Routing: RIP & OSPF, UDP

8 January 2025
Lecture 9

Some Slides Credits: Steve Zdancewic (UPenn)

Topics for Today

- Routing
  - Introduction and Goals
  - RIP
  - OSPF
- UDP

- Sources in PD:
  - RIP: 3.4.2
  - OSPF: 3.4.3
  - UDP: 5.1

Routing

Vatnsdalsvegur
Forsæludalur
Hjarðartunga
Grímstunga
Grímstungu-
heiði
Sunnuhlíð
Kárdalstunga
Haukagil
Saur-
bær
Þórorms-
tunga
Guðrúnar-
staðir
Marðarnúpur
Ásbrekka
Gilá
Ás
Brúsa-
staðir
Hof
Birkihlíð
Snærings-
staðir
Undirfell
Bakki
Nautabú
Kornsá
Eyjólfs-
staðir
Kornsá II
Hvammur I-II
Gilsstaðir
Flaga
Helgavatn
Hnjúkur
Hjallaland
Breiðabóls-
staður
Másstaðir
Miðhús
Steinkot
Flóðvangur
Bjarna-
staðir
Vatnsdals-
hólar
722
1
Blönduós
Reykjavík

CHARLESTON 31
SUMMERVILLE 17
EAST
52
SOUTH
ALT
17
TO
WEST
SOUTH CAROLINA
6
MUNICIPAL COMPLEX
POLICE DEPT.
FIRE DEPT.
MUNICIPAL COURT

Routing protocols on the internet

Routing information protocol (RIP)

- Distance vector routing
- Uses Bellman-Ford algorithm
- Outdated, suffers from count-to-infinity problem

Open shortest path first (OSPF)

- Link state routing
- Runs over Layer 3 (over IP)
- Uses Dijkstra algorithm to determine shortest paths

Intermediate System to Intermediate System IS-IS

- Link state routing – similar to OSPF
- Runs over Layer 2 (under IP)
- Uses Dijkstra's algorithm

Border gateway protocol (BGP)

- Between networks (administrative domains, Autonomous Systems)
- Path-vector routing
- Consideration of business agreements

Routing criteria

Correctness

Every packet is delivered to its destination

Efficiency

Choose paths with small delay and high throughput (network wide)

Complexity

Setting up routing tables

Making routing decisions

Robustness

Cope with topology changes

No network reboot

Adaptiveness

Load balancing and traffic control

Fairness

All users get the same degree of service

Path costs, Routing metrics

Minimum hop

Number of channels traversed

Shortest path

Channels have statically assigned weights

Cost of a path is the sum of costs of the edges

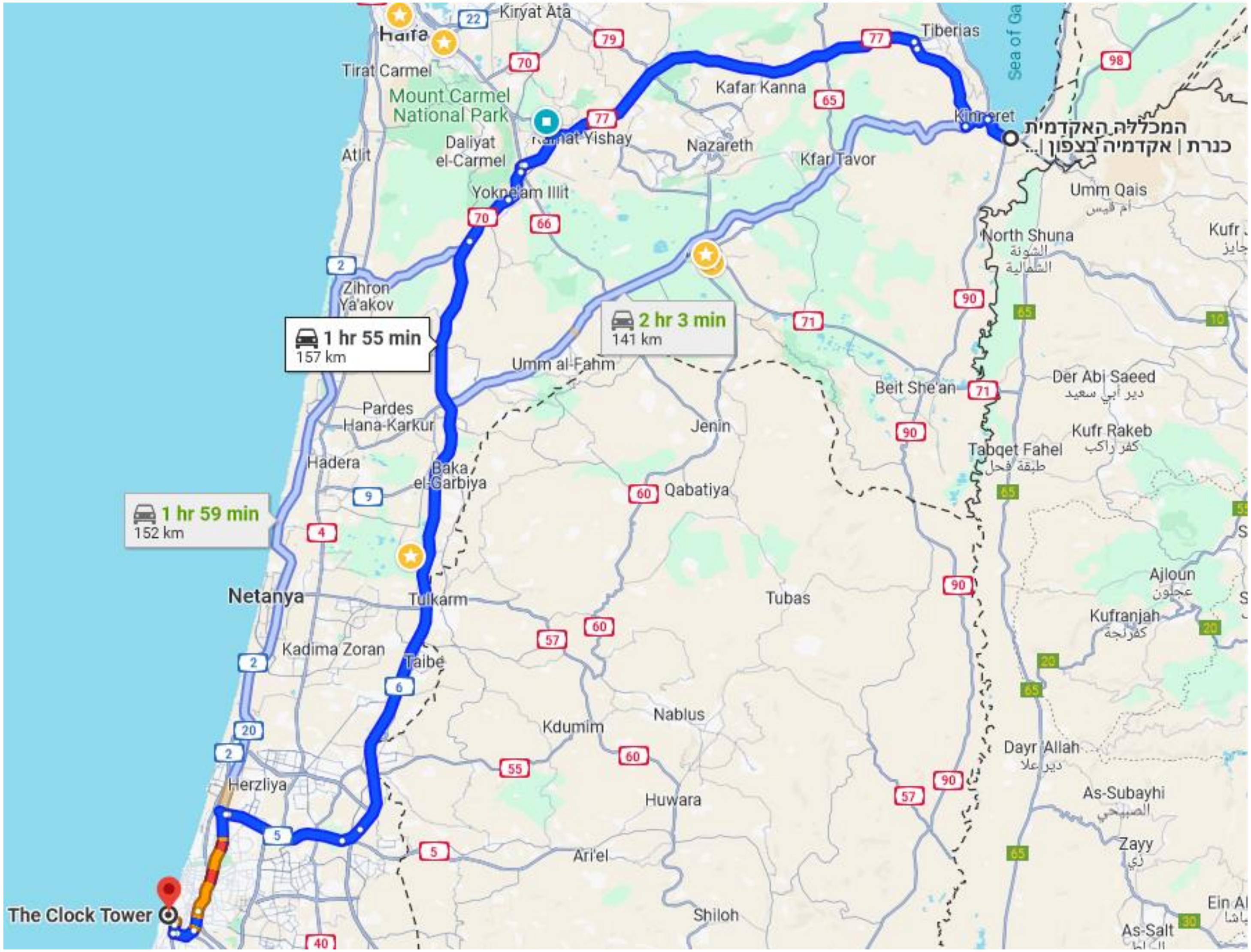
Presumes no negative-cost cycles

Minimum delay

Channels have dynamically assigned weights based on the traffic on the channel

Routing tables are repeatedly revised such that paths with close to minimal delays are chosen

Shortest vs Fastest

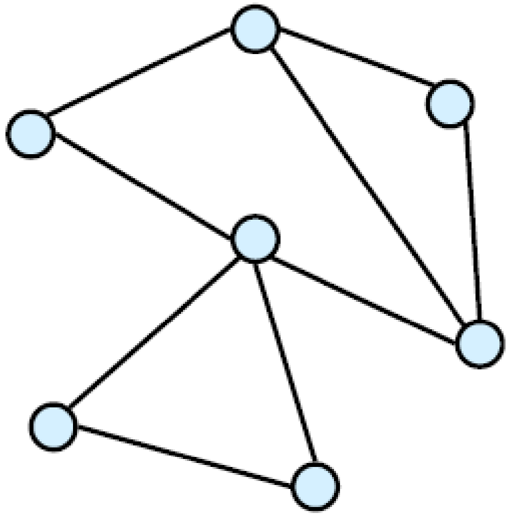


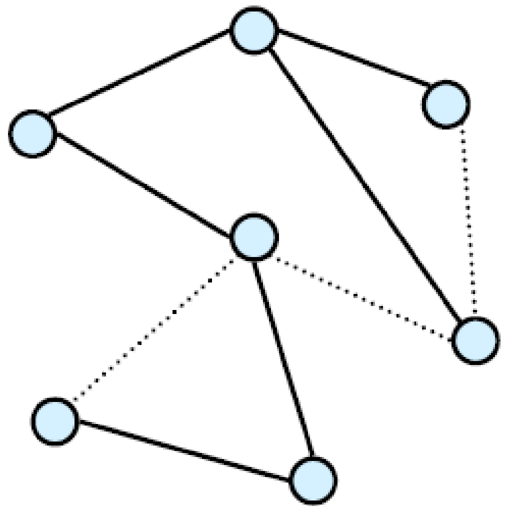
So Far

- Routing
  - Introduction and Goals
  - RIP
  - OSPF
- UDP

Spanning Trees (Abstractly)

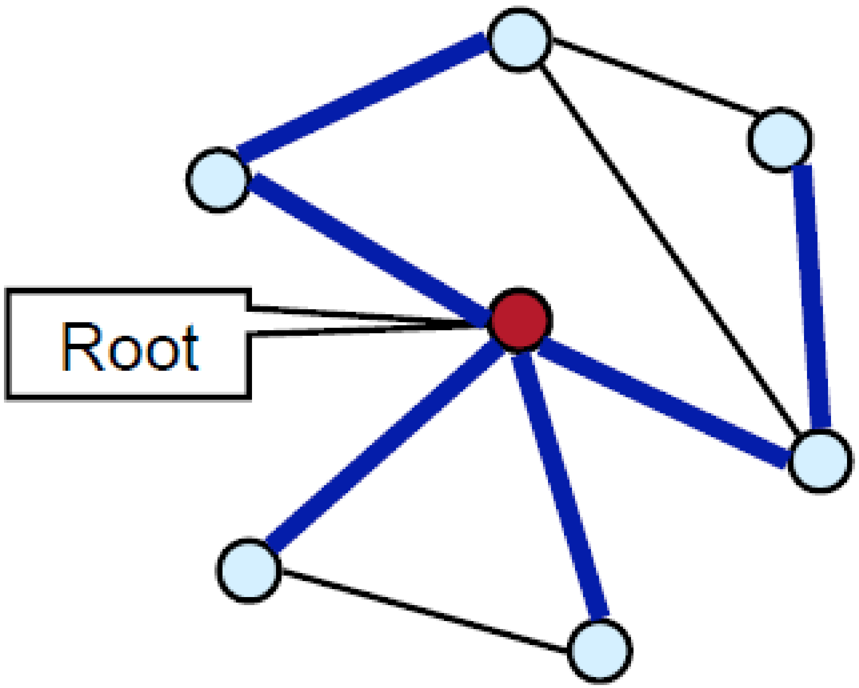
- Given a connected graph G
- A *spanning tree* is an acyclic, connected subgraph of G that contains all the nodes.





Spanning Tree Algorithm (Abstractly)

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

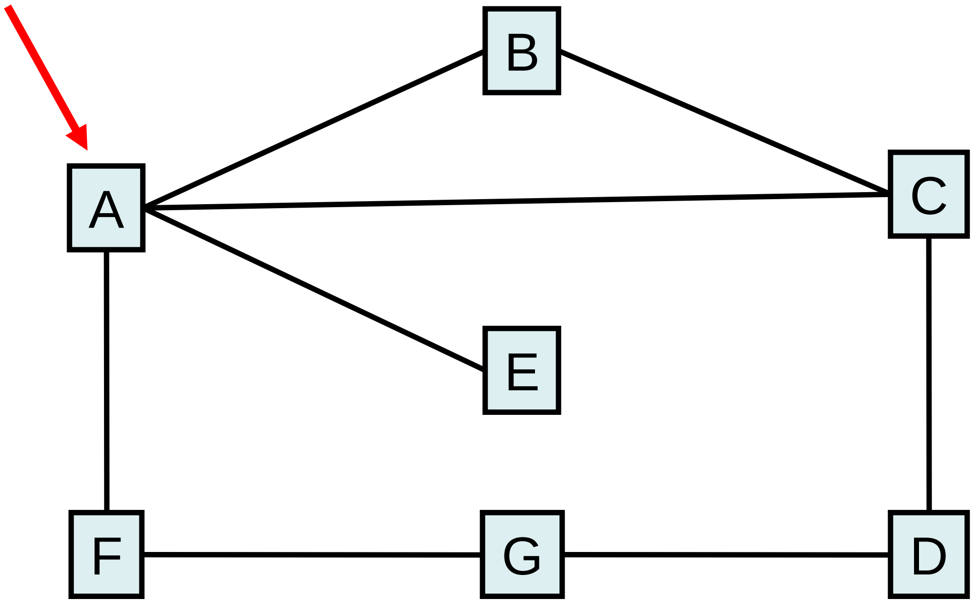


Distance Vector Algorithm (RIP)

- 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 update distance vectors and routing information accordingly.

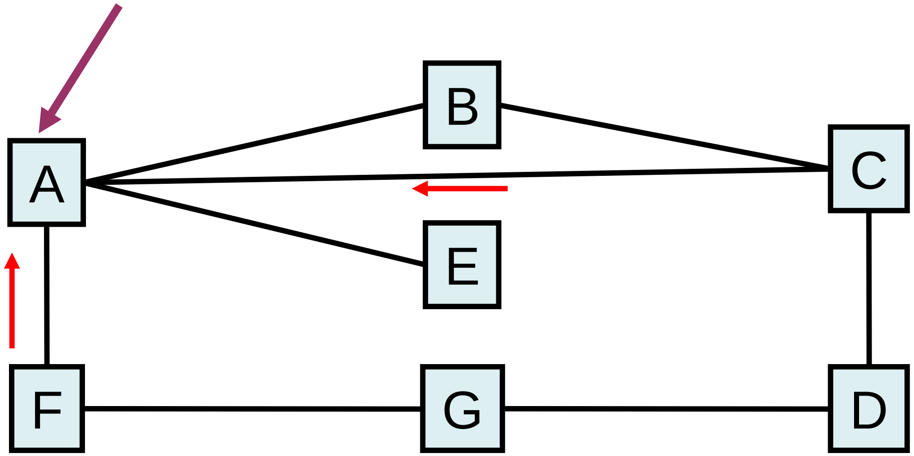
- Ignore worse information
- Update any better routes

If host changed its tables, send new DV to neighbors

After a few iterations, routing information *converges*

Example Iteration Steps

A	1
B	∞
C	∞
D	∞
E	∞
F	0
G	1



A	1
B	1
C	0
D	1
E	∞
F	∞
G	∞

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
B	1	B
C	1	C
D	∞	-
E	1	E
F	1	F
G	∞	-

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

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

Details

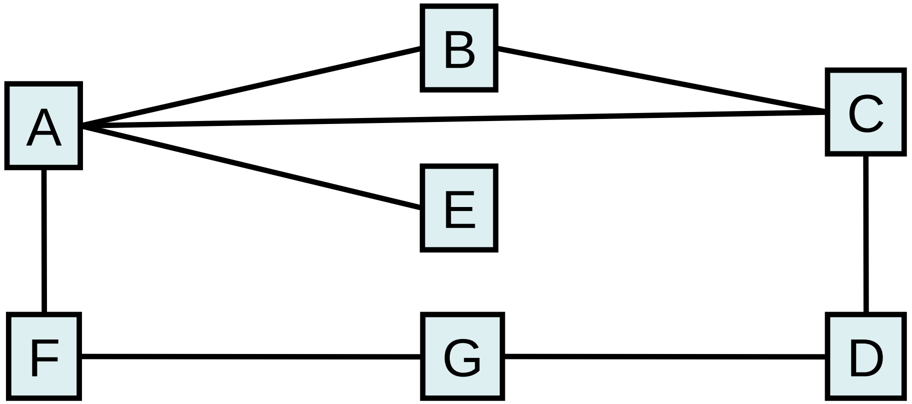
- Note: No single host has all routing information.
- When to send update vectors?
  - When your routing table changes (triggered)
  - Periodically (“I’m alive!”)

- Detecting link/node failure
  - (1) Periodically exchange “I’m alive!” messages.
  - (2) Timeout mechanism

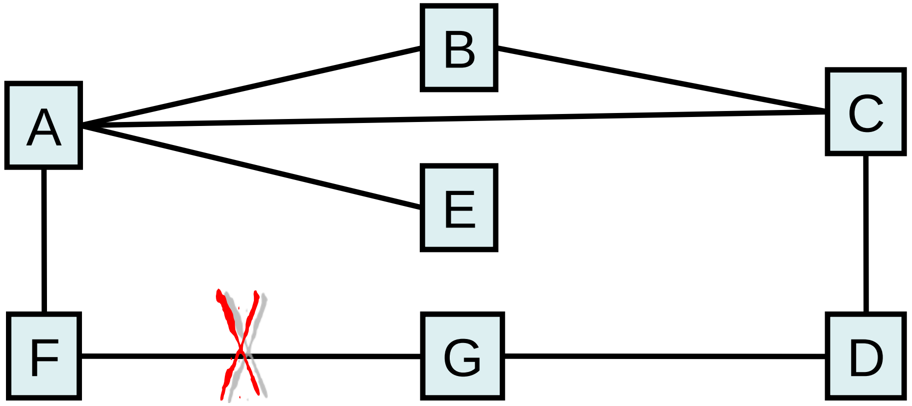
- In a static network, if all weights are 1, once A discovers a path to B (cost $< \infty$), no subsequent round will ever discover a better path to B
  - All weights are 1 is called “Hop Count” (Default RIP action)

Dynamic Networks

What about a dynamic network?

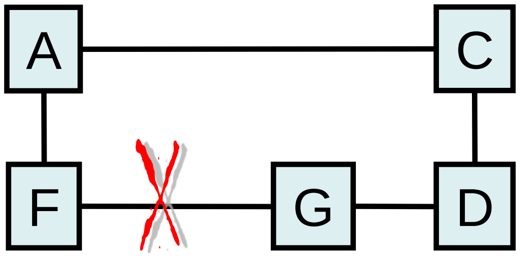


- A nodes enters
  - The new neighbors will eventually inform everyone else (easy)
- A node/link fails
  - Is the network partitioned?
  - How do nodes discover that a node/link is gone?



Link Failure Example 1

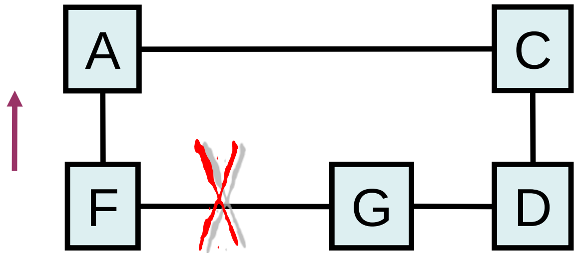
A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	2	F	G	2	D	G	1	G	G	1	G



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	∞	-



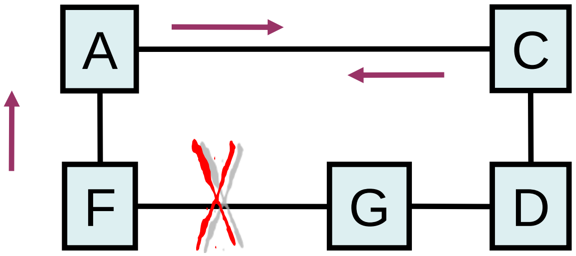
Case 1:

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	∞	-



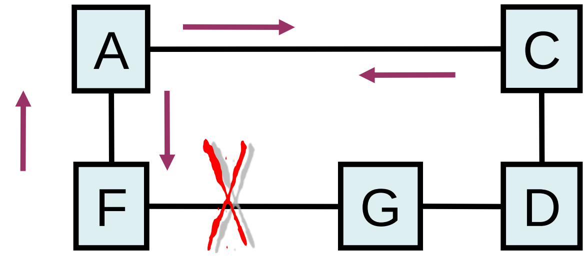
A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	3	C	G	2	D	G	1	G	G	∞	-

Case 1:

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

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	∞	-



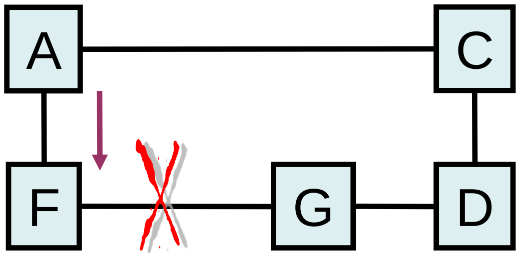
A			C			D			F		
Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop	Dest	C	Next hop
A	-	-	A	1	A	A	2	C	A	1	A
C	1	C	C	-	-	C	1	C	C	2	A
D	2	C	D	1	D	D	-	-	D	2	G
F	1	F	F	2	A	F	2	G	F	-	-
G	3	C	G	2	D	G	1	G	G	4	A

Case 1:

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

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	∞	-



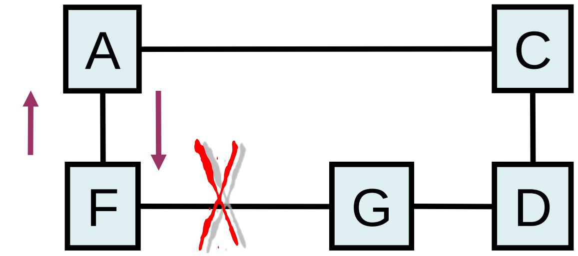
Case 2:

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:

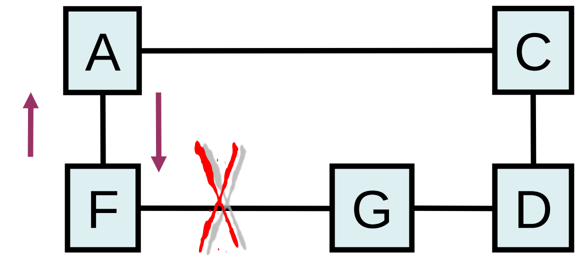
1. A sends to F
2. F sends to A

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

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

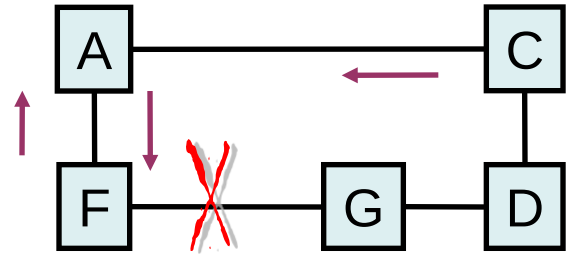


Case 2:

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

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



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

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

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)

Three RIP Solutions

Option 1: Infinity is small

- Don’t let the distance fields go above a predefined maximum
  - Say 15, 20, etc. (15 is the actual value)
  - The max must be greater than the diameter of the network

Option 2: Don’t send routes to the one who sent it to you

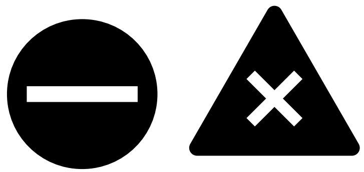
Example: F receives an update from A about a path to G.

- In the example before, F didn’t know that’s A’s “path” went via F
- A doesn’t send F an advertisement to get to G since it goes via F
- Called “split horizon”
- “Split horizon with poison reverse”- A sends route for G to F, but with infinite distance

Three RIP Solutions

Option 3: Route Poisoning and Holddowns

- Instead of just stopping to announce a lost destination, a router sends a poison message for the destination
  - E.g. the router announces distance 16 (≡ ∞) for the destination
- For a fixed period of time (say 3 minutes) any new offers for the destination via the sender will be rejected
  - E.g. Wait until everyone has heard the poison and then accept new offers
- Gives a chance for the route to be deleted by everyone.
- Cisco specific



RIP v1 (1986) Details

Classful routing only

- No subnet masks
- Assumes everyone has same mask

Supports multiple address families

Commands

- Request: Ask for someone to send their info
- Response: Respond with routing table

Updates sent

- Timer about 30 seconds between updates
- Due to changes

Split Horizon, Poison Reverse, Small Infinity

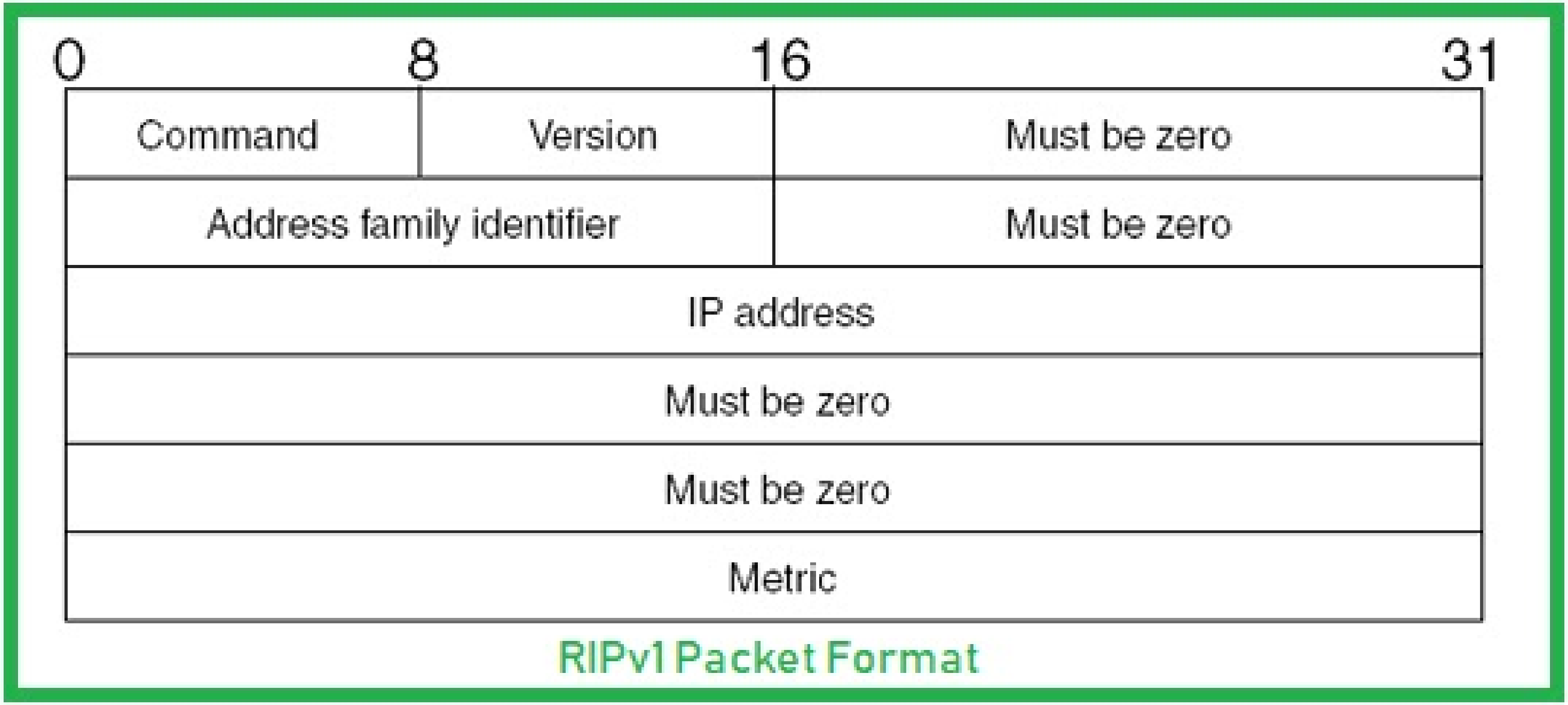
Takes shortest path

- If two are shortest, splits traffic on them equally

Route deletion

- If no update in 180 seconds or reaches infinity
- Marked as deleted
- Lives 120 seconds longer to send to others

RIP v1 (1986) Details



RIPv1 Packet Format

https://www.networkurge.com/2020/05/ripv1.html

RIP v2 (1998) Details

Added authentication

- Simple password
- Address family ID 0xFFFF

Route tag for exterior routing

```
 0                   1                   2                   3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Command (1)   | Version (1)   |            unused             |
+---------------+---------------+-------------------------------+
| Address Family Identifier (2) |        Route Tag (2)          |
+-------------------------------+-------------------------------+
|                         IP Address (4)                        |
+---------------------------------------------------------------+
|                         Subnet Mask (4)                       |
+---------------------------------------------------------------+
|                         Next Hop (4)                          |
+---------------------------------------------------------------+
|                         Metric (4)                            |
+---------------------------------------------------------------+
```

Next hop

- 0.0.0.0 for the sender
- Another address must be on the subnet, allows just some routers to run RIP

Message sent via Multicast 224.0.0.9

```
 0                   1                   2                   3 3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Command (1)   | Version (1)   |            unused             |
+---------------+---------------+-------------------------------+
|             0xFFFF            |    Authentication Type (2)    |
+-------------------------------+-------------------------------+
~                       Authentication (16)                     ~
+---------------------------------------------------------------+
```

So Far

- Routing
  - Introduction and Goals
  - RIP
  - OSPF
- UDP

Open Shortest Path First (OSPF)

OSPFv2 (1998)

- RFC 2328
- IPv4 ✓
- In protocol authentication of routers and completeness of messages
- Atoms are networks and subnets
- Routers identified by Router ID and/or IP address

OSPFv3 (2008)

- RFC 5340
- IPv4 + IPv6 ✓
- Authentication at IP layer (IPSec)
- Completeness at IP layer
- Atoms are links (may have multiple subnets)
- Router IDs and IP addresses are disjoint

Open Shortest Path First (OSPF)

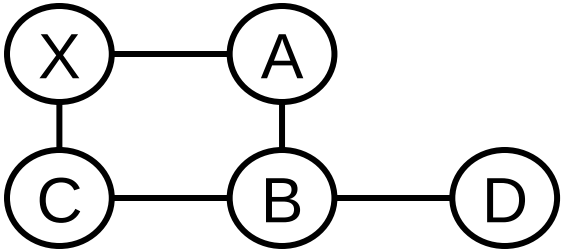
- Each node sends a *reliable flood* of information to all other nodes
  - In v3, can limit the flooding scope
- Link-State Packets (LSPs) contain

ID of the node that created the packet	List of (neighbor, cost) pairs associated with the source node	Sequence Number (64bits—no wrapping)
Time To Live (ensure old info is eventually removed)	Authentication (v2) – password or message authenticity code (MAC)	

OSPF Steps

Reliable Flooding

- Adjacent routers reliably send updates (ACKs)
- Source sends to all neighbors
- Recipients
  - Send to all neighbors except the one it got the message from
  - Ignores duplicates



Local Calculations

1. Once all of the link-state info has been flooded each node has complete network topology
2. Compute routing information using Dijkstra's shortest-path algorithm
3. Periodic updates and failure detection are like RIP.

OSPF Features

Authentication of routing messages (v2)

- Misconfigured or malicious host could advertise bad route info (i.e. reach anywhere in 0 hops)
- Prevent routers from accidentally joining a network
- Prevent malicious or accidental changes to route information
- (Eventually added to RIP too.)

Additional Hierarchy

- Partitions domains into *areas*
- Reduces transmission & storage overhead

Load Balancing

- Multiple routes with same cost
- Traffic evenly distributed

Dijkstra's Algorithm

Node has two lists – Confirmed and Tentative - pairs of (Destination, Cost, Next-Hop)

Algorithm:

1. Initialize Confirmed with an entry for self with cost 0.

2. For the node just added to Confirmed in the previous step, ($Next$), examine its Link State Packet (LPS).

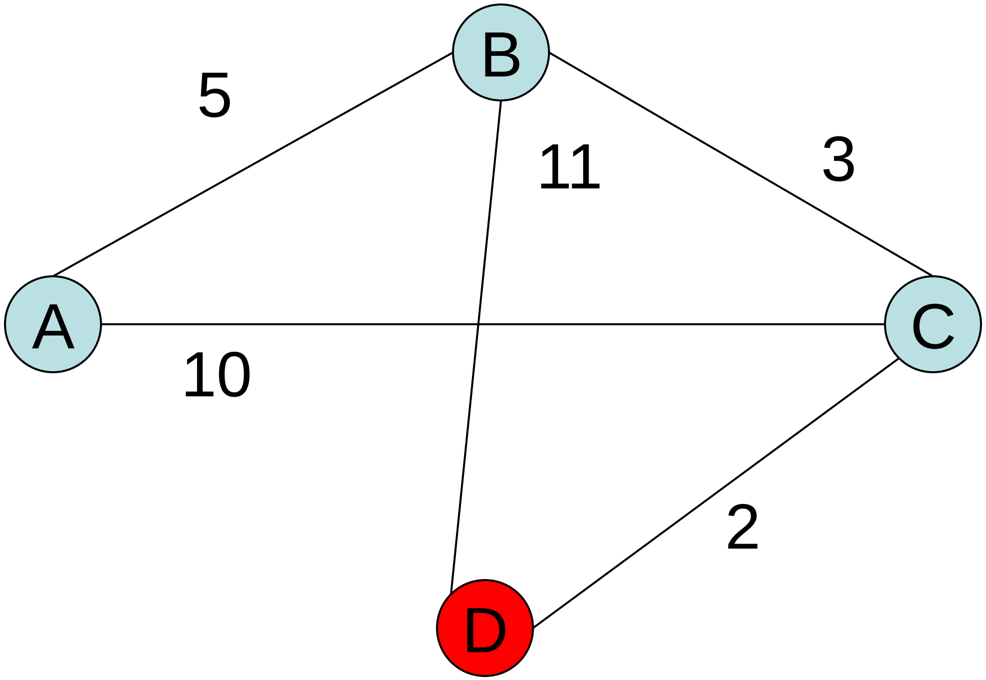
3. For each $Neigh$ of $Next$, calculate the distance to $Neigh$ via $Next$ $cost = (self \to Next) + (Next \to Neigh)$
   a) If $Neigh$ isn't on Confirmed or Tentative, add $(Neigh, cost, NH)$ to Tentative, where NH is the way to reach $Next$ in Confirmed.

   a) If $Neigh$ is on Tentative and $cost$ is better than the old cost, replace it with $(Neigh, cost, NH)$, where NH is the way to reach $Next$ in Confirmed.

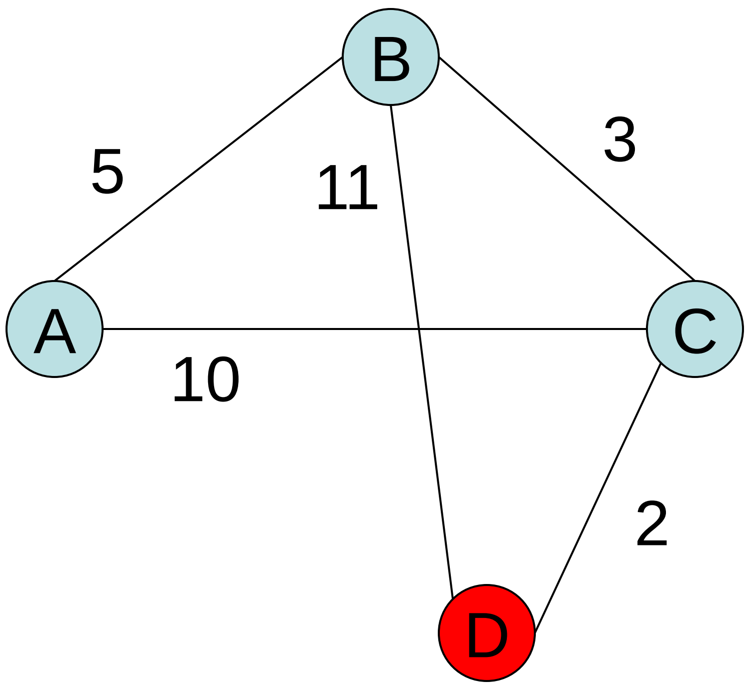
4. If Tentative is empty, **stop**.

5. Choose the node on Tentative with the lowest cost, move it to Confirmed, and go to step 2.

Dijkstra Example

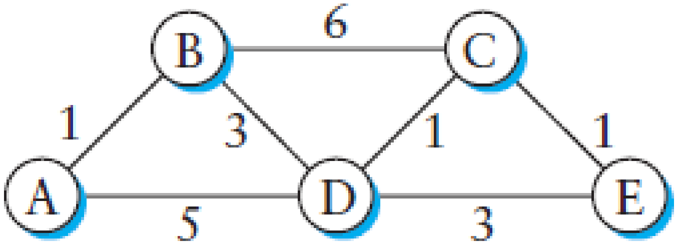


Dijkstra Example



Step	Confirmed	Tentative	Comments
1	(D,0,-)		Since D is the only new member of the confirmed list, look at its LSP.
2	(D,0,-)	(B,11,B) (C,2,C)	D's LSP says we can reach B through B at cost 11, which is better than anything else on either list, so put it on Tentative list; same for C.
3	(D,0,-) (C,2,C)	(B,11,B)	Put lowest-cost member of Tentative (C) onto Confirmed list. Next, examine LSP of newly confirmed member (C).
4	(D,0,-) (C,2,C)	(B,5,C) (A,12,C)	Cost to reach B through C is 5, so replace (B,11,B). C's LSP tells us that we can reach A at cost 12.
5	(D,0,-) (C,2,C) (B,5,C)	(A,12,C)	Move lowest-cost member of Tentative (B) to Confirmed, then look at its LSP.
6	(D,0,-) (C,2,C) (B,5,C)	(A,10,C)	Since we can reach A at cost 5 through B, replace the Tentative entry.
7	(D,0,-) (C,2,C) (B,5,C) (A,10,C)		Move lowest-cost member of Tentative (A) to Confirmed, and we are all done.

Dijkstra Example 2

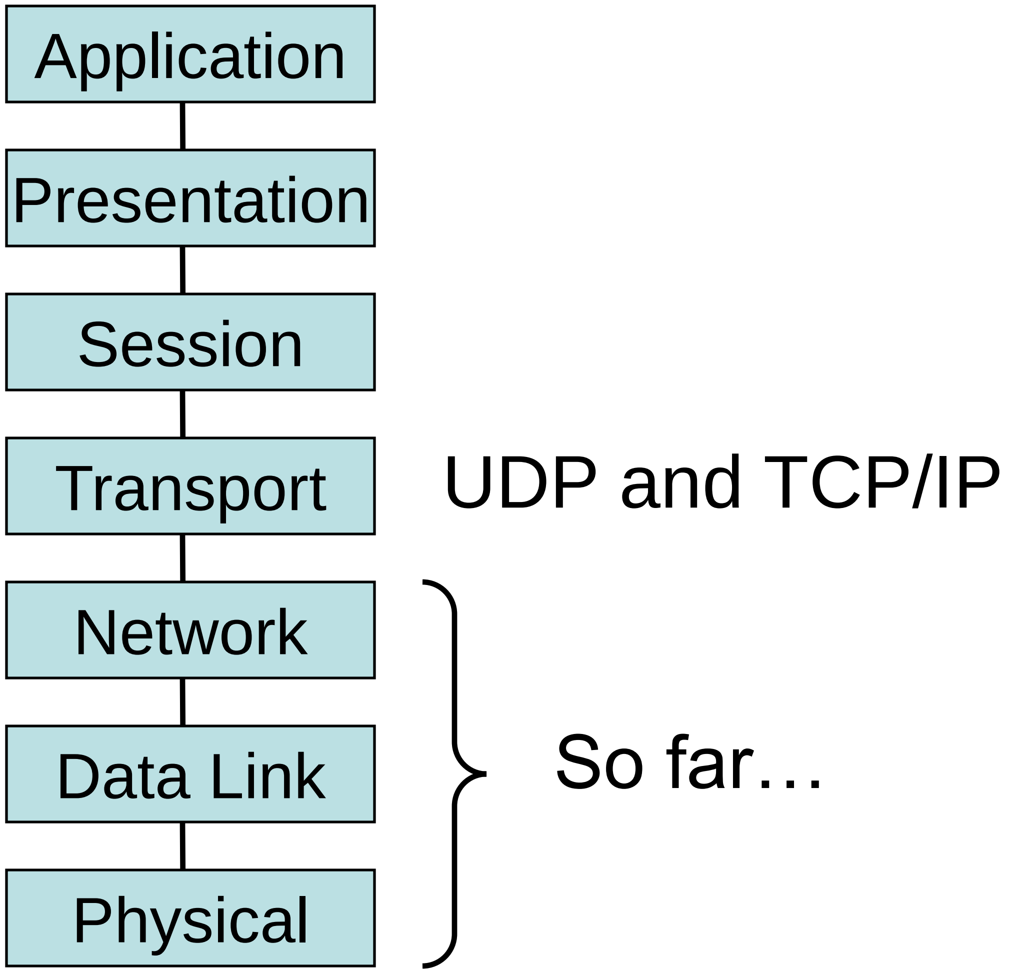


Step	Confirmed	Tentative
1	(A,0,-)	
2	(A,0,-)	(B,1,B) (D,5,D)
3	(A,0,-) (B,1,B)	(D,4,B) (C,7,B)
4	(A,0,-) (B,1,B) (D,4,B)	(C,5,B) (E,7,B)
5	(A,0,-) (B,1,B) (D,4,B) (C,5,B)	(E,6,B)
6	(A,0,-) (B,1,B) (D,4,B) (C,5,B) (E,6,B)	

So Far

- Routing
  - Introduction and Goals
  - RIP
  - OSPF
- UDP

Protocol Stack Revisited



Application vs. Network

Application Needs	Network Characteristics
Reliable, Ordered, Single-Copy Message Delivery	Drops, Duplicates and Reorders Messages
Arbitrarily large messages	Finite message size (MTU)
Timely receipt of messages	Arbitrary delay
Supports multiple applications per host	Host-level addressing and delivery mechanisms (no ability to discern conversations)
Flow control by Receiver	No end-to-end flow control mechanisms

User Datagram Protocol (UDP)

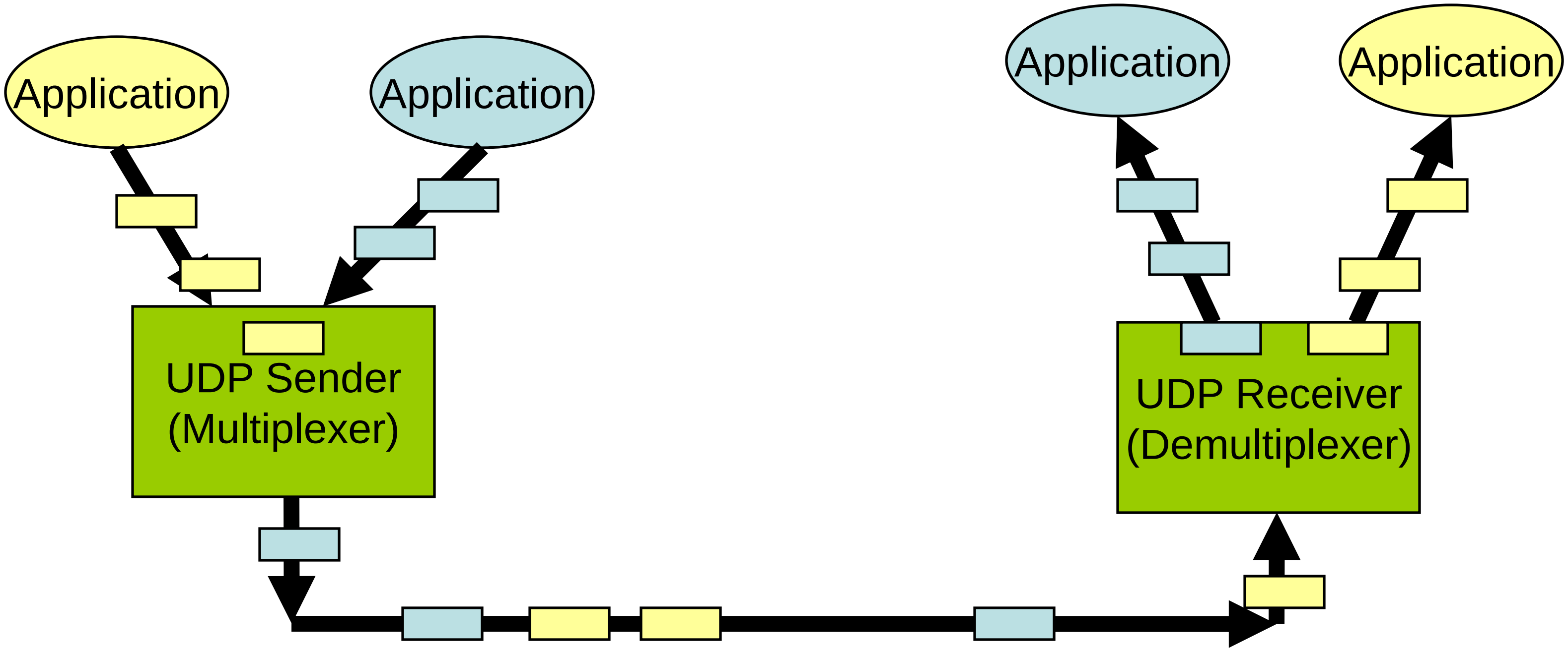
0	16	31
SrcPort	DestPort	
Length	Checksum	
IP Packet Data		

- 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
  - HTTP: port 80
  - Unix talk : port 517
  - In unix, ports are listed in /etc/services
  - In Windows: C:\Windows\System32\drivers\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

Conclusion

- Routing
  - Introduction and Goals
  - RIP
  - OSPF
- UDP