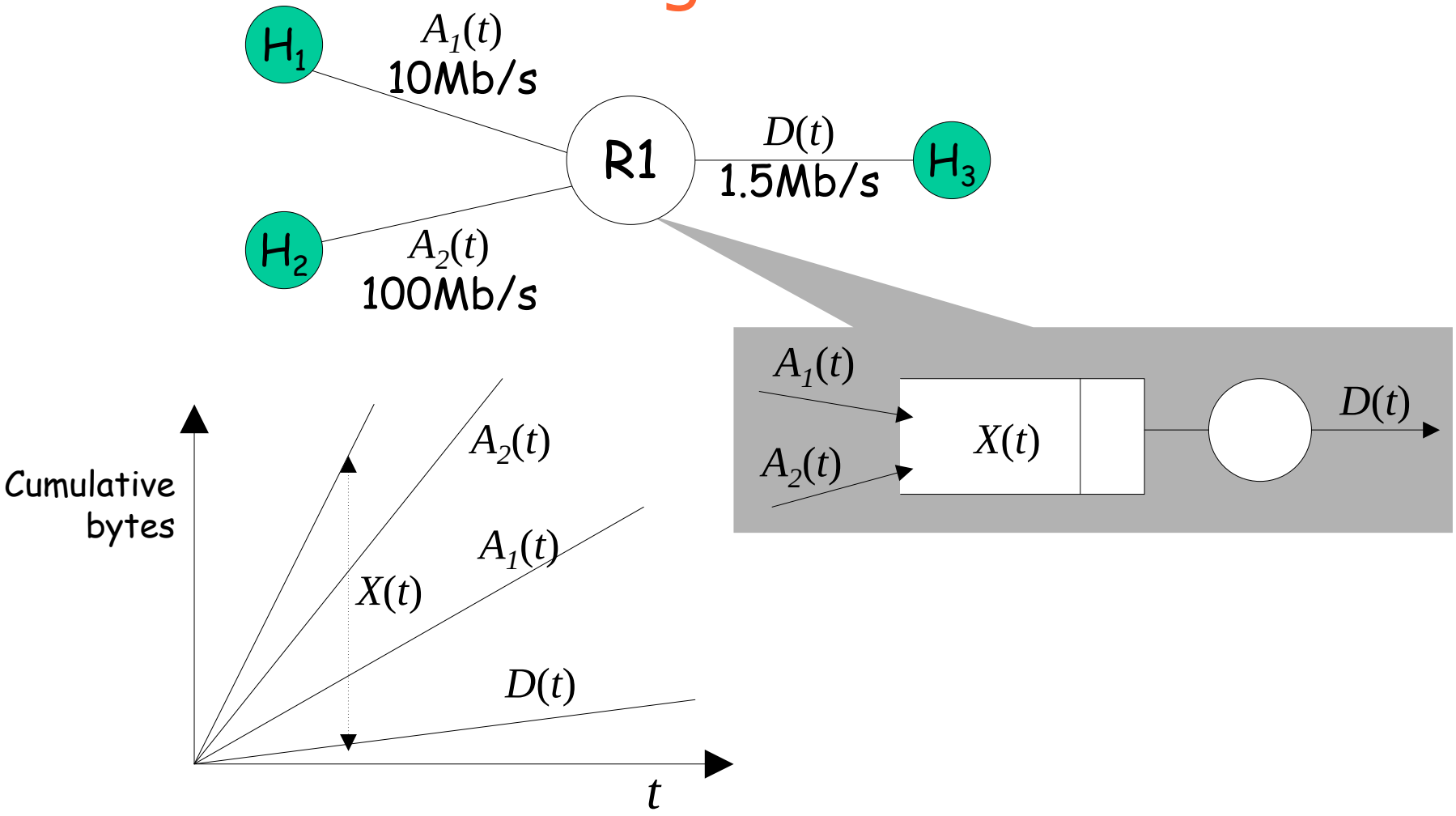
This Week: Routing and Transport

- Routing techniques
  - Flooding
  - Distributed Bellman Ford Algorithm
  - Dijkstra's Shortest Path First Algorithm
- Routing in the Internet
  - Hierarchy and Autonomous Systems
  - Interior Routing Protocols: RIP, OSPF
  - Exterior Routing Protocol: BGP
- Transport: achieving reliability
- **Transport: achieving fair sharing of links**

Main points

- Congestion is inevitable
- TCP sources detect congestion and, co-operatively, reduce the rate at which they transmit
- The rate is controlled using the TCP window size
- TCP modifies the rate according to “Additive Increase, Multiplicative Decrease (AIMD)”
- To jump start flows, TCP uses a fast restart mechanism (called “slow start”!)
- TCP achieves high throughput by encouraging high delay

Congestion

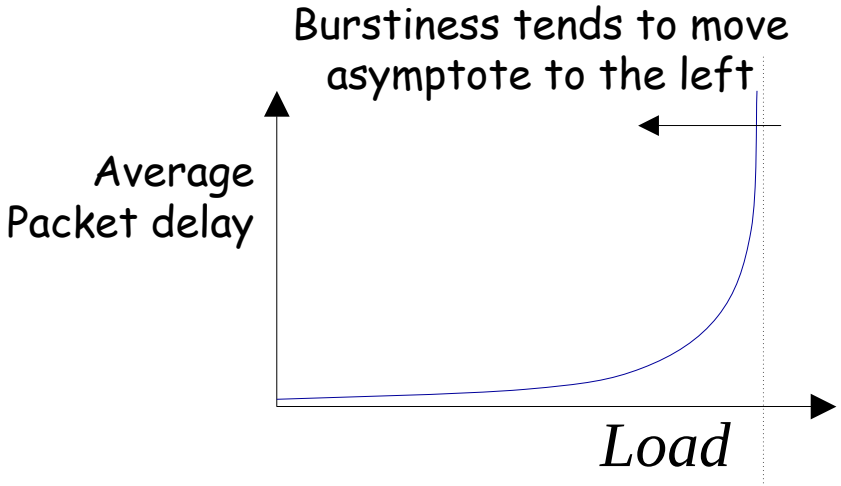

H1
A1(t)
10Mb/s
R1
D(t)
1.5Mb/s
H3
H2
A2(t)
100Mb/s
A1(t)
A2(t)
X(t)
D(t)
Cumulative
bytes
A2(t)
A1(t)
X(t)
D(t)
t

Congestion is unavoidable
Arguably it's good!

- We use packet switching because it makes efficient use of the links. Therefore, buffers in the routers are frequently occupied
- If buffers are always empty, delay is low, but our usage of the network is low
- If buffers are always occupied, delay is high, but we are using the network more efficiently
- So how much congestion is too much?

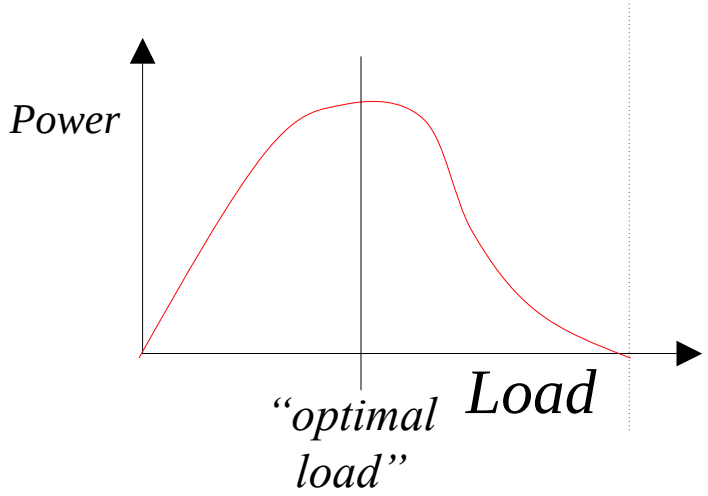
Load, Delay and Power

Typical behavior of queueing systems with random arrivals:



A simple metric of how well the network is performing:

$$Power = \frac{Load}{Delay}$$



Options for Congestion Control

1. Implemented by host versus network
2. Reservation-based, versus feedback-based
3. Window-based versus rate-based

TCP Congestion Control

- TCP implements host-based, feedback-based, window-based congestion control.
- TCP sources attempts to determine how much capacity is available
- TCP sends packets, then reacts to observable events (loss)

TCP Congestion Control

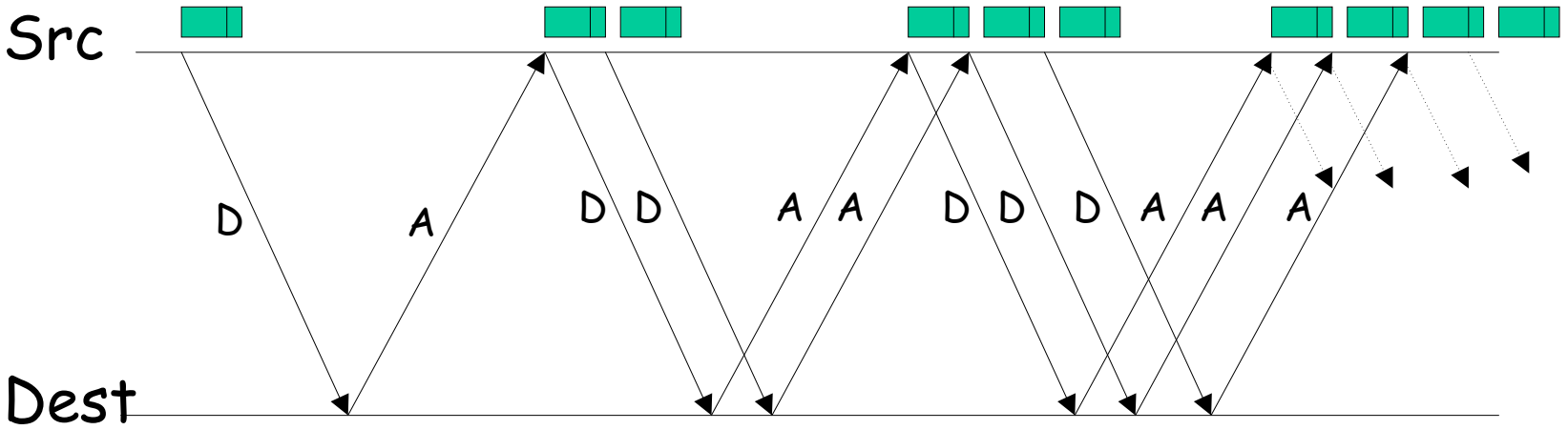
- TCP sources change the sending rate by modifying the window size:

  Window = min{Advertized window, Congestion Window}

  Receiver — Transmitter ("cwnd")

- In other words, send at the rate of the slowest component: network or receiver
- "cwnd" follows additive increase/multiplicative decrease
  - On receipt of Ack: cwnd += 1
  - On packet loss (timeout): cwnd *= 0.5

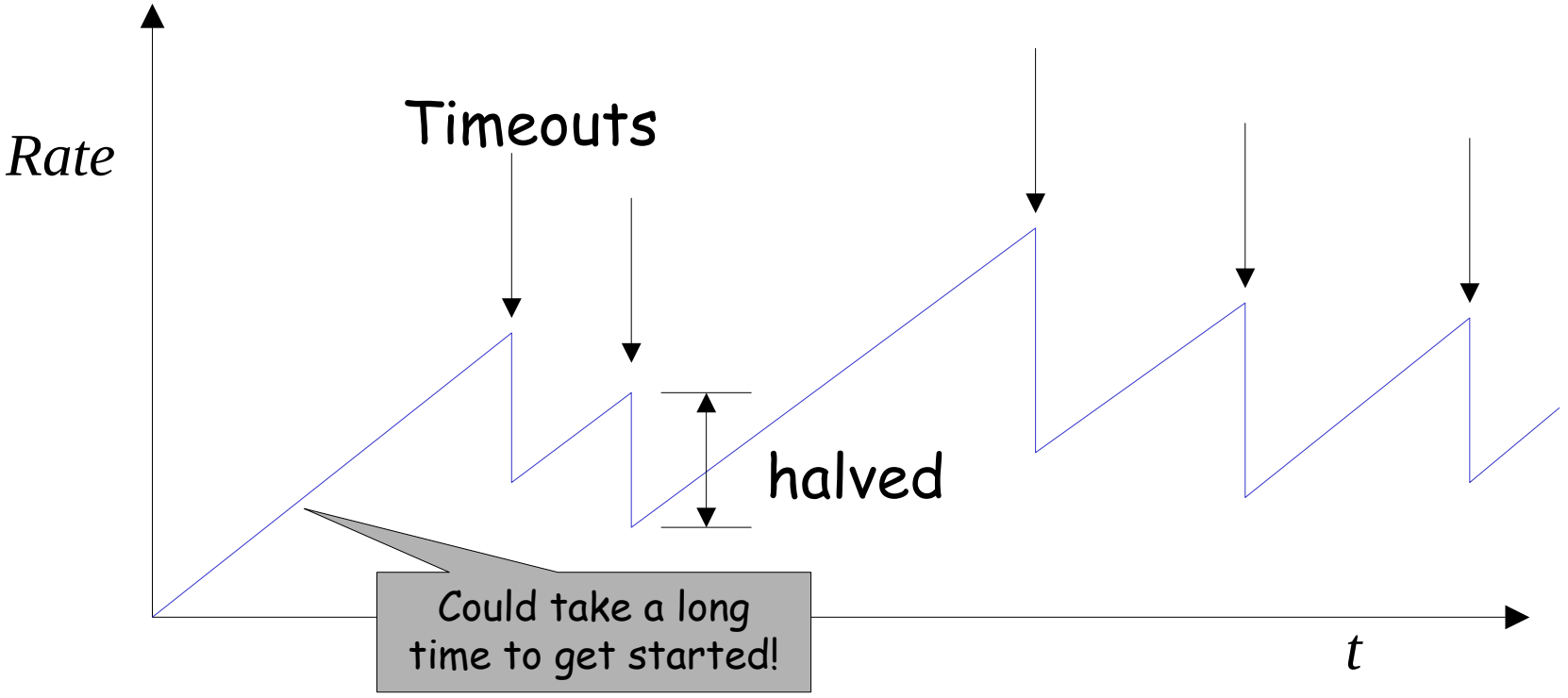
Additive Increase



Actually, TCP uses bytes, not segments to count:
When ACK is received:

$$cwnd += MSS\left(\frac{MSS}{cwnd}\right)$$

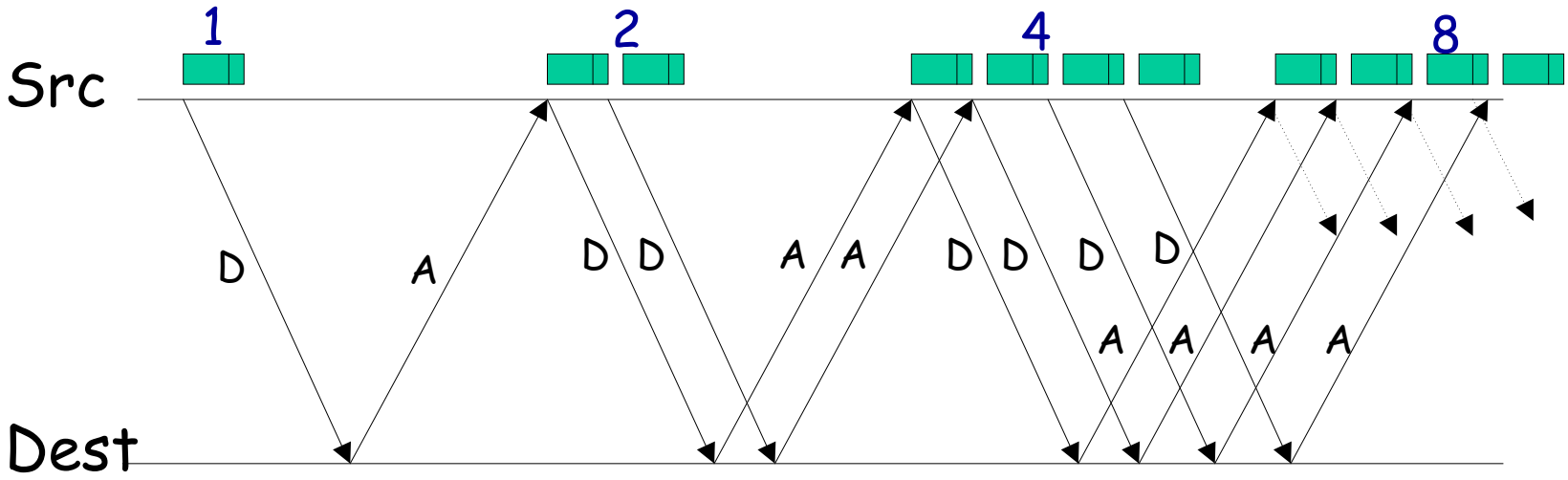
Leads to the TCP "sawtooth"



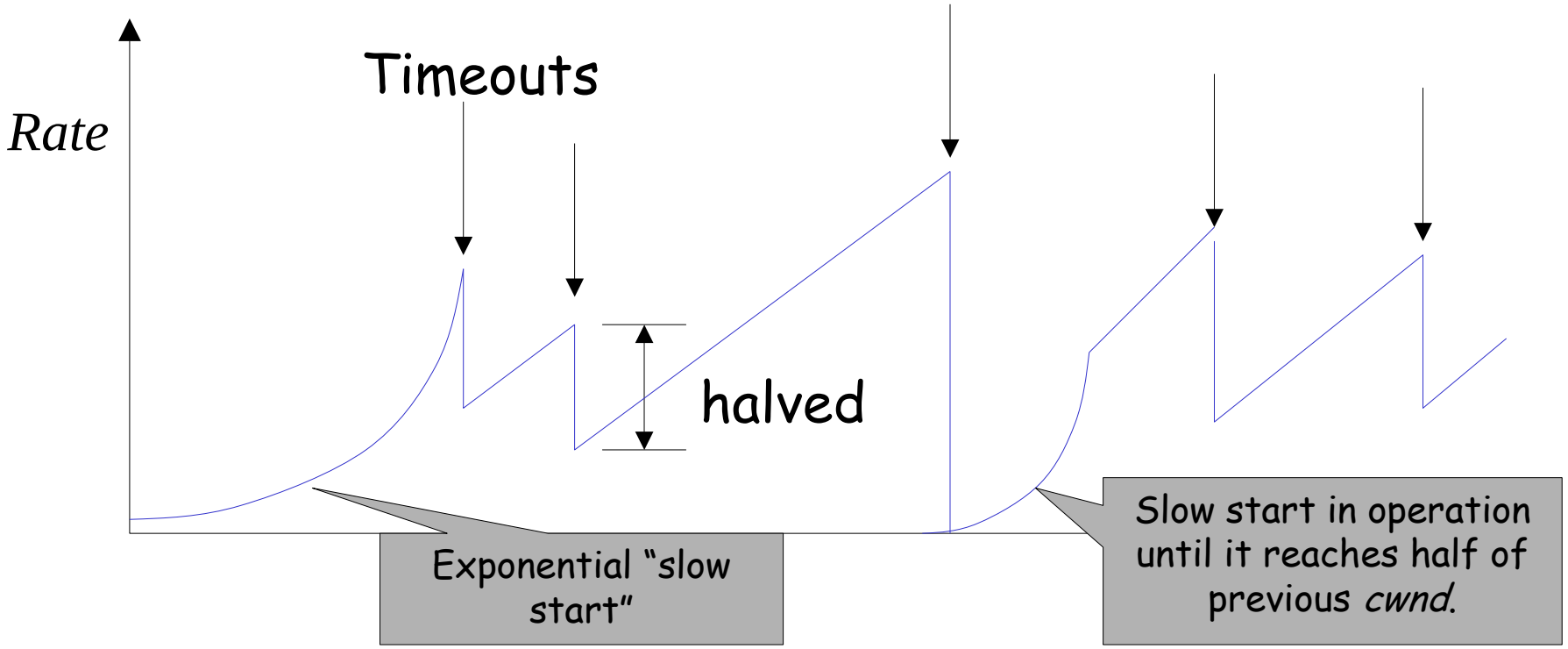
"Slow Start"

Designed to cold-start connection quickly at startup or if a connection has been halted (e.g. window dropped to zero, or window full, but ACK is lost).

How it works: increase cwnd by 1 for each ACK received.



Slow Start



Why is it called slow-start? Because TCP originally had no congestion control mechanism. The source would just start by sending a whole window's worth of data.

Fast Retransmit & Fast Recovery

- TCP source can take advantage of an additional hint: if a duplicate ACK arrives out of sequence, there was probably some data lost, even if it hasn't yet timed out.
- Upon 3 duplicate ACKs, TCP retransmits.
- Does not enter slow-start: there are probably ACKs in the pipe that will continue correct AIMD operation.

Course Outline (Subject to Change)

1. (January 9th) Internet design principles and protocols
2. (January 16th) Internetworking, transport, routing
3. (January 23rd) Mapping the Internet and other networks
4. (January 30th) Security (with guest lecturer: Gene Spafford)
5. (February 6th) P2P technologies & applications (Matei Ripeanu)

(plus midterm)

6. (February 13th) Optical networks (Charlie Catlett)
7. *(February 20th) Web and Grid Services (Steve Tuecke)
8. (February 27th) Network operations (Greg Jackson)
9. *(March 6th) Advanced applications (with guest lecturers: Terry Disz, Mike Wilde)
10. (March 13th) Final exam

* Ian Foster is out of town.