
RIP, UDP, TCP I

9 May 2010
Lecture 8

Slides Credits: Steve Zdancewic (UPenn)

Topics for Today

- RIP
- UDP
- TCP

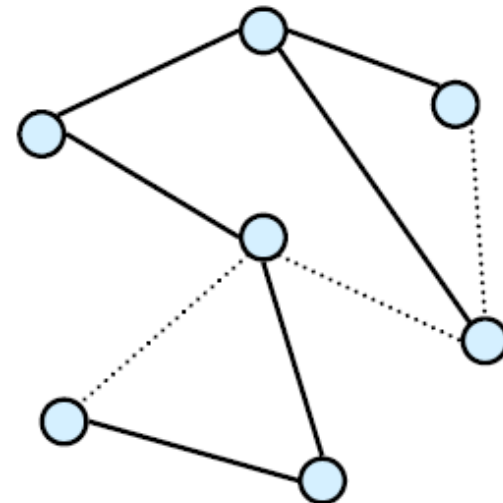
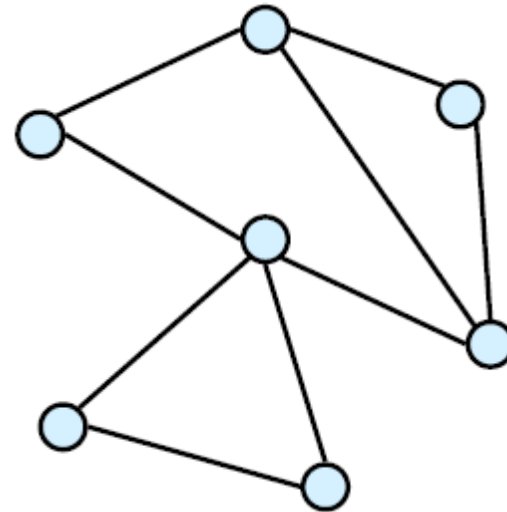
- Sources in PD:
 - RIP: 4.2.1-4.2.2
 - UDP: 5.1
 - TCP: 5.2, 6.3

Intradomain Routing

- RIP - Routing Information Protocol
 - Uses distance vector algorithm
 - Limited to small nets; <15 hops
- OSPF - Open Shortest Path First
 - Augmented version of link-state
 - Augmentation includes authentication, load balancing, and defined areas
 - Due to time pressure we'll talk about this later (only חשמל ואלקטרוניקה)

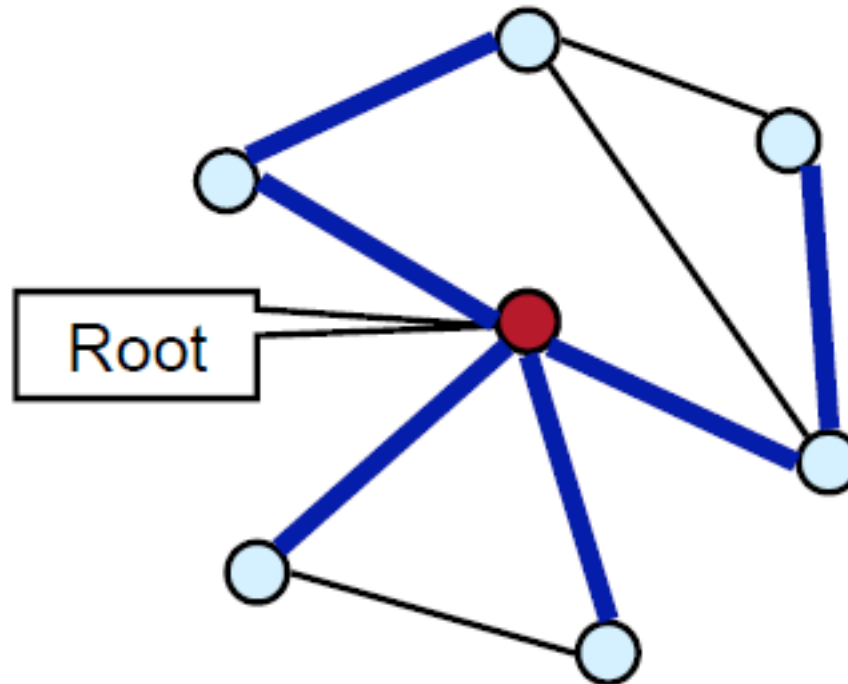
Spanning Trees (Abstractly)

- Given a connected graph G
- A *spanning tree* is an acyclic, connected subgraph of G that contains all the nodes.



Spanning Tree Algorithm (Abstractly)

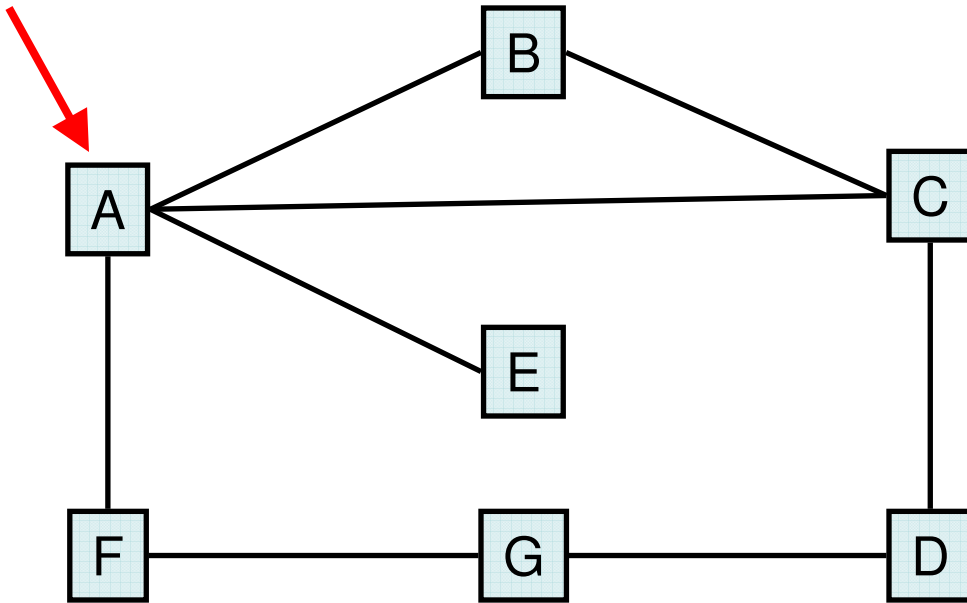
- Pick a root node
 - Compute shortest paths to root
 - Need to break ties



Distance Vector Algorithm (RIP)

- Similar to the Spanning Tree Algorithm
 - Except that information about distance to ALL nodes is forwarded (not just info. about root.)
 - Sometimes called Bellman-Ford algorithm
- Each node constructs a *Distance Vector*
 - Contains distances (costs) to reach all other node
 - Initially:
 - Distance to neighbors (a simplification for now) = 1
 - Distance to others = ∞
 - Routing table reflects node's beliefs

Example Network Graph



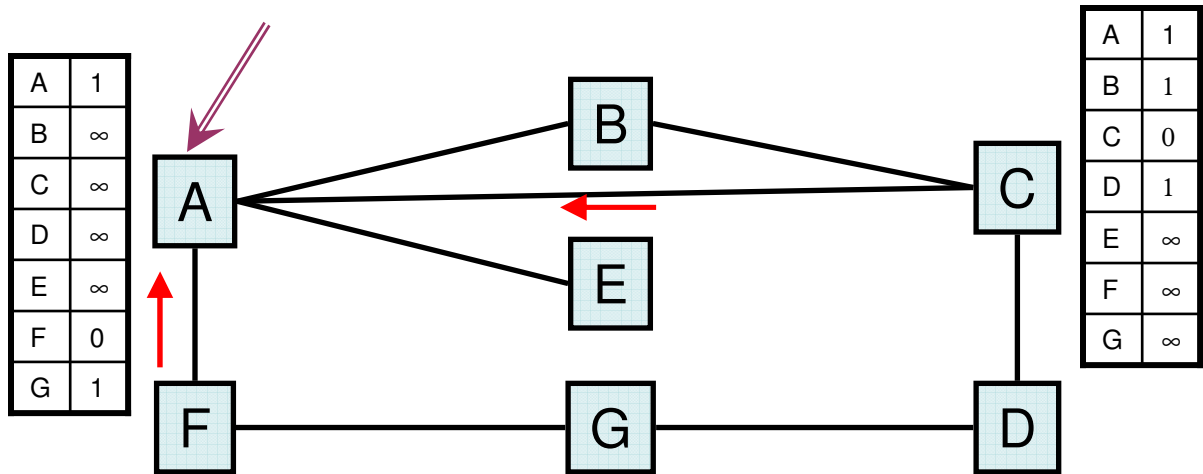
A's initial information:

Dest	Cost	NextHop
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-

Iteration Steps

- Each host sends its DV (cost) to its neighbors
- Neighbors can update their distance vectors and routing information accordingly.
 - As in spanning tree, the nodes ignore worse information
 - Update any better routes
- If host changed its tables, send new DV to neighbors
- After a few iterations, routing information *converges*

Example Iteration Steps



F sends A its DV

- A discovers that G can be reached in to two hops via F

C send A its DV

- A discovers that D can be reached in two hops via C

Dest	Cost	NextHop	Dest	Cost	NextHop	Dest	Cost	NextHop
B	1	B	B	1	B	B	1	B
C	1	C	C	1	C	C	1	C
D	∞	-	D	∞	-	D	2	C
E	1	E	E	1	E	E	1	E
F	1	F	F	1	F	F	1	F
G	∞	-	G	2	F	G	2	F

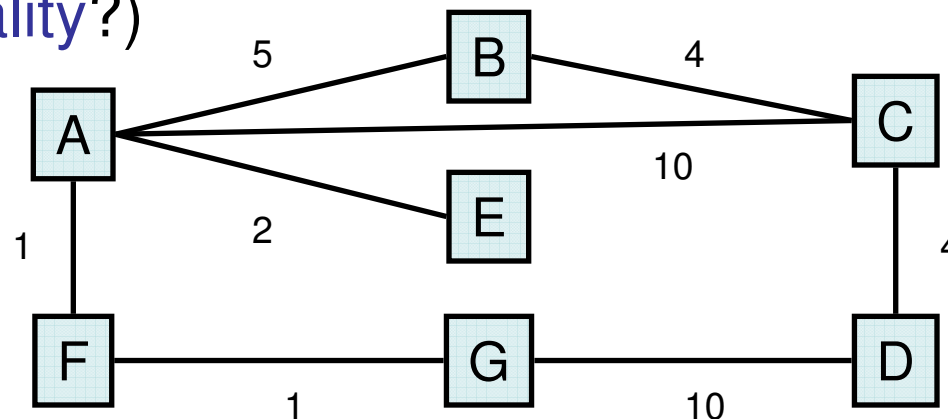
Details

- Note: No single host has all routing information.
- When to send update vectors?
 - When your routing table changes (triggered)
 - Periodically (“I’m alive!”)
- Detecting link/node failure
 - (1) Periodically exchange “I’m alive!” messages.
 - (2) Timeout mechanism

Adding Weights

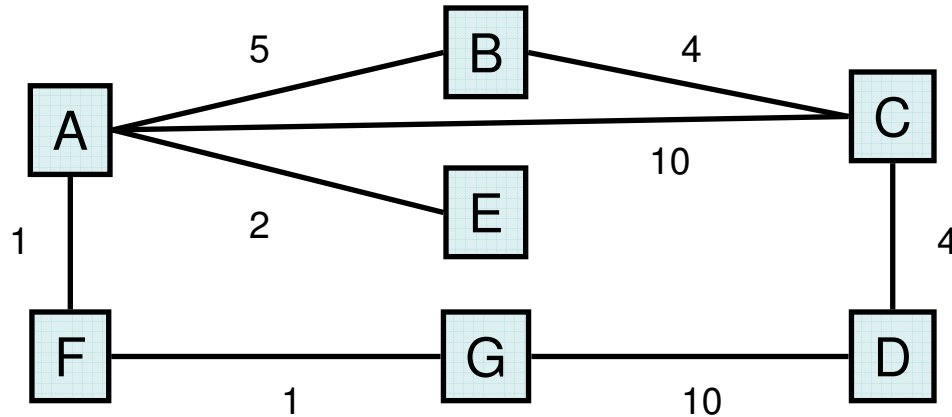
In a **static network**, if all weights are 1, once A discovers a path to B (cost $< \infty$), no subsequent round will ever discover a better path to B

What if weights are greater than 1 (and may break the triangle inequality?)



- We may discover **better routes** later
- We **may** need to **update the table** each round

Weights Example

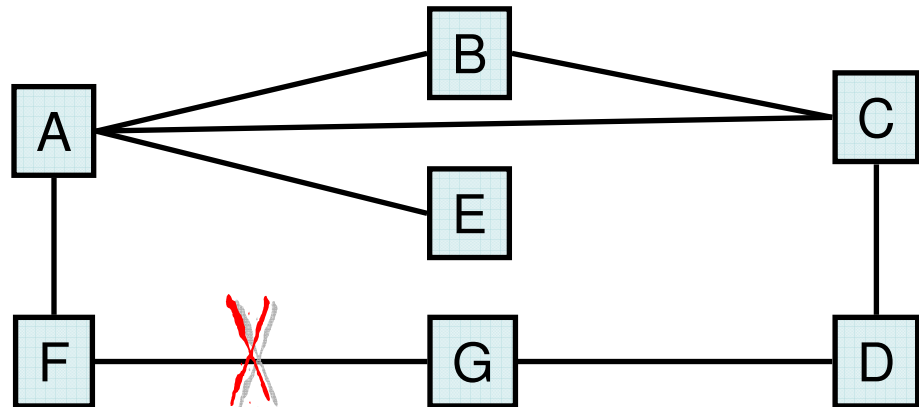
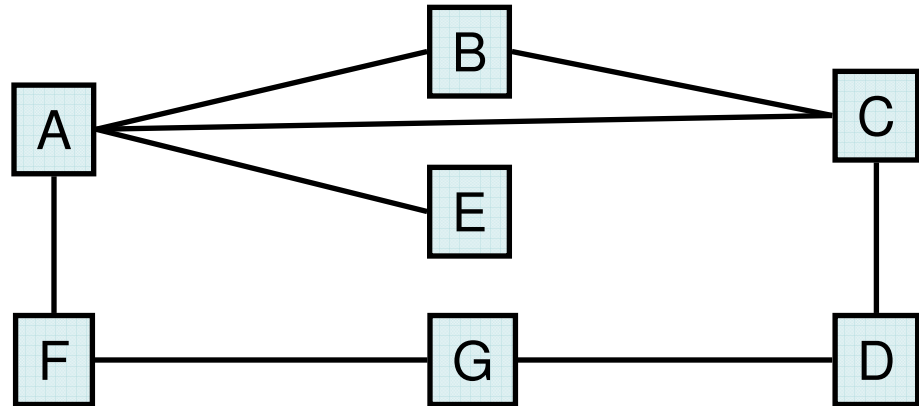


A			B			C			D			E			F			G		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A			A			A			A			A			A		
B			B	-	-	B			B			B			B			B		
C			C			C	-	-	C			C			C			C		
D			D			D			D	-	-	D			D			D		
E			E			E			E			E			E			E		
F			F			F			F			F			F			F		
G			G			G			G			G			G			G		

Dynamic Networks

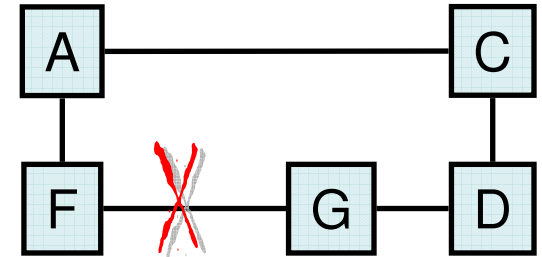
What about a dynamic network?

- A node enters
 - The new neighbors will eventually inform everyone else (easy)
- A node/link fails
 - Is the network partitioned?
 - How do nodes discover that a node/link is gone?



Link Failure Example

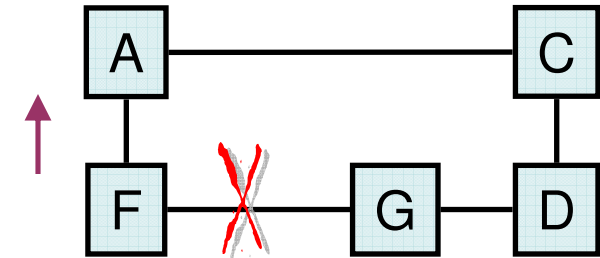
A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	2	F	G	2	D	G	1	G	G	1	G



A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	2	F	G	2	D	G	1	G	G	∞	-

Link Failure Example 1

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	2	F	G	2	D	G	1	G	G	∞	-

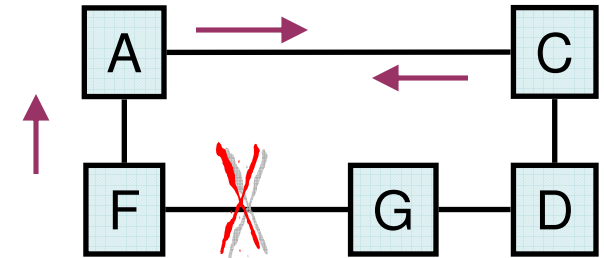


Case 1:
1. F sends to A

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	∞	-	G	2	D	G	1	G	G	∞	-

Link Failure Example 1

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	∞	-	G	2	D	G	1	G	G	∞	-



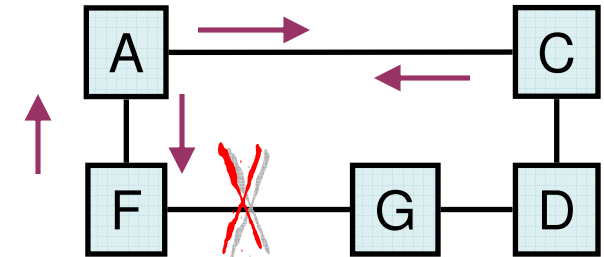
Case 1:

1. F sends to A
2. A sends to C
 - (nothing)
3. C sends to A

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	3	C	G	2	D	G	1	G	G	∞	-

Link Failure Example 1

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	3	C	G	2	D	G	1	G	G	∞	-



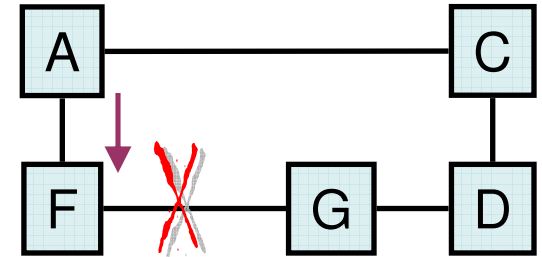
Case 1:

1. F sends to A
2. A sends to C
 - (nothing)
3. C sends to A
4. A sends to F

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	3	C	G	2	D	G	1	G	G	4	A

Link Failure Example 2

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	2	F	G	2	D	G	1	G	G	∞	-



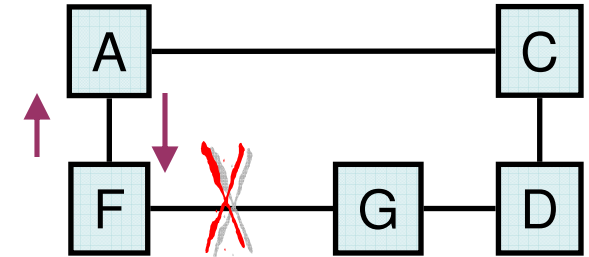
Case 2:

1. A sends to F

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	2	F	G	2	D	G	1	G	G	3	A

Link Failure Example 2

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	2	F	G	2	D	G	1	G	G	3	A



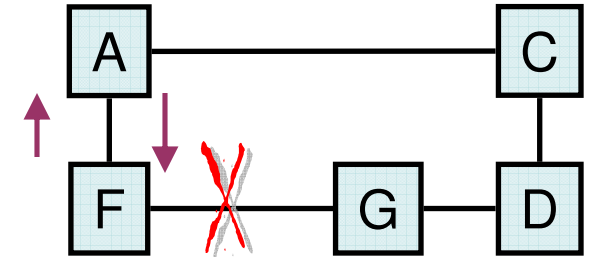
Case 2:

1. A sends to F
2. F sends to A

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	4	F	G	2	D	G	1	G	G	3	A

Link Failure Example 2

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	4	F	G	2	D	G	1	G	G	3	A



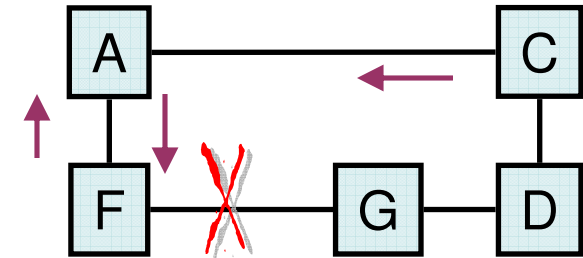
Case 2:

1. A sends to F
2. F sends to A
3. A sends to F

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	4	F	G	2	D	G	1	G	G	5	A

Link Failure Example 2

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	4	F	G	2	D	G	1	G	G	5	A



Case 2:

1. A sends to F
2. F sends to A
3. A sends to F
4. ...
5. C sends to A
6. A sends to F

A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	3	C	G	2	D	G	1	G	G	4	A

Network Partition

- The loop between A and F is broken only by C's update
 - C has **real information** about another path to G
- What if C's update **never** comes? What if C's information is **false**?
 - The network is partitioned –OR–
 - G is completely offline
- The other nodes will continue counting until infinity (or the distance field reaches its max)

Two RIP Solutions

Option 1: Infinity is small

- Don't let the distance fields go above a predefined maximum
 - Say 15, 20, etc.
 - The max must be greater than the diameter of the network
-

Option 2: Include “Next hop” in the Distance Vector

- Ignore routes which pass via you

Example: F receives an update from A about a path to G.

- In the example before, F didn't know that's A's “path” went via F
- If A had sent

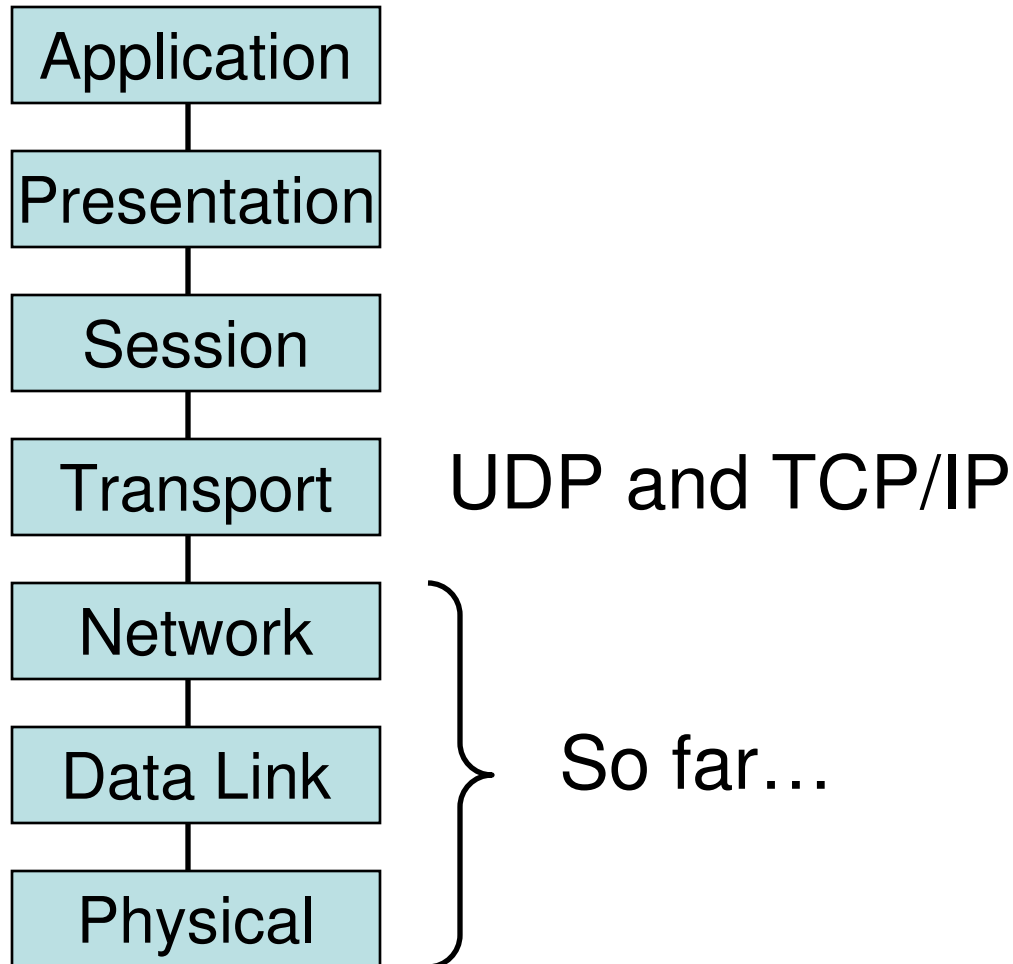
Dest	C	Nexthop
G	2	F

 instead, F would ignore the fake path
- Called “split horizon”

So Far

- RIP
- UDP
- TCP

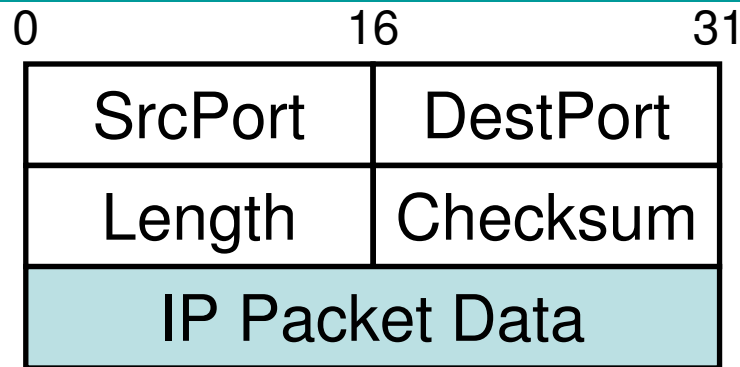
Protocol Stack Revisited



Application vs. Network

Application Needs	Network Char.
Reliable, Ordered, Single-Copy Message Delivery	Drops , Duplicates and Reorders Messages
Arbitrarily large messages	Finite message size
Flow Control by Receiver	Arbitrary Delay
Supports multiple applications per-host	...

User Datagram Protocol (UDP)

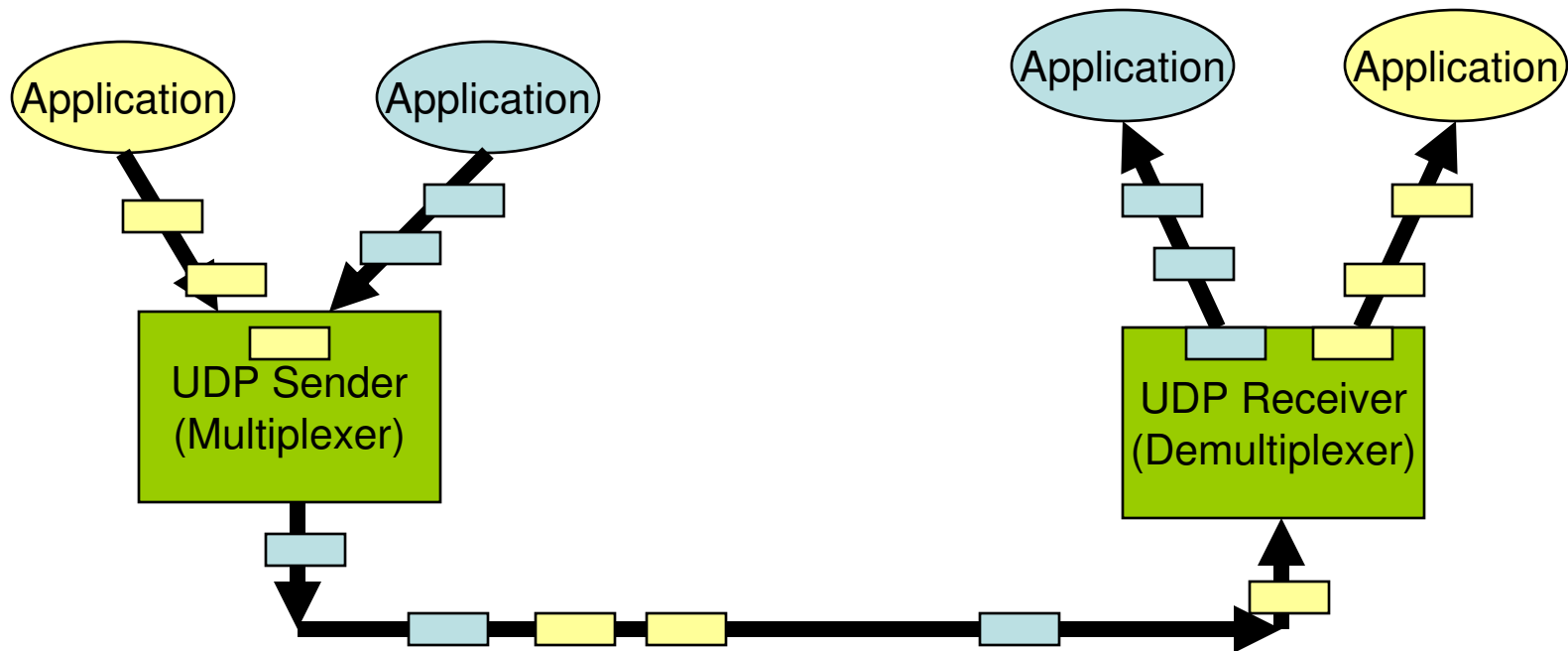


- Simplest transport-layer protocol
- Just exposes IP packet functionality to application level
- *Ports* identify sending/receiving process
 - Demultiplexing information
 - (port, host) pair identifies a network process

Question: Why is there another length field?

UDP End-to-End Model

- Multiplexing/Demultiplexing with Port number



Using Ports

- Client contacts Server at a *well-known port*
 - SMTP: port 25
 - DNS: port 53
 - POP3: port 110
 - Unix talk : port 517
 - In unix, ports are listed in `/etc/services`
- Client and Server agree on a different port for subsequent communication
- Ports are an abstraction
 - Implemented differently on different OS's
 - Typically a message queue

So Far

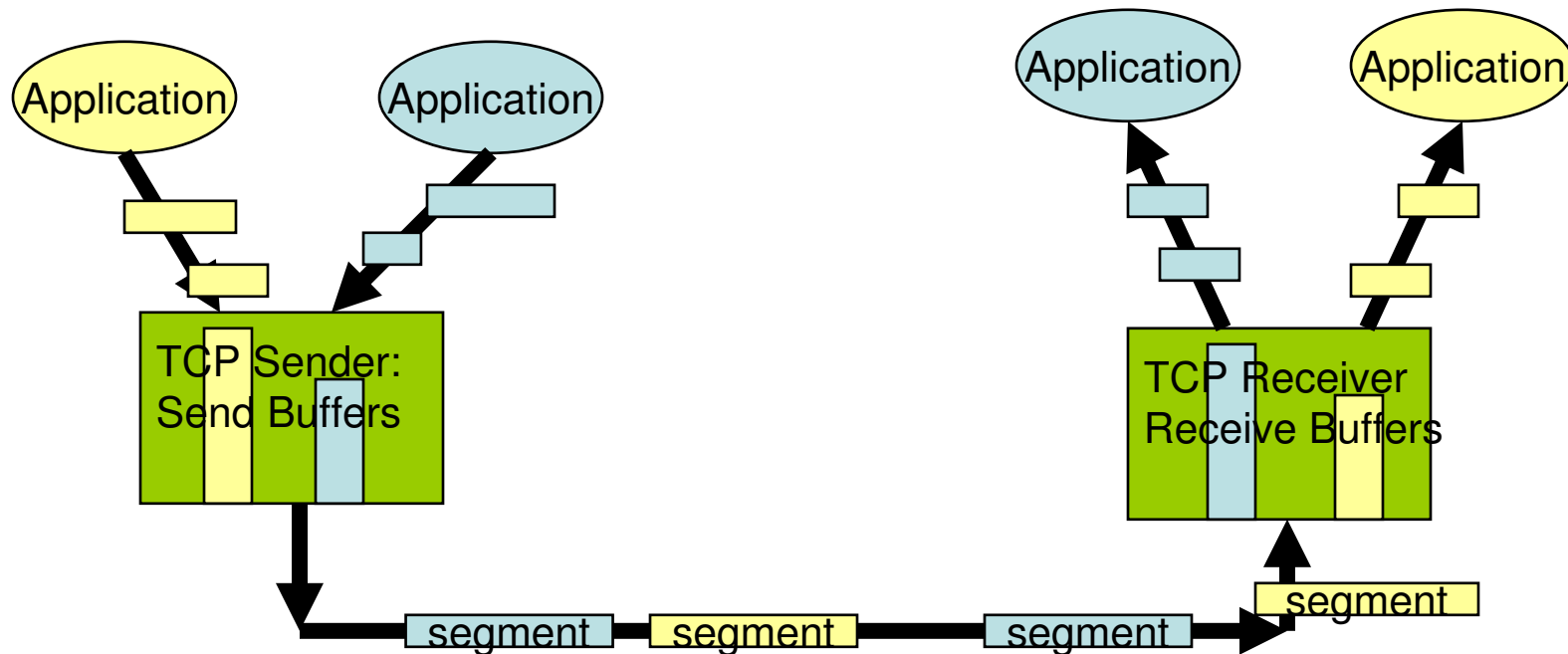
- RIP
- UDP
- TCP

Transmission Control Protocol (TCP)

- Most widely used protocol for reliable byte streams
 - Reliable, in-order delivery of a stream of bytes
 - Full duplex: pair of streams, one in each direction
 - Flow and congestion control mechanisms
 - Like UDP, supports ports
- Built on top of IP (hence TCP/IP)

TCP End-to-End Model

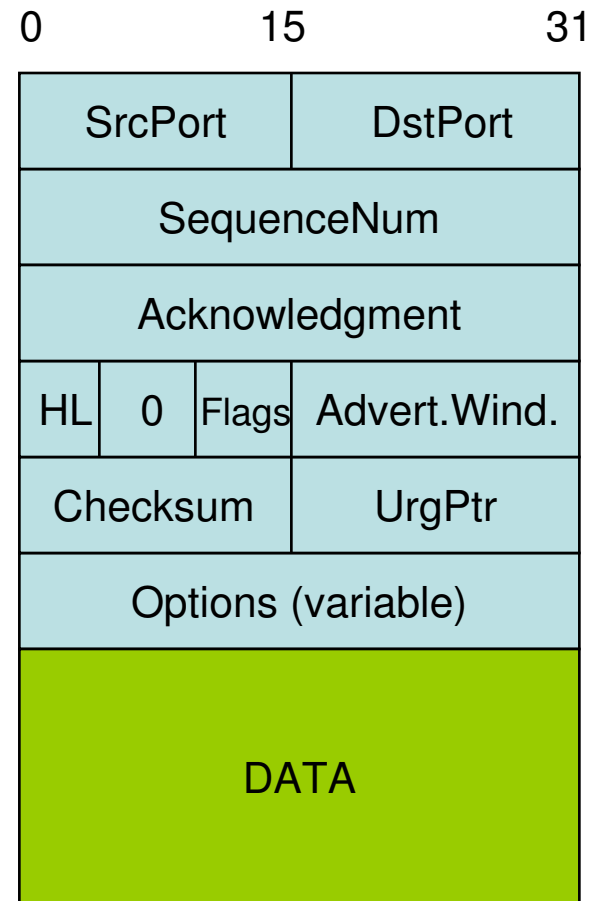
- Buffering corrects errors but may introduce delays



Packet Format

- Flags
 - SYN
 - FIN
 - RESET
 - PUSH
 - URG
 - ACK

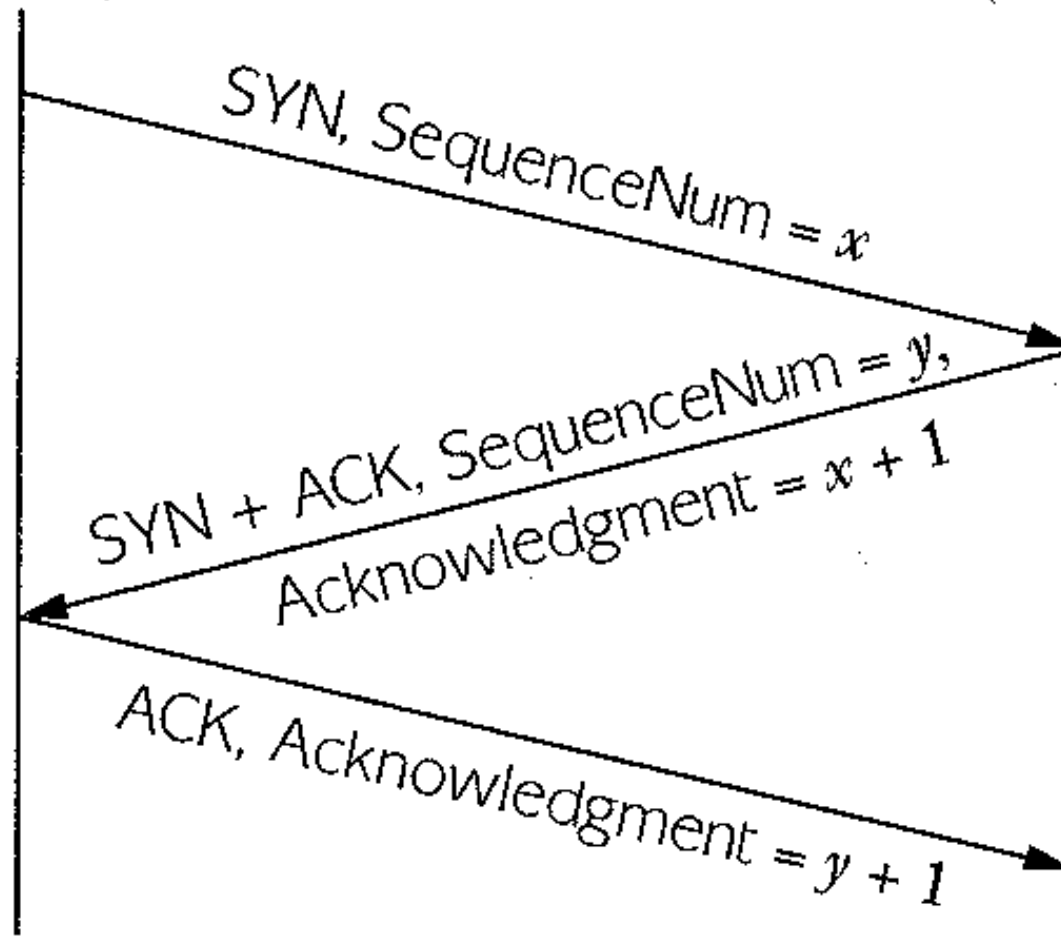
- Fields



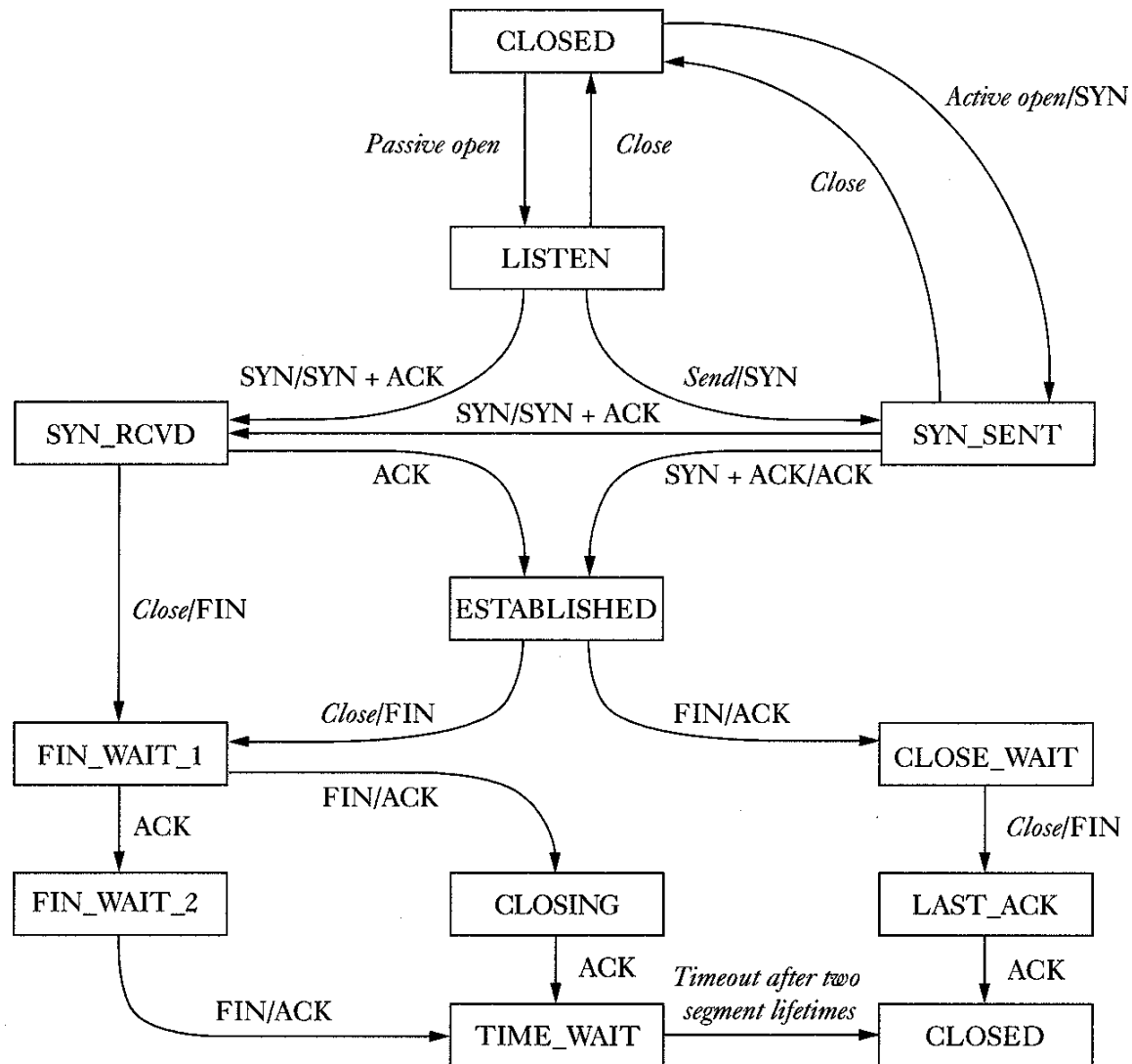
Three-Way Handshake

Active participant
(client)

Passive participant
(server)



TCP State Transitions



TCP Receiver

- Maintains a buffer from which application reads
- Advertises $N < \text{buffer size}$ as the window for sliding window
- Responds with Acknowledge and AdvertisedWindow on each send; updates byte counts when data O.K.
- Application blocked until read() O.K.

TCP Sender

- Maintains a buffer; sending application is blocked until room in the buffer for its write
- Holds data until acknowledged by receiver *as successfully received*
- Implements window expansion and contraction
 - Note difference between *flow* and *congestion* control
 - Dynamic adjustment to sliding window size

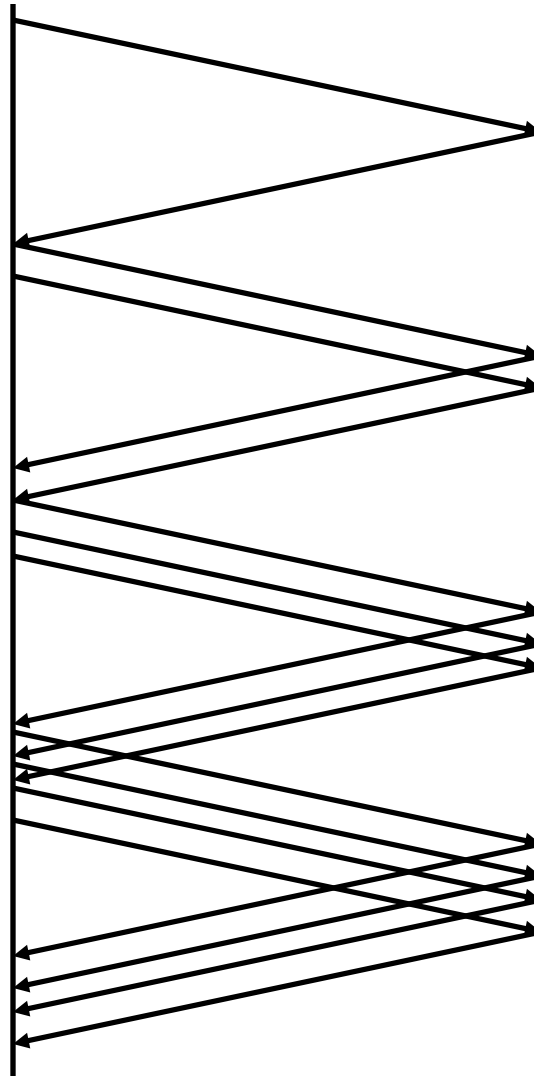
TCP Flow & Congestion Control

- Flow vs. Congestion Control
 - Flow control protects the recipient from being overwhelmed.
 - Congestion control protects the network from being overwhelmed.
- TCP Congestion Control
 - Additive Increase / Multiplicative Decrease
 - Slow Start
 - Fast Retransmit and Fast Recovery

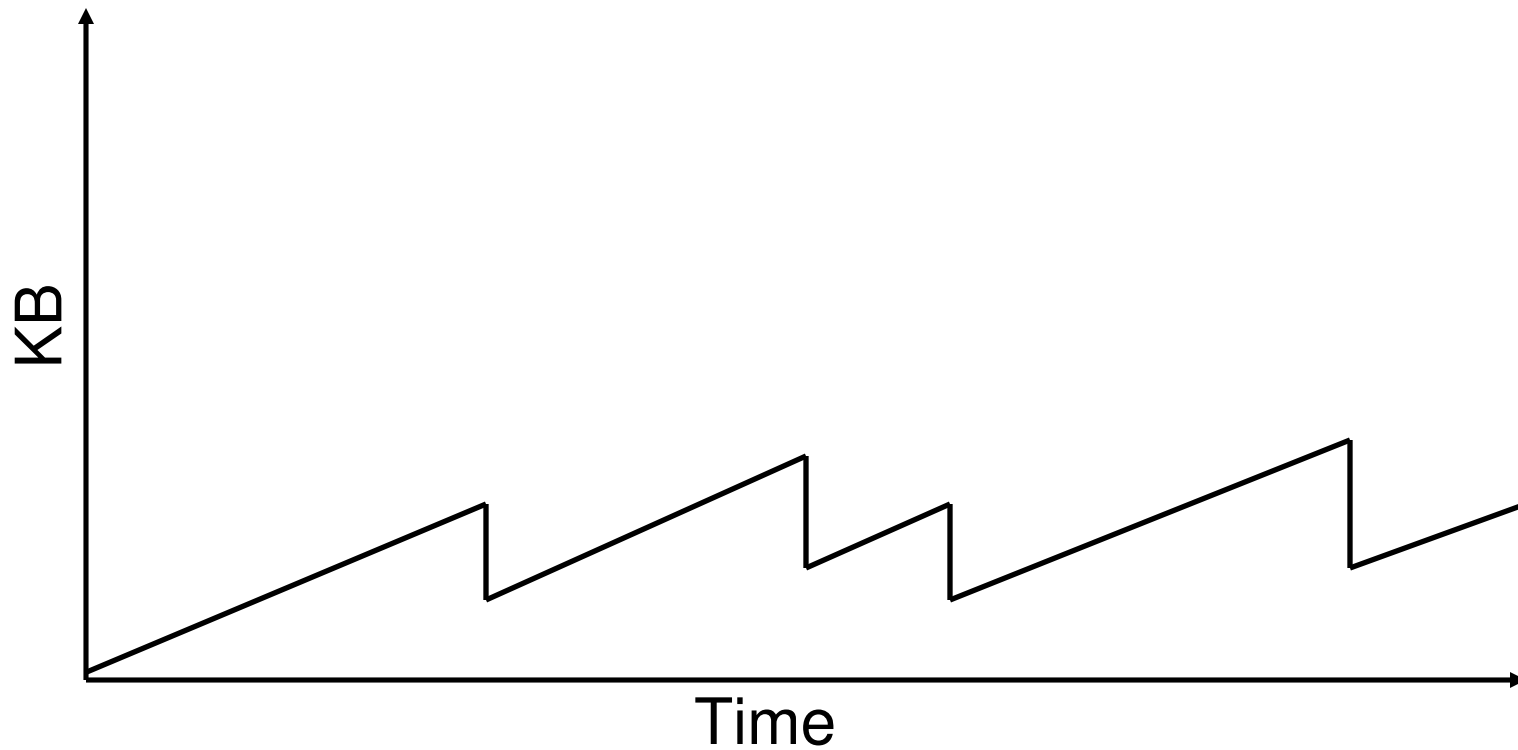
Increase and Decrease

- A value CongestionWindow is used to control the number of unacknowledged transmissions.
- This value is increased linearly until timeouts for ACKs are missed.
- When timeouts occur, CongestionWindow is decreased by half to reduce the pressure on the network quickly.
- The strategy is called “additive increase / multiplicative decrease”.

Additive Increase



TCP Sawtooth Pattern



Conclusion

- RIP
- UDP
- TCP