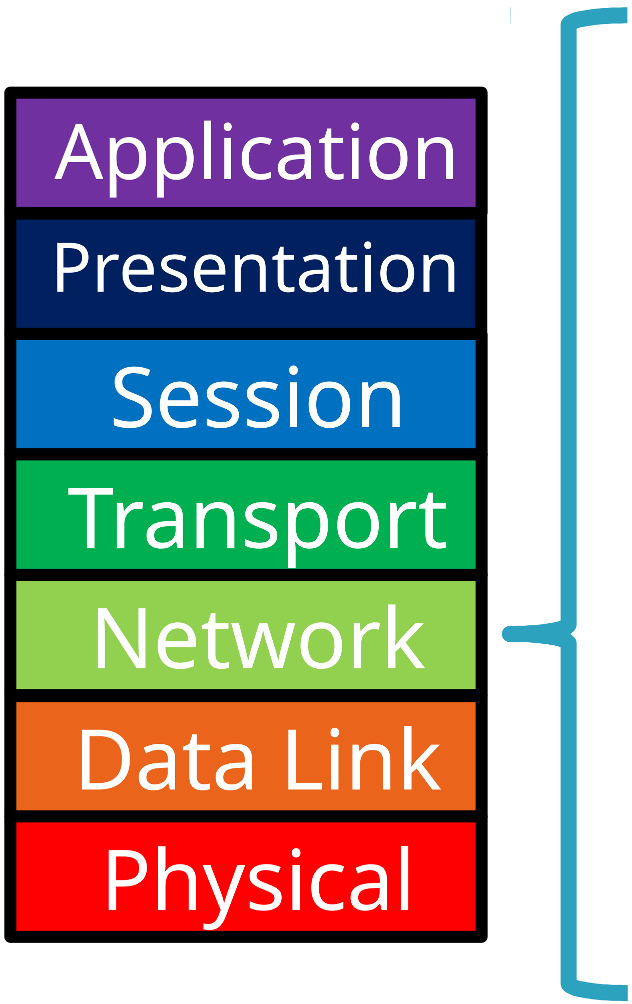
Computer Networks

Lecture 7: Network Layer – Part II

Based on slides from D. Choffnes Northeastern U. and P. Gill from StonyBrook University
Revised Autumn 2015 by S. Laki

Network Layer



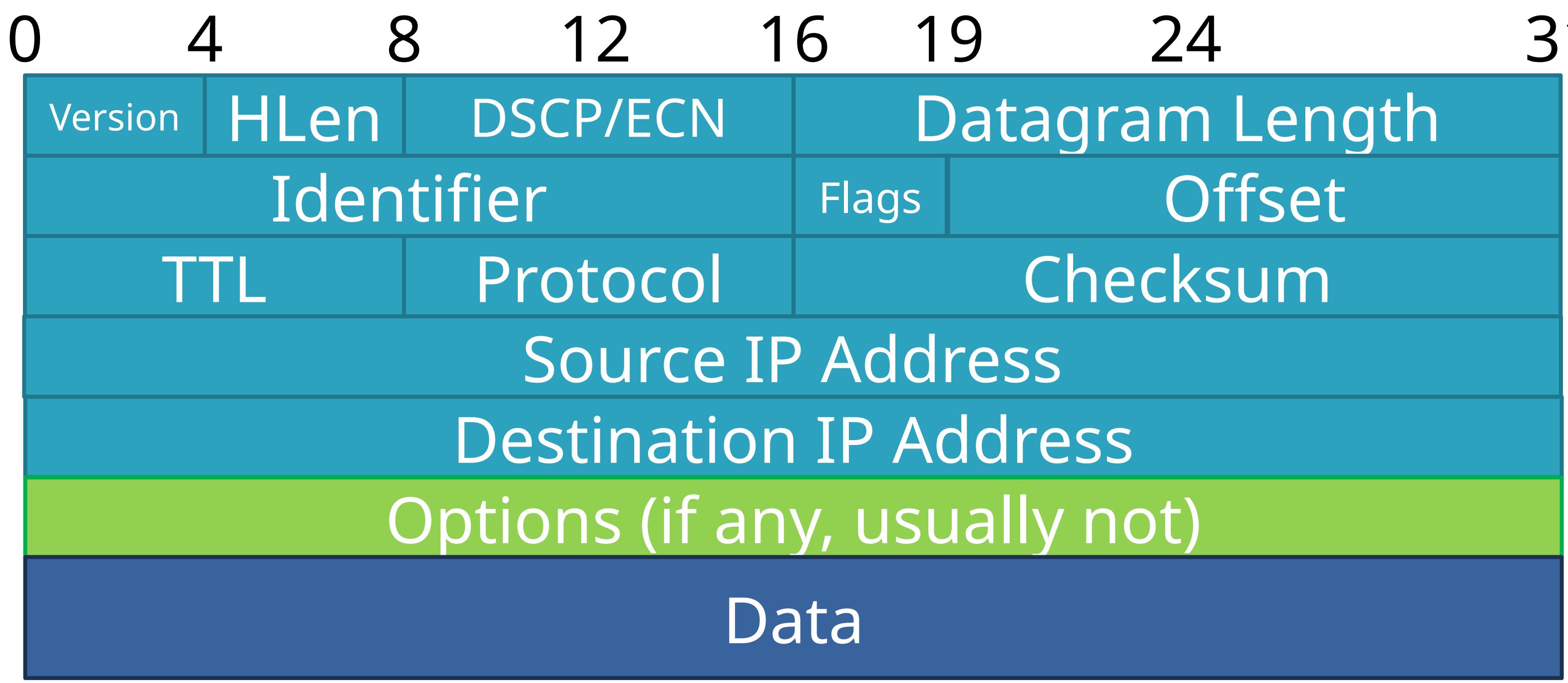
- Function:
  - Route packets end-to-end on a network, through multiple hops
- Key challenge:
  - How to represent addresses
  - How to route packets
    - Scalability
    - Convergence

Outline

- Addressing
  - Class-based
  - CIDR
- IPv4 Protocol Details
  - Packed Header
  - Fragmentation
- IPv6

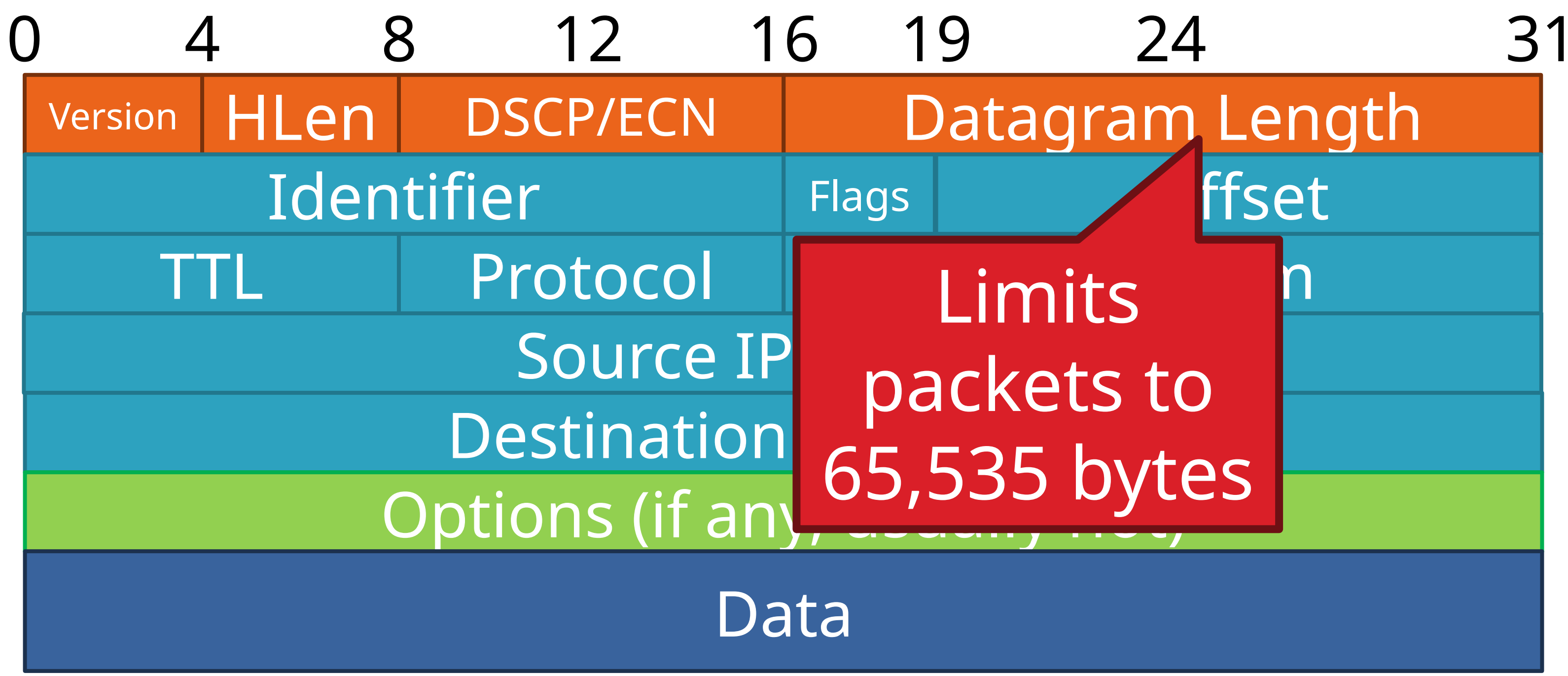
IP Datagrams

- IP Datagrams are like a letter
  - Totally self-contained
  - Include all necessary addressing information
  - No advanced setup of connections or circuits



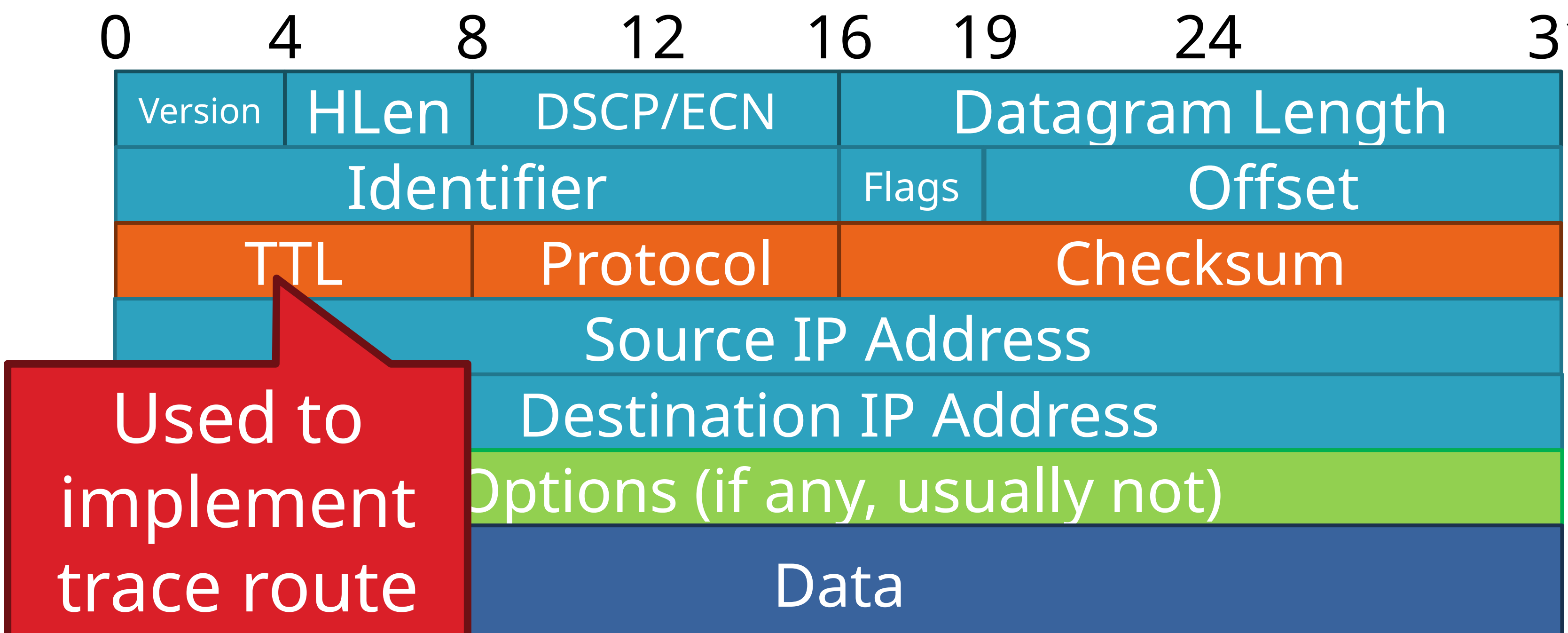
IP Header Fields: Word 1

- Version: 4 for IPv4
- Header Length: Number of 32-bit words (usually 5)
- Type of Service: Priority information (unused)
- Datagram Length: Length of header + data in bytes



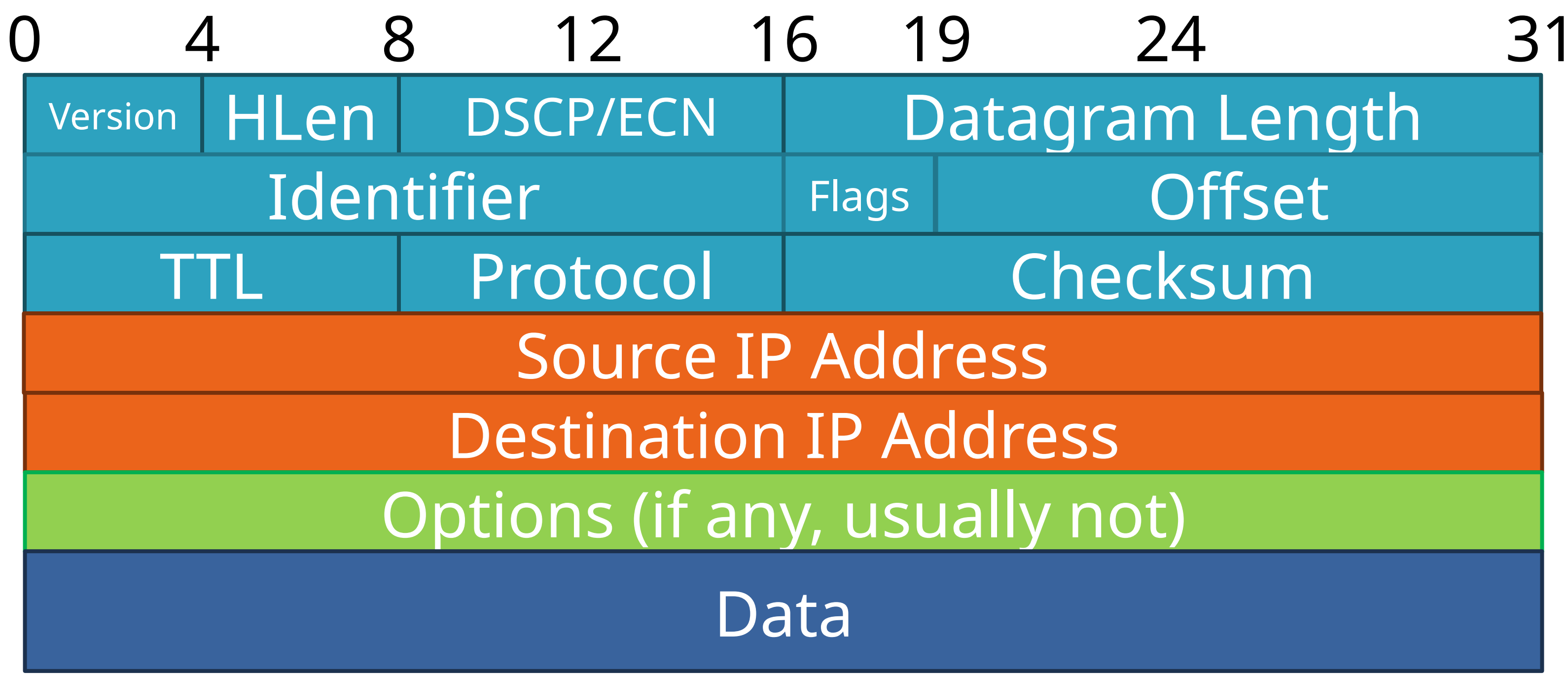
IP Header Fields: Word 3

- Time to Live: decremented by each router
  - Used to kill looping packets
- Protocol: ID of encapsulated protocol
  - 6 = TCP, 17 = UDP
- Checksum

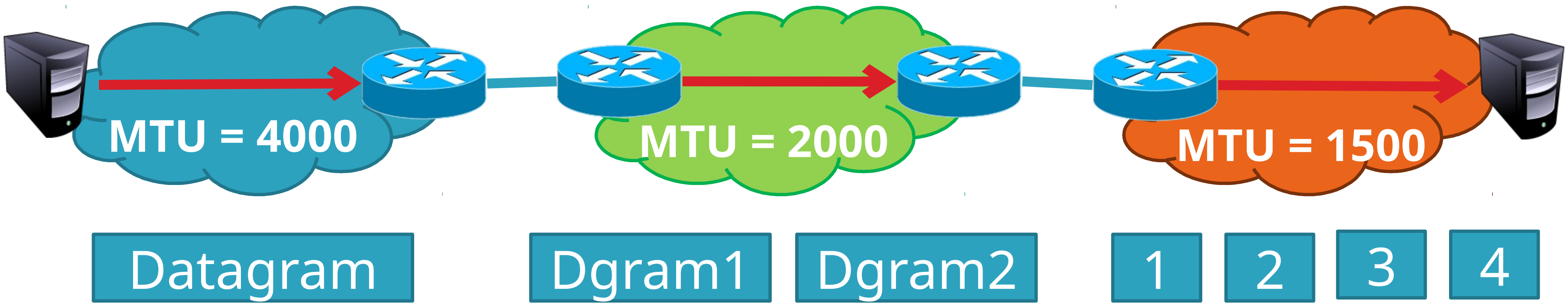


IP Header Fields: Word 4 and 5

- Source and destination address
  - In theory, must be globally unique
  - In practice, this is often violated



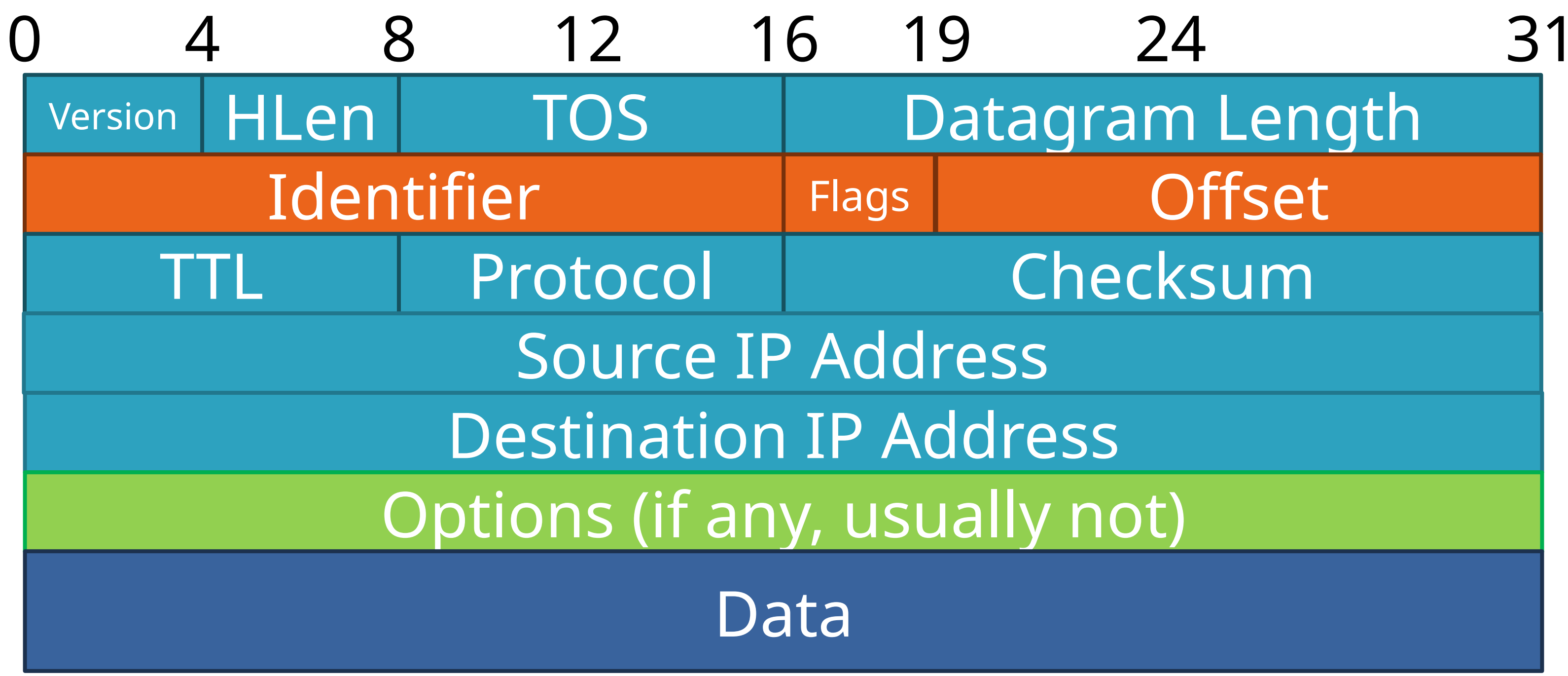
Problem: Fragmentation



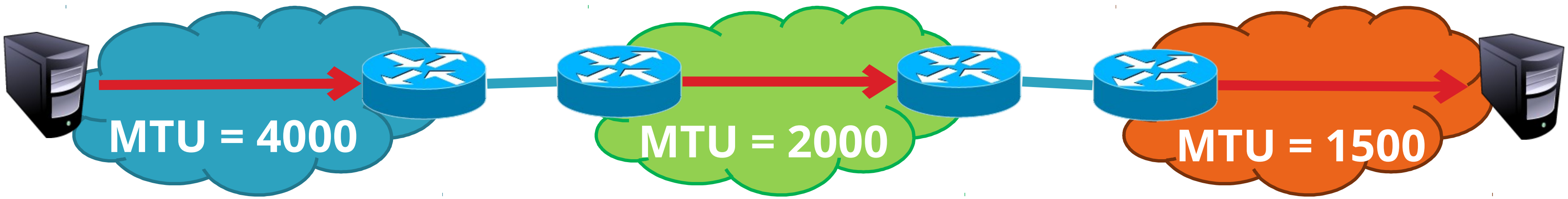
- Problem: each network has its own MTU
  - DARPA principles: networks allowed to be heterogeneous
  - Minimum MTU may not be known for a given path
- IP Solution: fragmentation
  - Split datagrams into pieces when MTU is reduced
  - Reassemble original datagram at the receiver

IP Header Fields: Word 2

- Identifier: a unique number for the original datagram
- Flags: M flag, i.e. this is the last fragment
- Offset: byte position of the first byte in the fragment
  - Divided by 8



Fragmentation Example



Length = 2000, M = 1
Offset = 0

IP	Data
20	1980

Length = 3820, M = 0

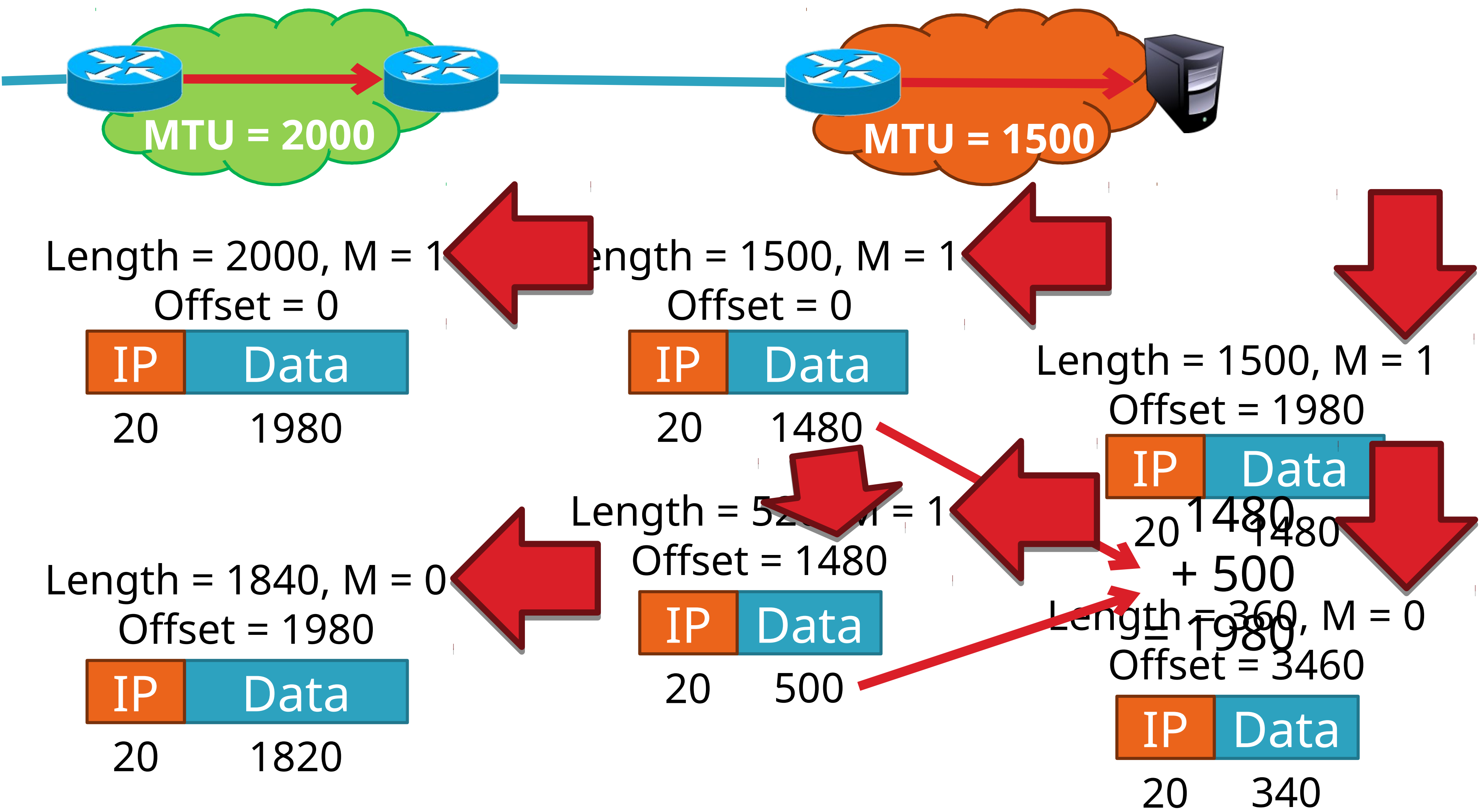
IP Hdr	Data
20	3800

Length = 184 M = 0
Offset = 1980

IP	Data
20	1820

1980
+ 1820
= 3800

Fragmentation Example



IP Fragment Reassembly

Length = 1500, M = 1, Offset = 0

IP	Data
20	1480

Length = 520, M = 1, Offset = 1480

IP	Data
20	500

Length = 1500, M = 1, Offset = 1980

IP	Data
20	1480

Length = 360, M = 0, Offset = 3460

IP	Data
20	340

- Performed at destination
- M = 0 fragment gives us total data size
  - 360 – 20 + 3460 = 3800
- Challenges:
  - Out-of-order fragments
  - Duplicate fragments
  - Missing fragments
- Basically, memory management

Fragmentation Concepts

- Highlights many key Internet characteristics
  - Decentralized and heterogeneous
    - Each network may choose its own MTU
  - Connectionless datagram protocol
    - Each fragment contains full routing information
    - Fragments can travel independently, on different paths
  - Best effort network
    - Routers/receiver may silently drop fragments
    - No requirement to alert the sender
  - Most work is done at the endpoints
    - i.e. reassembly

Fragmentation in Reality

- Fragmentation is expensive
  - Memory and CPU overhead for datagram reconstruction
  - Want to avoid fragmentation if possible
- MTU discovery protocol
  - Send a packet with “don’t fragment” bit set
  - Keep decreasing message length until one arrives
  - May get “can’t fragment” error from a router, which will explicitly state the supported MTU
- Router handling of fragments
  - Fast, specialized hardware handles the common case
  - Dedicated, general purpose CPU just for handling fragments

Outline

- Addressing
  - Class-based
  - CIDR
- IPv4 Protocol Details
  - Packed Header
  - Fragmentation
- IPv6

The IPv4 Address Space Crisis

- Problem: the IPv4 address space is too small
  - 2^{32} = 4,294,967,296 possible addresses
  - Less than one IP per person
- Parts of the world have already run out of addresses
  - IANA assigned the last /8 block of addresses in 2011

Region	Regional Internet Registry (RIR)	Exhaustion Date
Asia/Pacific	APNIC	April 19, 2011
Europe/Middle East	RIPE	September 14, 2012
North America	ARIN	13 Jan 2015 (Projected)
South America	LACNIC	13 Jan 2015 (Projected)
Africa	AFRINIC	17 Jan 2022(Projected)

IPv6

- IPv6, first introduced in 1998(!)
  - 128-bit addresses
  - $4.8 * 10^{28}$ addresses per person
- Address format
  - 8 groups of 16-bit values, separated by ':'
  - Leading zeroes in each group may be omitted
  - Groups of zeroes can be omitted using '::'

2001:0db8:<u>0000:0000:0000</u>:ff00:<u>004</u>2:8329

2001:0db8:<u>0:0:0</u>:ff00:42:8329

2001:0db8::ff00:42:8329

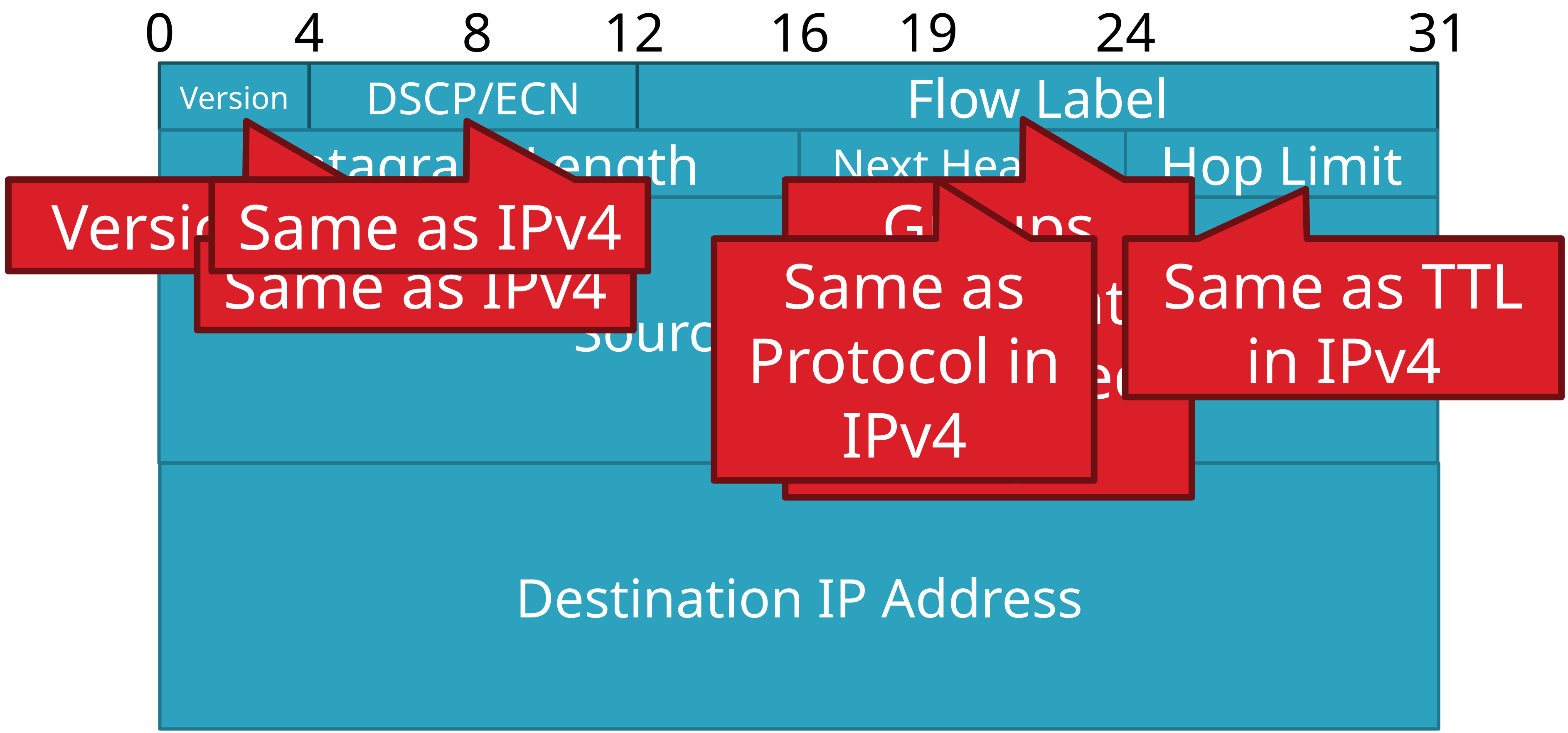
IPv6 Trivia

- Who knows the IP for localhost?
  - 127.0.0.1

- What is localhost in IPv6?
  - ::1

IPv6 Header

- Double the size of IPv4 (320 bits vs. 160 bits)



Differences from IPv4 Header

- Several header fields are missing in IPv6
  - Header length – rolled into Next Header field
  - Checksum – was useless, so why keep it
  - Identifier, Flags, Offset
    - IPv6 routers do not support fragmentation
    - Hosts are expected to use path MTU discovery
- Reflects changing Internet priorities
  - Today's networks are more homogeneous
  - Instead, routing cost and complexity dominate

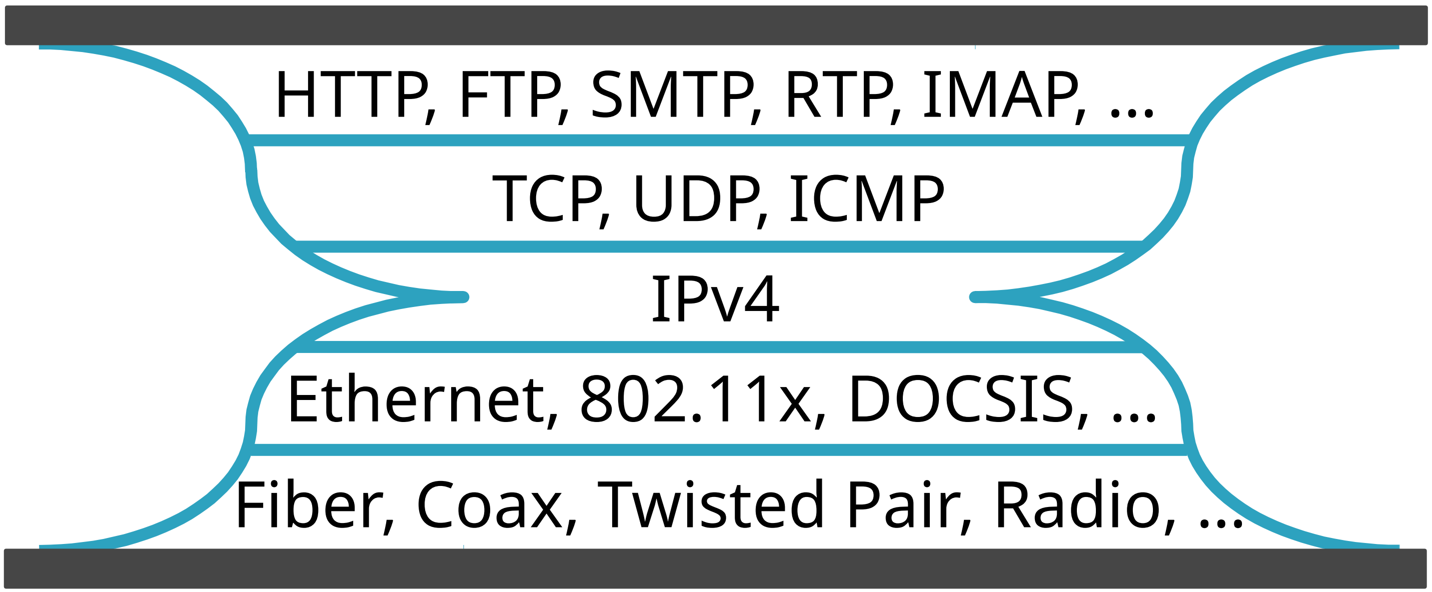
Performance Improvements

- No checksums to verify
- No need for routers to handle fragmentation
- Simplified routing table design
  - Address space is huge
  - No need for CIDR (but need for aggregation)
  - Standard subnet size is 2^{64} addresses
- Simplified auto-configuration
  - Neighbor Discovery Protocol
  - Used by hosts to determine network ID
  - Host ID can be random!

Additional IPv6 Features

- Source Routing
  - Host specifies the route to wants packet to take
- Mobile IP
  - Hosts can take their IP with them to other networks
  - Use source routing to direct packets
- Privacy Extensions
  - Randomly generate host identifiers
  - Make it difficult to associate one IP to a host
- Jumbograms
  - Support for 4Gb datagrams

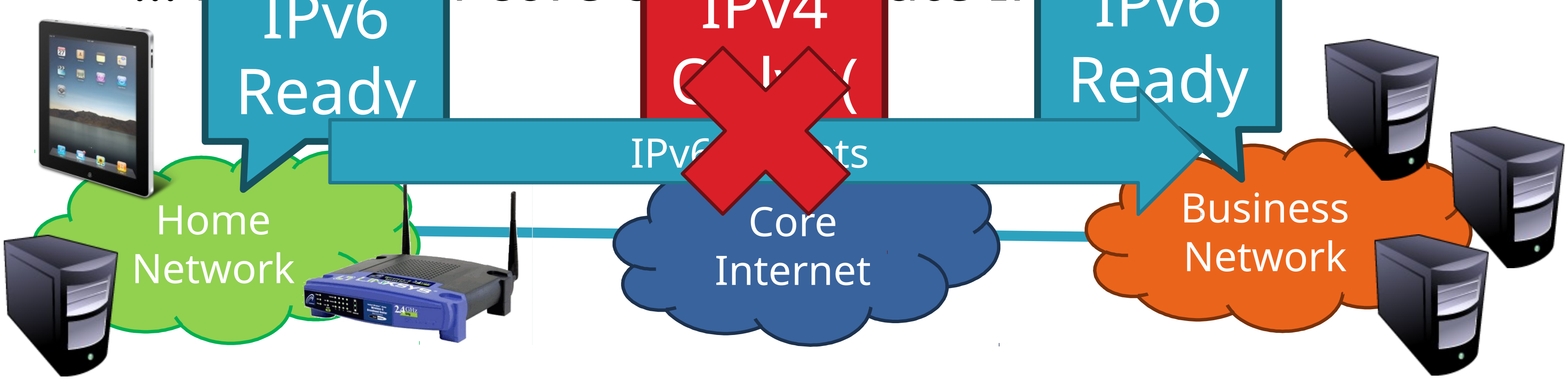
Deployment Challenges



- Switching to IPv6 is a whole-Internet upgrade
  - All routers, all hosts
  - ICMPv6, DHCPv6, DNSv6
- 2013: 0.94% of Google traffic was IPv6, 2.5% today

Transitioning to IPv6

- How do we ease the transition from IPv4 to IPv6?
  - Today, most network edges are IPv6 ready
    - Windows/OSX/iOS/Android all support IPv6
    - Your wireless access point probably supports IPv6
  - The Internet core is hard to upgrade
  - … 4 core c ute IP

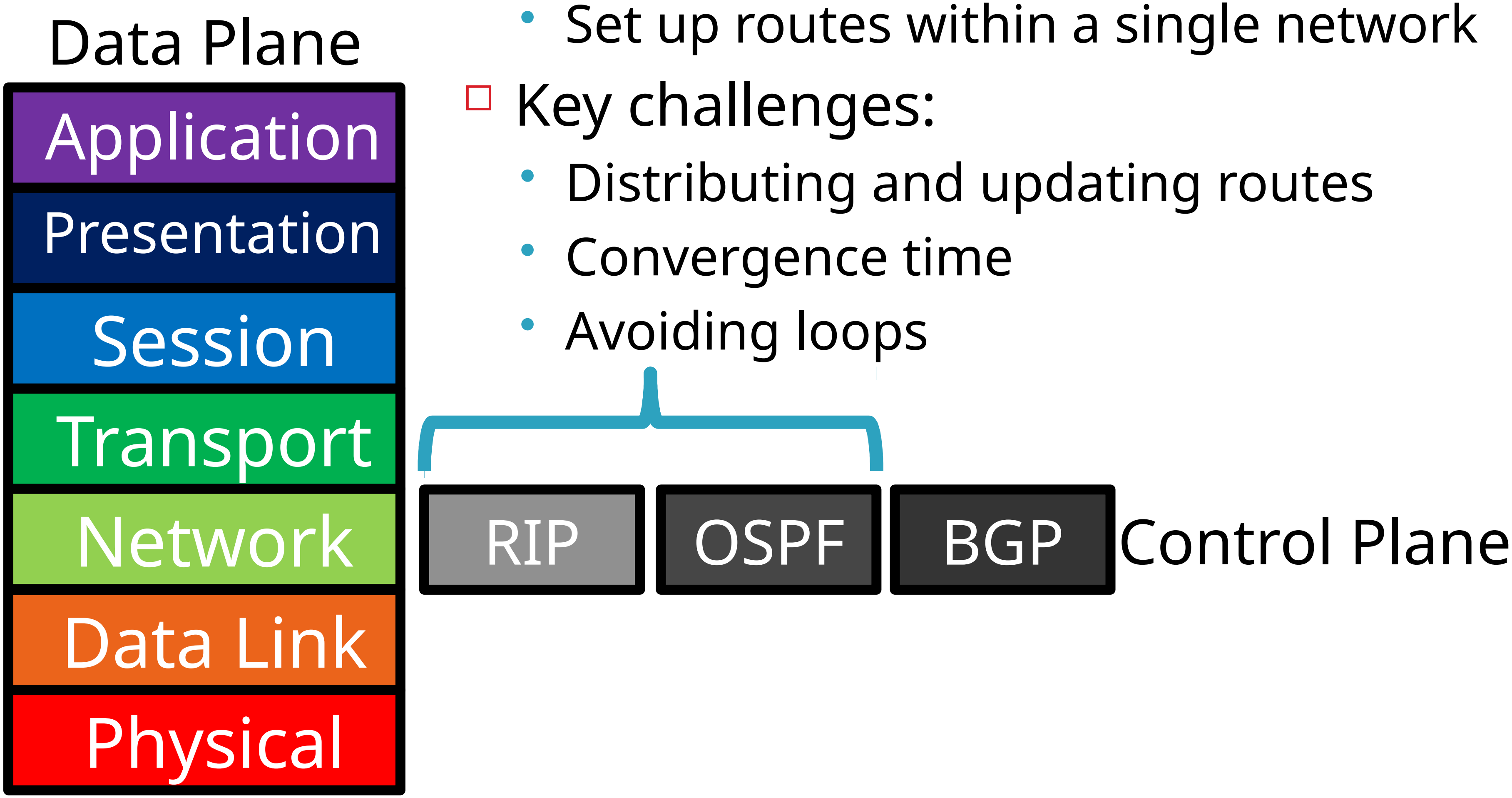


Transition Technologies

- How do you route IPv6 packets over an IPv4 Internet?
- Transition Technologies
  - Use tunnels to encapsulate and route IPv6 packets over the IPv4 Internet
  - Several different implementations
    - 6to4
    - IPv6 Rapid Deployment (6rd)
    - Teredo
    - … etc.

Network Layer, Control Plane

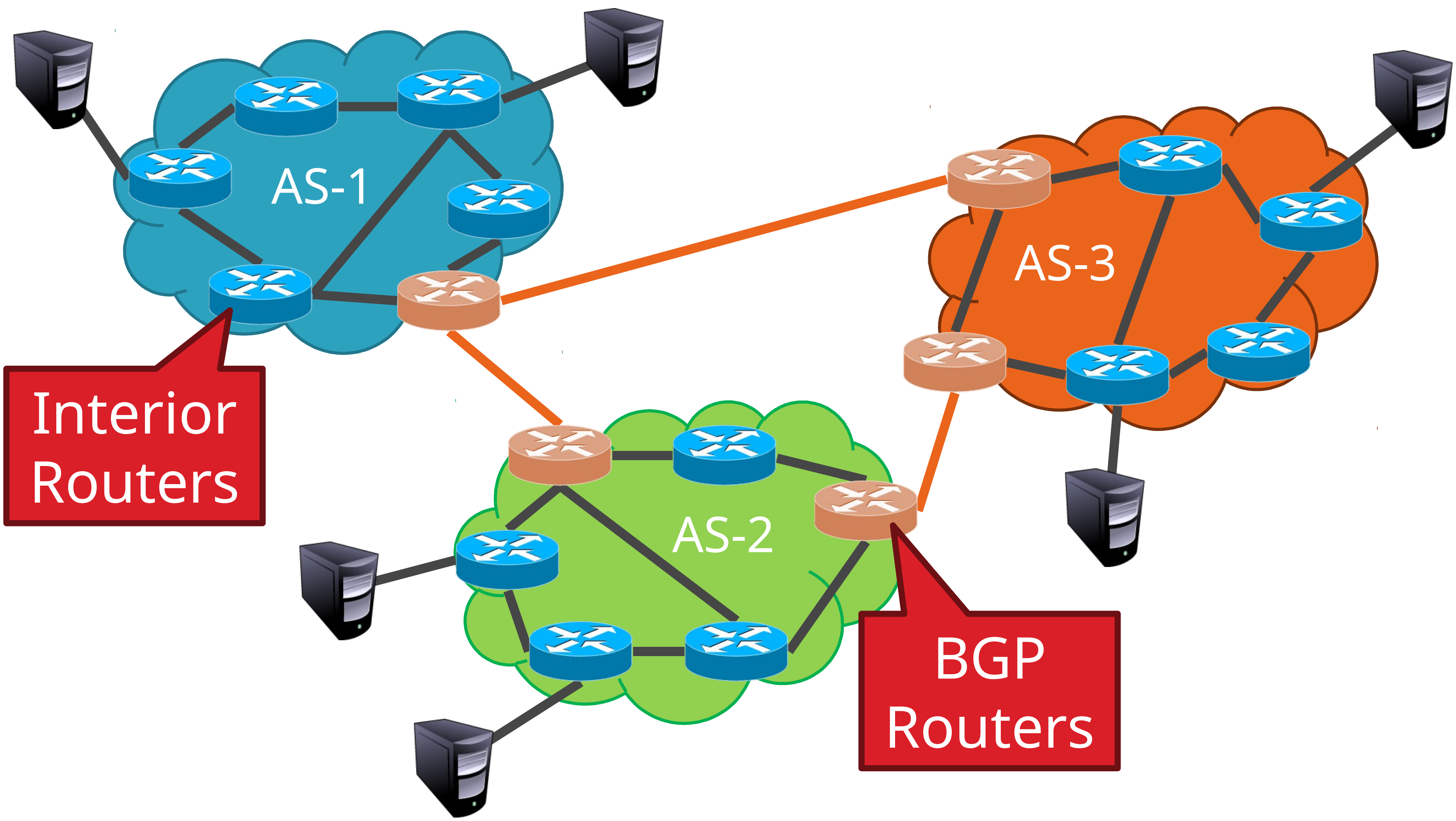
- Function:
  - Set up routes within a single network
- Key challenges:
  - Distributing and updating routes
  - Convergence time
  - Avoiding loops



Internet Routing

- Internet organized as a two level hierarchy
- First level – autonomous systems (AS's)
  - AS – region of network under a single administrative domain
  - Examples: Comcast, AT&T, Verizon, Sprint, etc.
- AS's use intra-domain routing protocols internally
  - Distance Vector, e.g., Routing Information Protocol (RIP)
  - Link State, e.g., Open Shortest Path First (OSPF)
- Connections between AS's use inter-domain routing protocols
  - Border Gateway Routing (BGP)
  - De facto standard today, BGP-4

AS Example



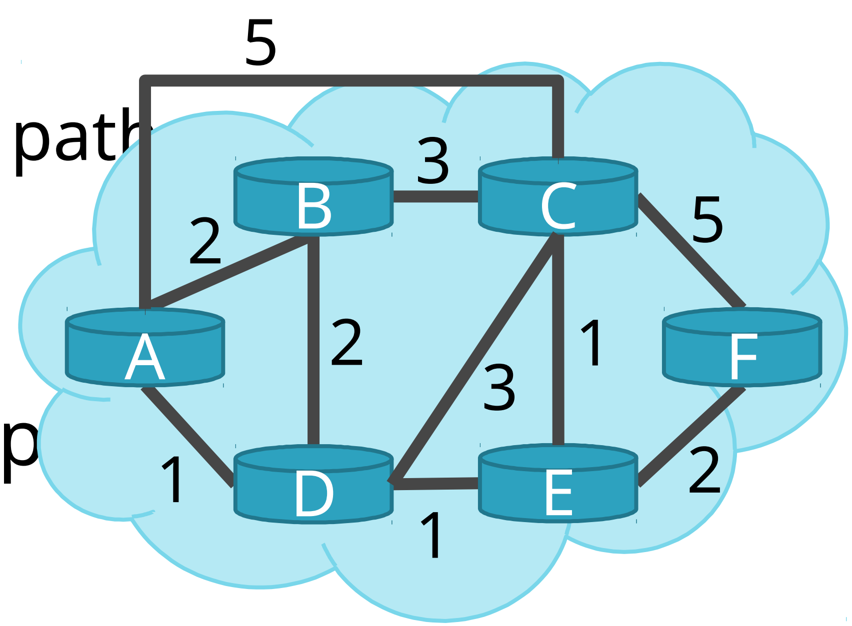
Why Do We Need ASs?

- Routing algorithms are not efficient enough to execute on the entire Internet topology
- Diff
  rou
- Allo
  net
- Allows organizations to choose how to route across each other (BGP)

- Easier to compute routes
- Greater flexibility
- More autonomy/independence

Routing on a Graph

- Goal: determine a “good” path through the network from source to destination
- What is a good path?
  - Usually means the shortest path
  - Load balanced
  - Lowest $$$ cost
- Network modeled as a graph
  - Routers nodes
  - Link edges
    - Edge cost: delay, congestion level, etc.



Shortest Path Routing

1. Bellman-Ford Algorithm [Distance Vector]
2. Dijkstra's Algorithm [Link State]

What does it mean to be the shortest (or optimal) route?

a. **Minimize mean packet delay**
b. **Maximize the network throughput**
c. **Mininize the number of hops along the path**

Dijkstra's Shortest Path Algorithm

Initially mark all nodes (except source) with infinite distance.

working node = source node

Sink node = destination node

While the working node is not equal to the sink

1. Mark the working node as permanent.
2. Examine all adjacent nodes in turn

   If the sum of label on working node plus distance from working node to adjacent node is less than current labeled distance on the adjacent node, this implies a shorter path. Relabel the distance on the adjacent node and label it with the node from which the probe was made.

3. Examine all tentative nodes (not just adjacent nodes) and mark the node with the smallest labeled value as permanent. This node becomes the new working node.

Reconstruct the path backwards from sink to source.

Dijkstra's Algorithm

Dijkstra(**graph** (G,w), **vertex** s)
    InitializeSingleSource(G, s)
    S $\leftarrow \emptyset$
    Q $\leftarrow$ V[G]
    while Q $\neq$ 0 **do**
        u $\leftarrow$ *ExtractMin*(Q)
        S $\leftarrow$ S $\cup$ {u}
        for u $\in$ *Adj*[u] **do**
            Relax(u,v,w)

executed $\Theta(V)$ times

$\Theta(E)$ times in total

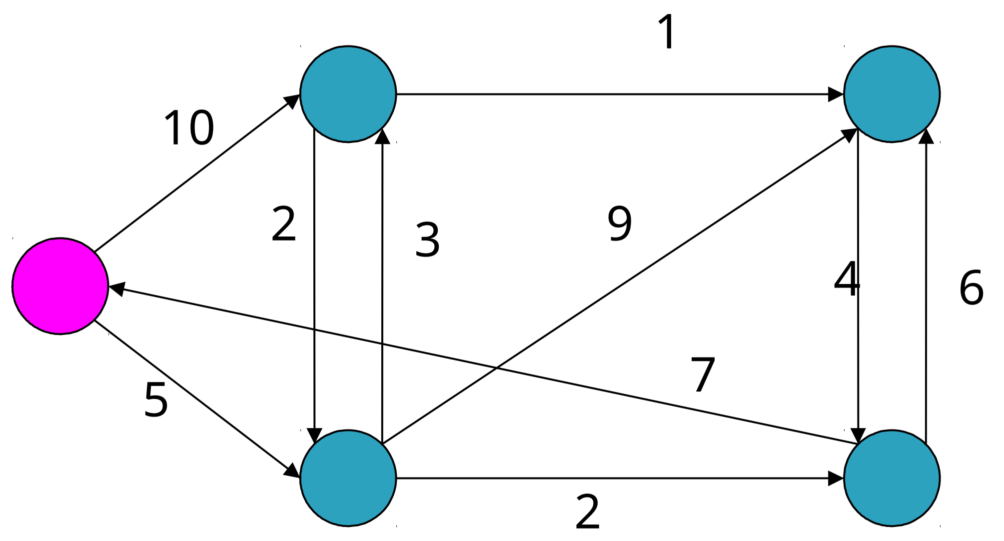
InitializeSingleSource(**graph** G, **vertex** s)
    for v $\in$ V[G] **do**
        d[v] $\leftarrow \infty$
        p[v] $\leftarrow$ 0
    d[s] $\leftarrow$ 0

$\Theta(V)$

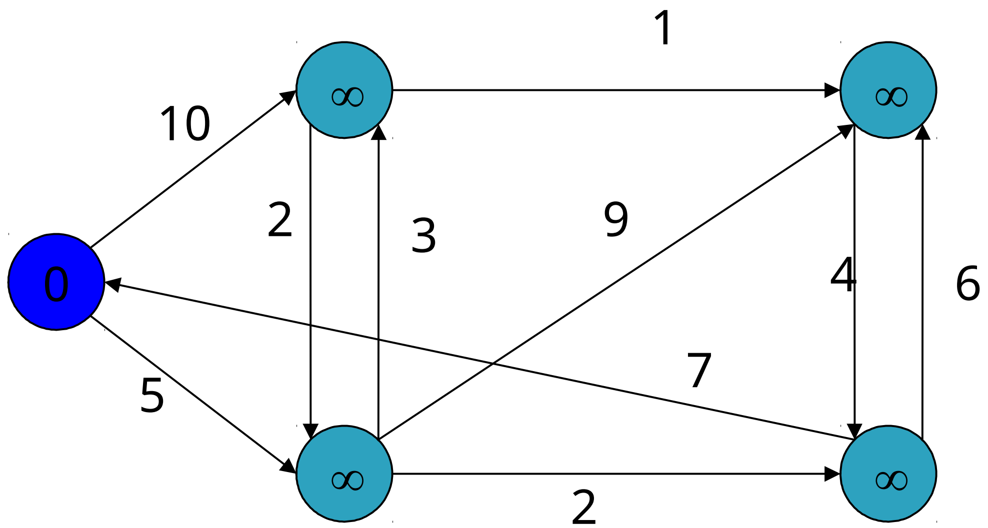
Relax(**vertex** u, **vertex** v, **weight** w**)**
    if d[v] > d[u] + w(u,v) **then**
        d[v] $\leftarrow$ d[u] + w(u,v)
        p[v] $\leftarrow$ u

$\Theta(1)$?

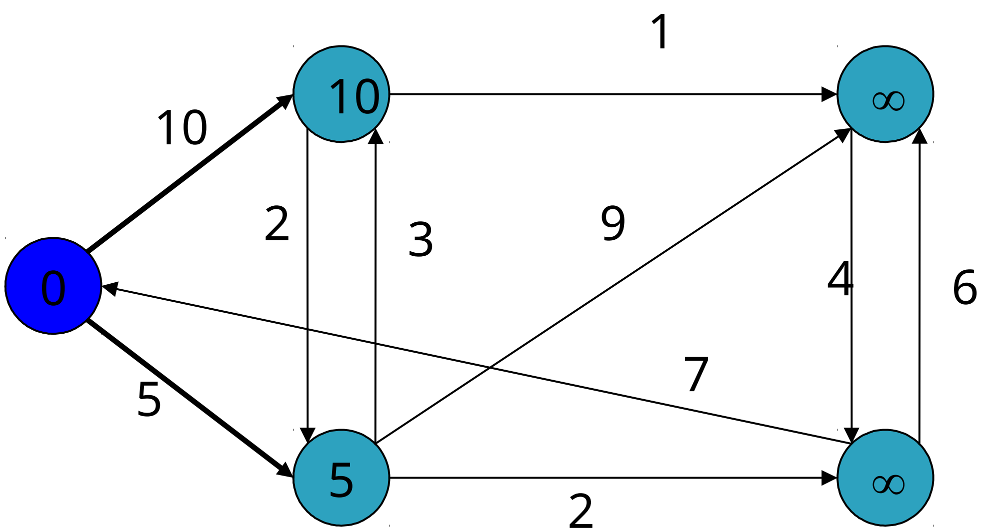
Dijkstra’s Algorithm - Example



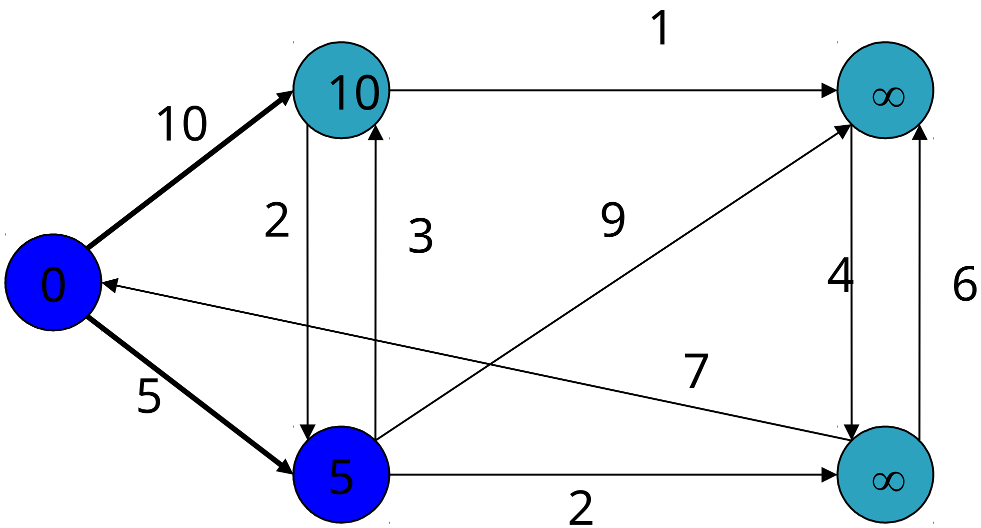
Dijkstra’s Algorithm - Example



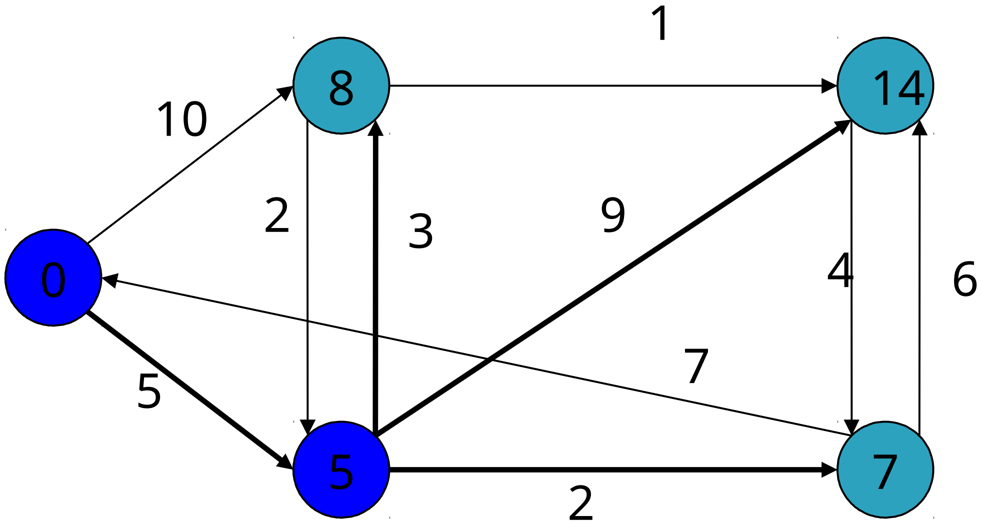
Dijkstra’s Algorithm - Example



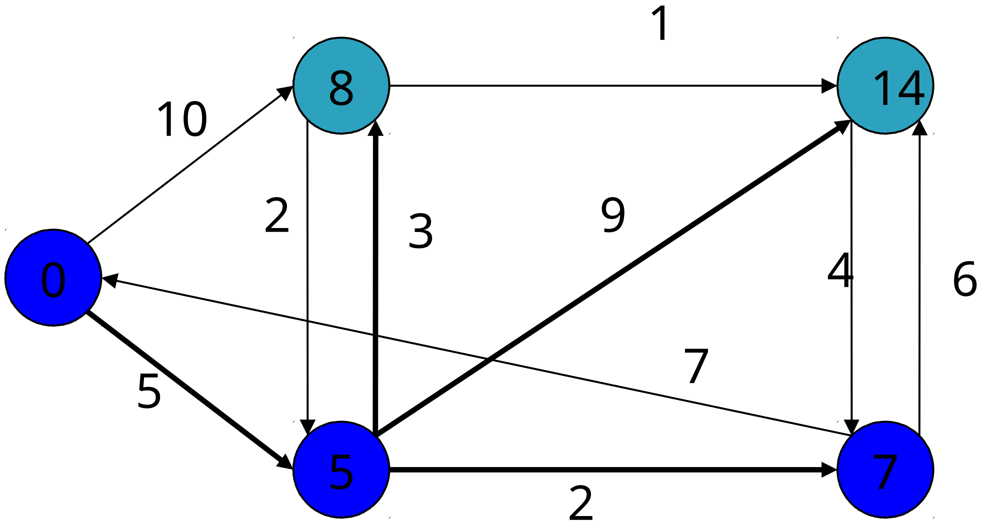
Dijkstra’s Algorithm - Example



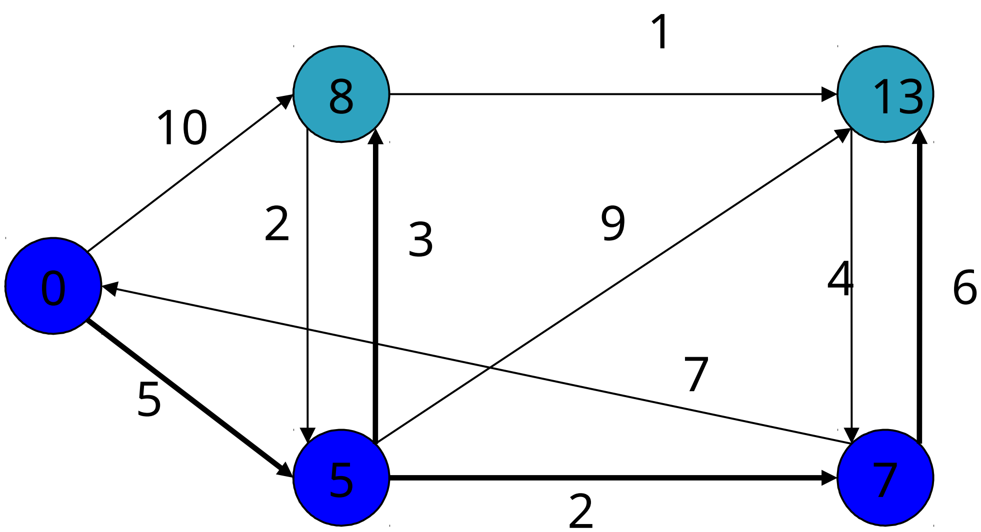
Dijkstra's Algorithm - Example



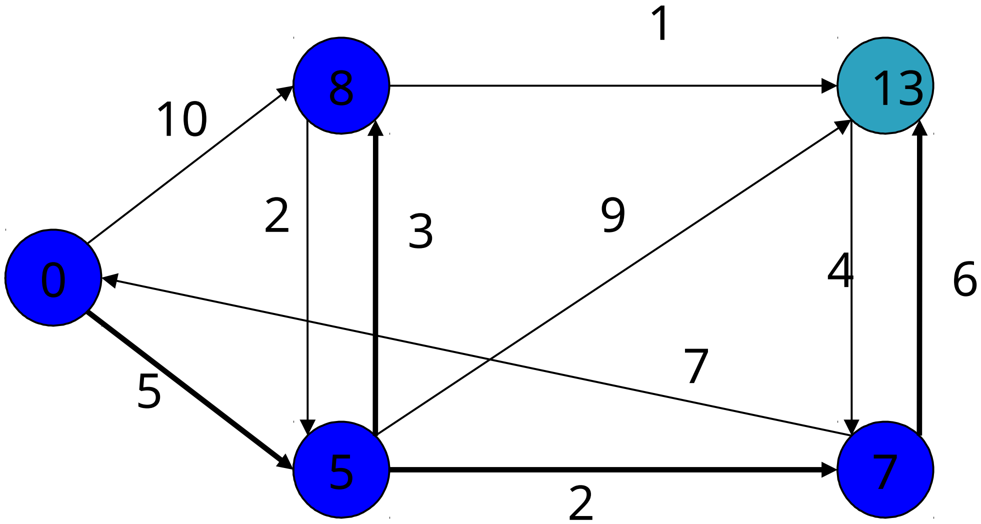
Dijkstra’s Algorithm - Example



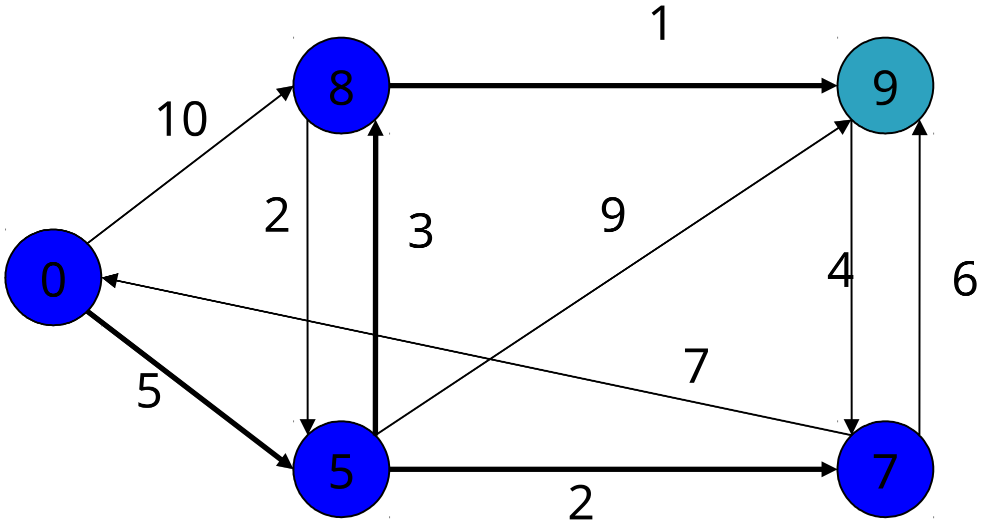
Dijkstra’s Algorithm - Example



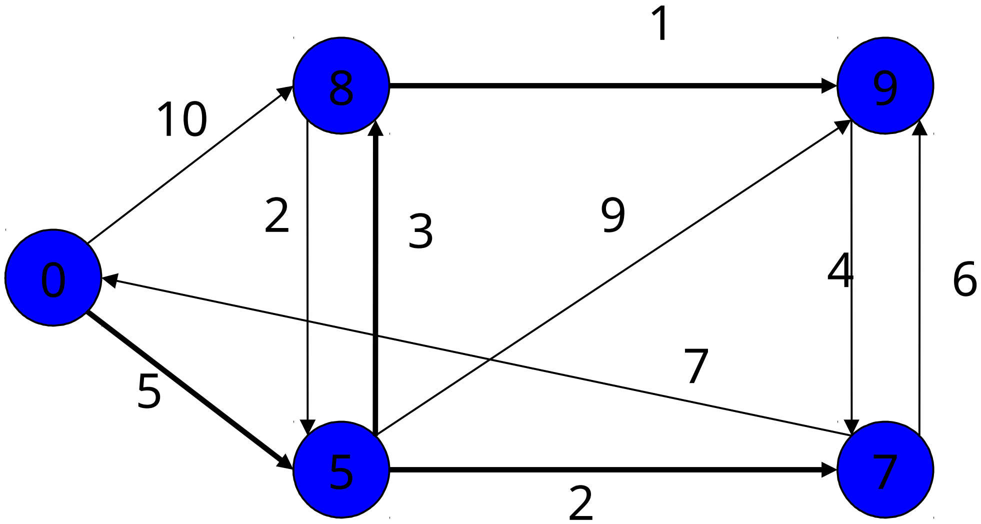
Dijkstra’s Algorithm - Example



Dijkstra’s Algorithm - Example



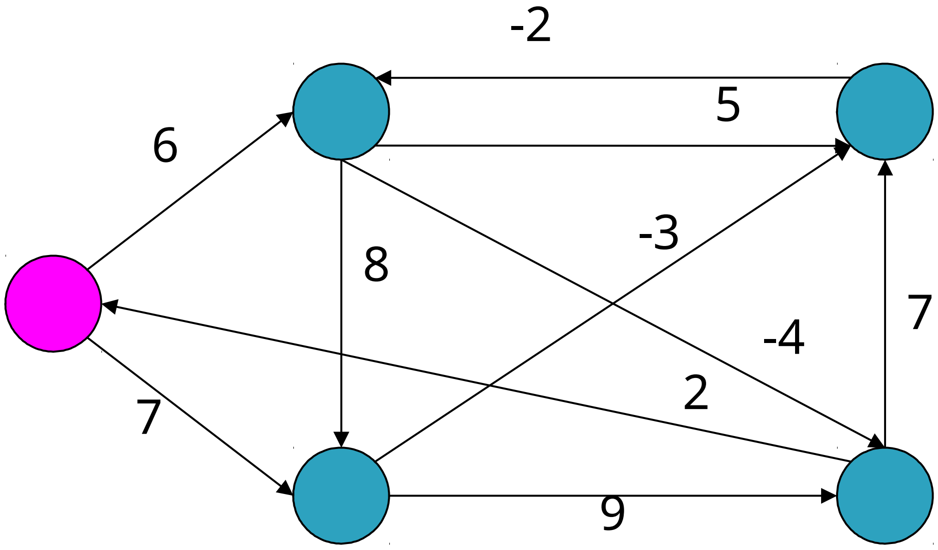
Dijkstra’s Algorithm - Example



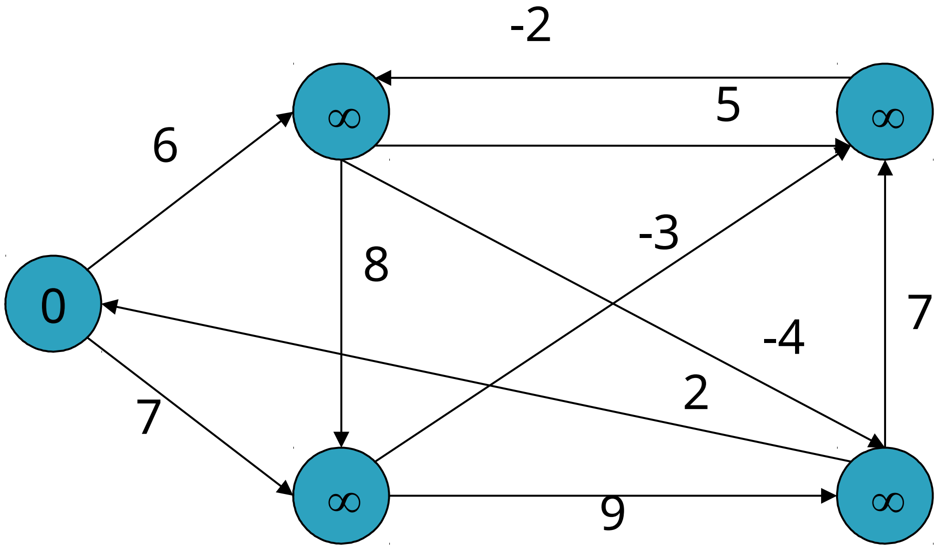
Bellman-Ford Algorithm

```
BellmanFord(graph (G,w), vertex s)
    InitializeSingleSource(G, s)
    for i ← 1 to |V[G] - 1| do
        for (u,v) ∈ E[G] do
            Relax(u,v,w)
    for (u,v) ∈ E[G] do
        if d[v] > d[u] + w(u,v) then
            return false
    return true
```

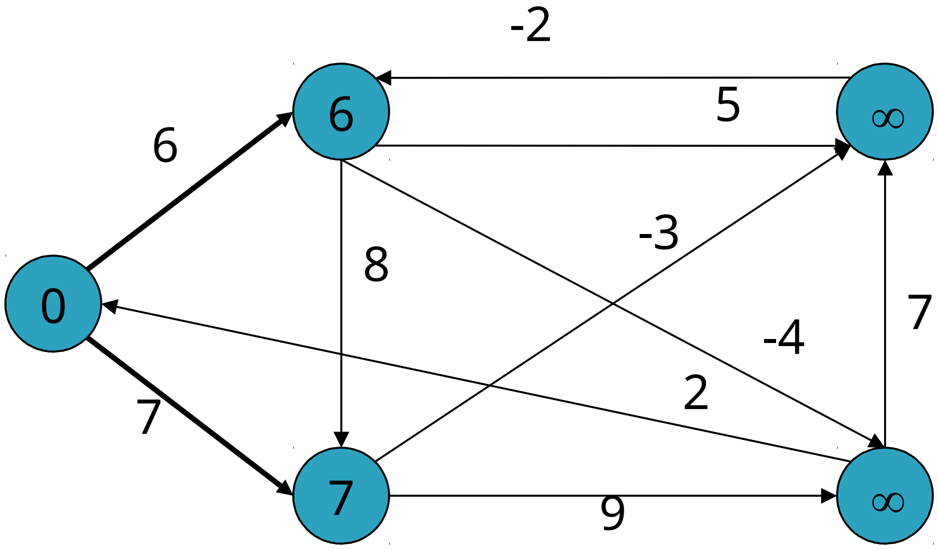
Bellman-Ford Algorithm - Example



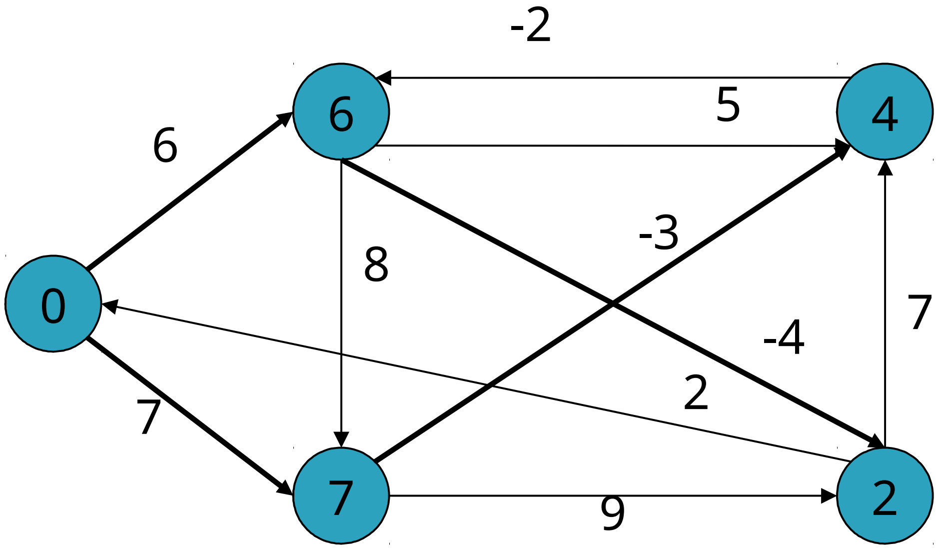
Bellman-Ford Algorithm - Example



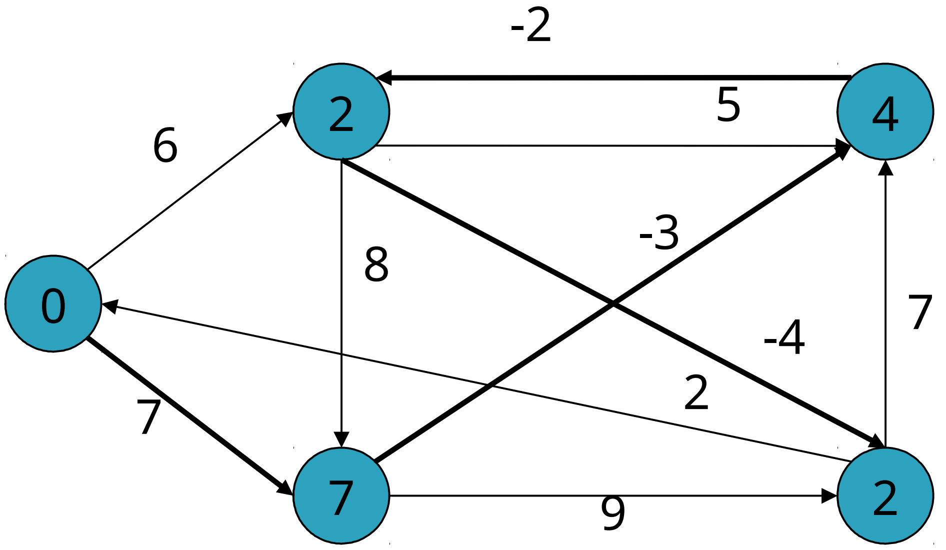
Bellman-Ford Algorithm - Example



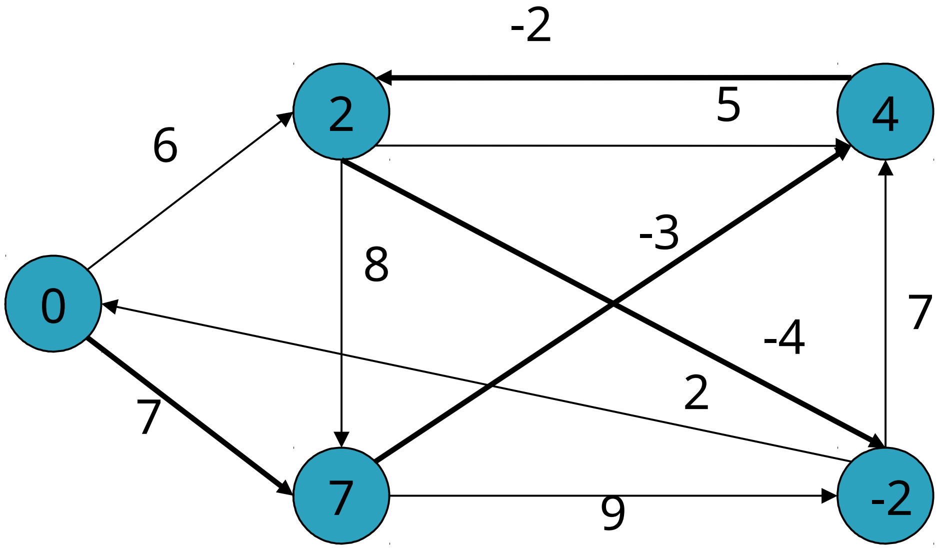
Bellman-Ford Algorithm - Example



Bellman-Ford Algorithm - Example



Bellman-Ford Algorithm - Example



Bellman-Ford Algorithm - Complexity

BellmanFord(**graph** (G,w), **vertex** s)
    InitializeSingleSource(G, s)
    for *i* $\leftarrow$ 1 **to** |V[G] **-** 1| **do**
        for *(u,v)* $\in$ E[G] **do**
            Relax(u,v,w)
    for *(u,v)* $\in$ E[G] **do**
        if *d[v]* > *d[u]* + *w(u,v)* **then**
            return false
    return true

executed $\Theta(V)$ times

$\Theta(E)$

$\Theta(1)$

$\Theta(E)$

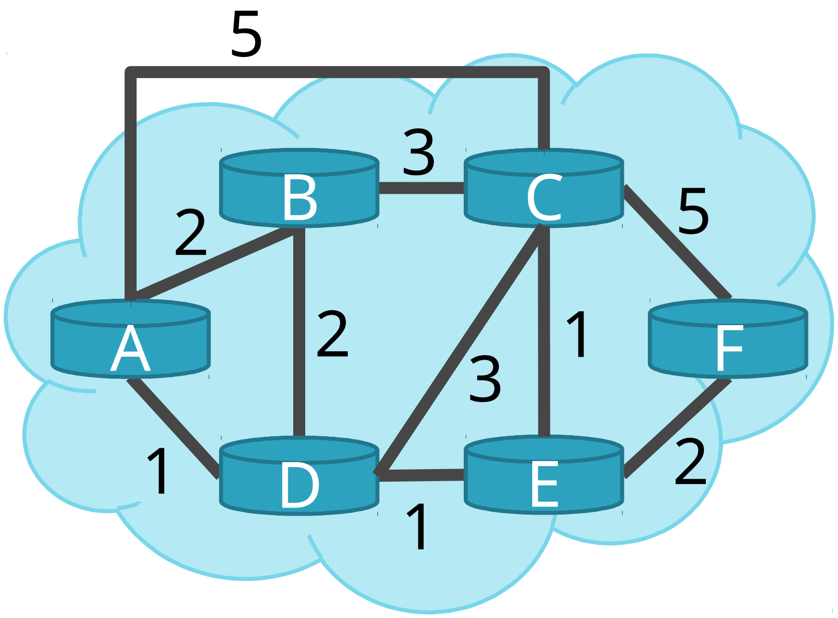
Internetwork Routing [Halsall]

Adaptive Routing

- Centralized
  - [RCC]
- Distributed
  - [IGP] Intradomain routing — Interior Gateway Protocols
    - Distance Vector routing
      - [RIP]
    - Link State routing
      - [OSPF,IS-IS,PNNI]
  - Interdomain routing [EGP] — Exterior Gateway Protoco
    - [BGP,IDRP]

Routing Problems

- Assume
  - A network with N nodes
  - Each node only knows
    - Its immediate neighbors
    - The cost to reach each neighbor
- How does each node learn the shortest path to every other node?



Intra-domain Routing Protocols

- Distance vector
  - Routing Information Protocol (RIP), based on Bellman-Ford
  - Routers periodically exchange reachability information with neighbors
- Link state
  - Open Shortest Path First (OSPF), based on Dijkstra
  - Each network periodically floods immediate reachability information to all other routers
  - Per router local computation to determine full routes

Outline

- Distance Vector Routing
  - RIP
- Link State Routing
  - OSPF
  - IS-IS

Distance Vector Routing

- What is a distance vector?
  - Current best known cost to reach a destination
- Idea: exchange vectors among neighbors to learn about lowest cost paths

DV Table at Node C

Destination	Cost
A	7
B	1
D	2
E	5
F	1

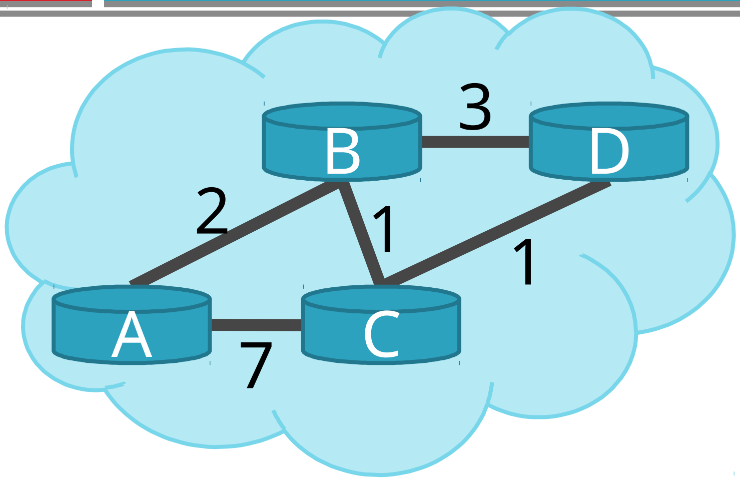
- No entry for C
- Initially, only has info for immediate neighbors
  - Other destinations cost = ∞
- Eventually, vector is filled

- Routing Information Protocol (RIP)

Distance Vector Routing Algorithm

1. Wait for change in local link cost or message from neighbor
2. Recompute distance table
3. If least cost path to any destination has changed, notify neighbors

Distance Vector Initialization



Node A

Dest.	Cost	Next
B	2	B
C	7	C
D	∞	

Node B

Dest.	Cost	Next
A	2	A
C	1	C
D	3	D

1. **Initialization:**
2. **for all** neighbors *V* **do**
3. **if** *V* adjacent to *A*
4. D(*A*, *V*) = c(*A*,*V*);
5. **else**
6. D(*A*, *V*) = ∞;

…

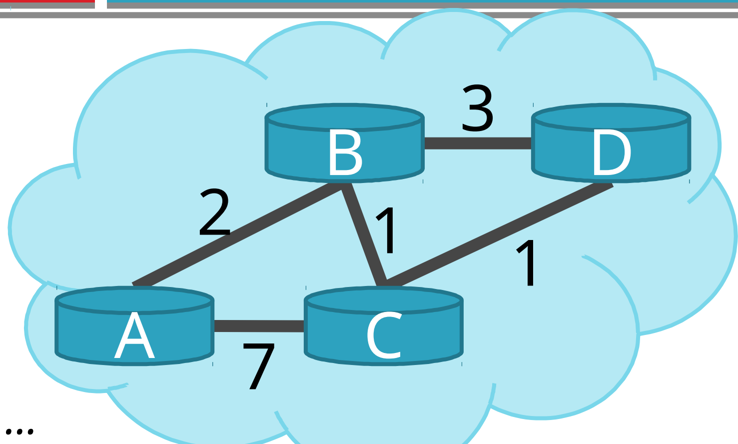
Node C

Dest.	Cost	Next
A	7	A
B	1	B
D	1	D

Node D

Dest.	Cost	Next
A	∞	
B	3	B
C	1	C

Distance Vector: End of 3rd Iteration




```
…
loo
…
els
  fo



    else
      D(A, Y) =
              min(D(A, Y),
              D(A, V) + D(V, Y));
if (there is a new min. for dest. Y)
  send D(A, Y) to all neighbors
forever
```

Node A

Dest.	Cost	Next
B	2	B
C	3	B
D	4	B

Node B

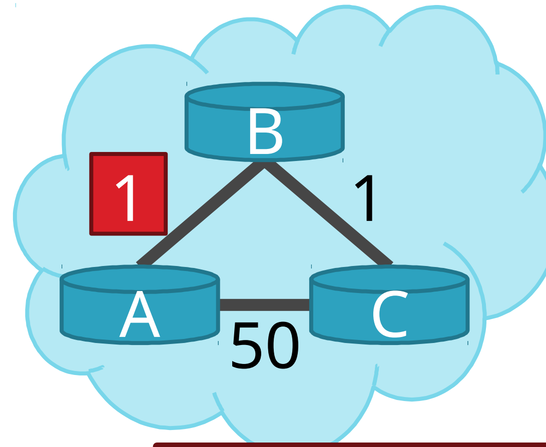
Dest.	Cost	Next
A	2	A
C	1	C
D	2	C

- Nothing changes, algorithm terminates
- Until something changes…

Dest.	Cost	Next
A	3	B
B	1	B
D	1	D

Dest.	Cost	Next
A	4	C
B	2	C
C	1	C

```
loop:
  wait (link cost update or update message)
  if (c(A,V) changes by d)
    for all destinations Y through V do
      D(A,Y) = D(A,Y) + d
  else if (update D(V, Y) received from V)
    for all destinations Y do
      if (destination Y through V)
        D(A,Y) = D(A,V) + D(V, Y);
      else
        D(A, Y) = min(D(A, Y), D(A, V) + D(V, Y));
```



Link Cos
Algorit

Good news travels fast

lgorithm
erminates

Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

D	C	N
A	1	A
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	2	B
B	1	B

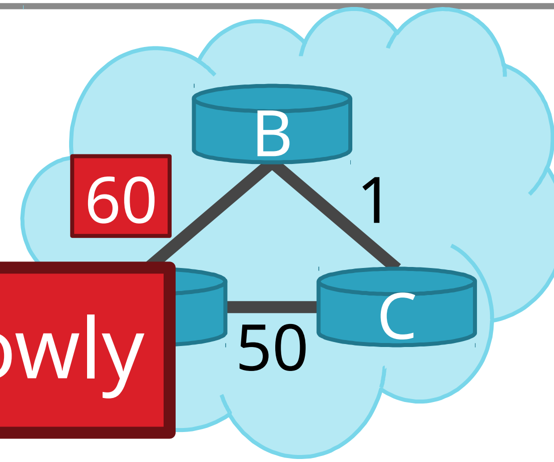
D	C	N
A	2	B
B	1	B

Time

Count to Infinity Problem

- Node B knows D(C, A) = 5
- However, B does not know the path
- Thus, D(

Bad news travels slowly



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	6	C
C	1	B

D	C	N
A	8	C
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

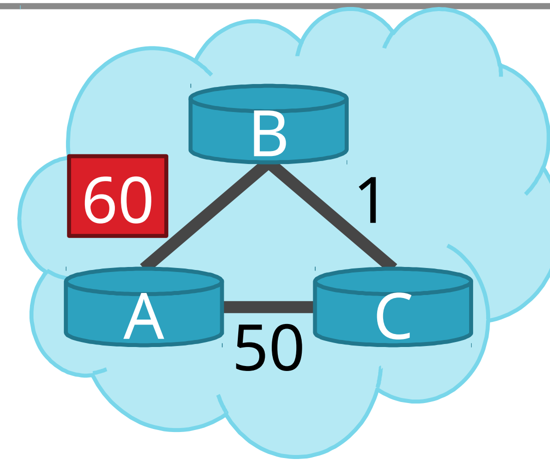
D	C	N
A	7	B
B	1	B

D	C	N
A	7	B
B	1	B

Time

Poisoned Reverse

- If C routes through B to get to A
  - C tells B that D(C, A) = ∞
  - Thus, B won't route to A via C



Node B

D	C	N
A	4	A
C	1	B

D	C	N
A	**60**	A
C	1	B

D	C	N
A	60	A
C	1	B

D	C	N
A	**51**	**C**
C	1	B

Node C

D	C	N
A	5	B
B	1	B

D	C	N
A	5	B
B	1	B

D	C	N
A	**50**	**A**
B	1	B

D	C	N
A	50	A
B	1	B

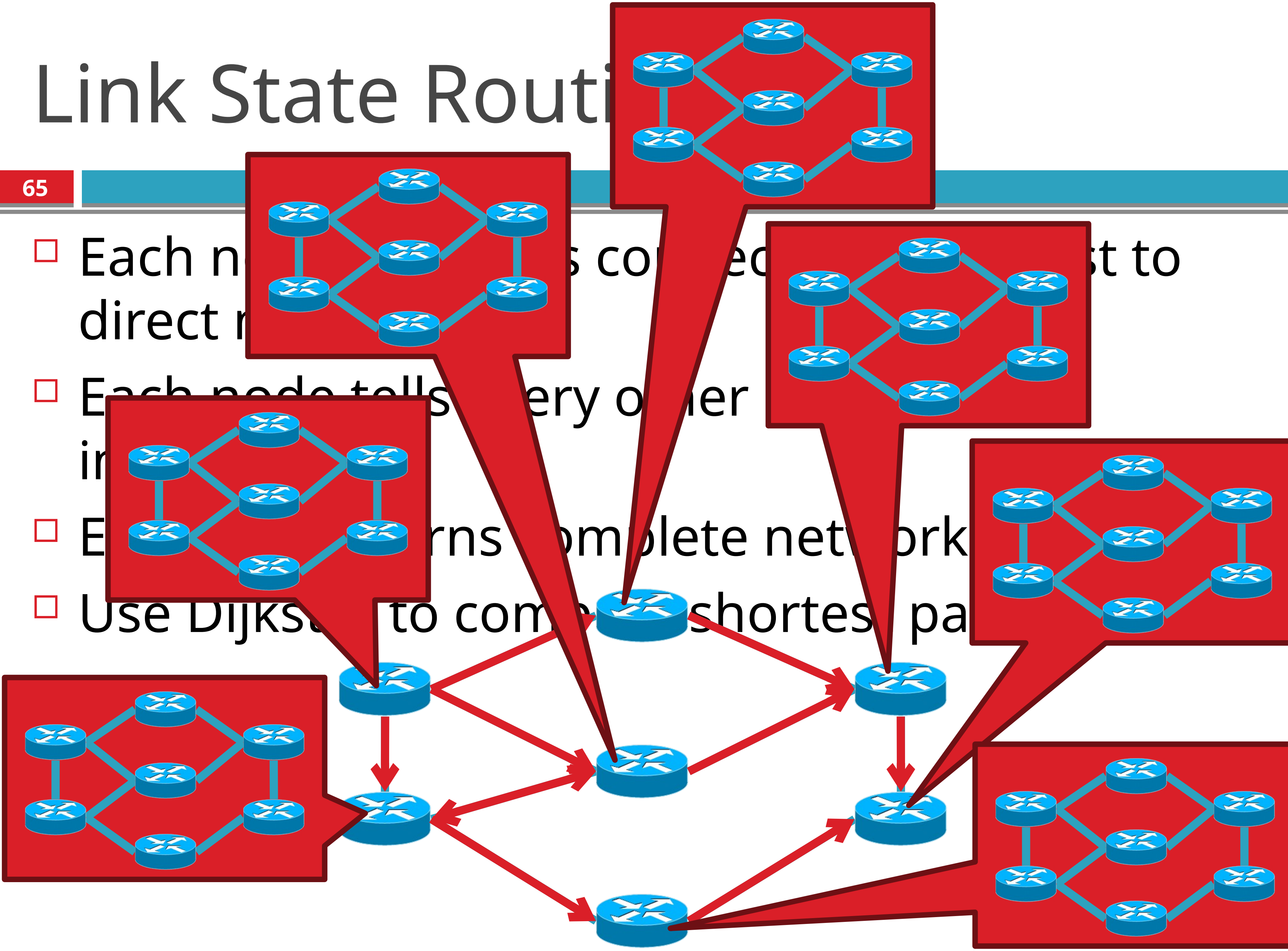
Time

Outline

- Distance Vector Routing
  - RIP
- Link State Routing
  - OSPF
  - IS-IS

Link State Routi

- Each n s co ec st to direct r
- Each node tells ery o er in
- E rns omplete net ork
- Use Dijks to com shortes pa

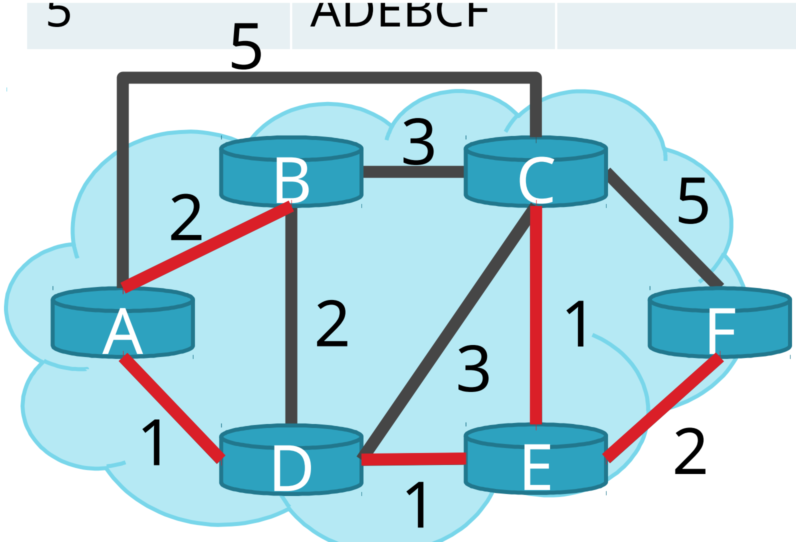


Flooding Details

- Each node periodically generates Link State Packet
  - ID of node generating the LSP
  - List of direct neighbors and costs
  - Sequence number (64-bit, assumed to never wrap)
  - Time to live
- Flood is reliable (ack + retransmission)
- Sequence number "versions" each LSP
- Receivers flood LSPs to their own neighbors
  - Except whoever originated the LSP
- LSPs also generated when link states change

Dijkstra's Algorithm

Step	Start S	B	C	D	E	F
0	A	2, A	5, A	1, A	∞	∞
5	ADEBCF					



1. ***Initialization:***
2. S = {A};
3. for all nodes *v*
4. if *v* adjacent to *A*
5. then D(v) = c(*A*,*v*);
6. else D(v) = ∞;

…

8. ***Loop***
9. find w not in S s.t. D(w) is a minimum;
10. add w to S;
11. update D(v) for all v adjacent to w and not in S:
12. D(v) = min(D(v), D(w) + c(w,v));

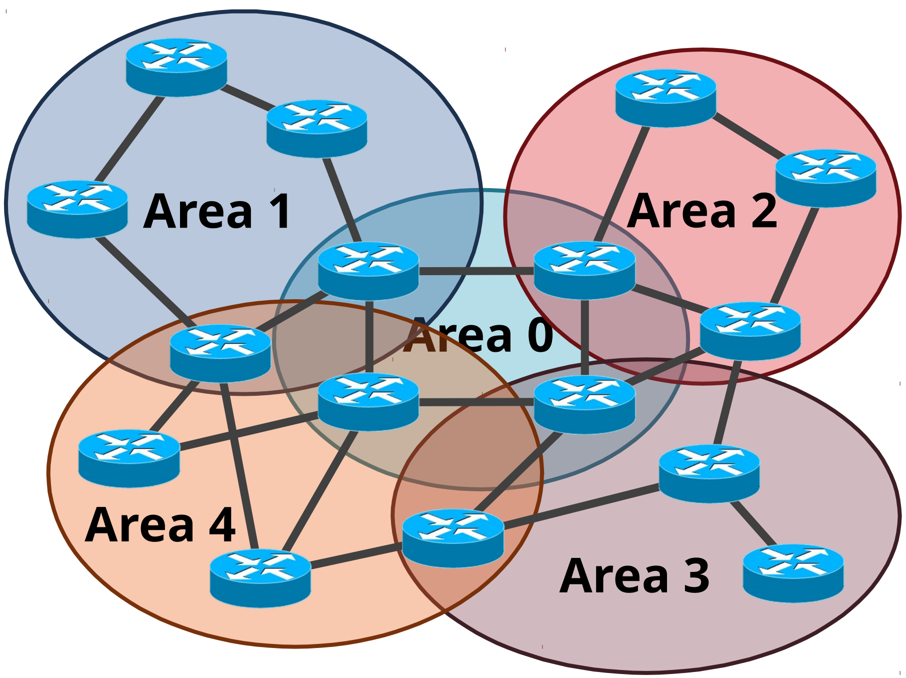
OSPF vs. IS-IS

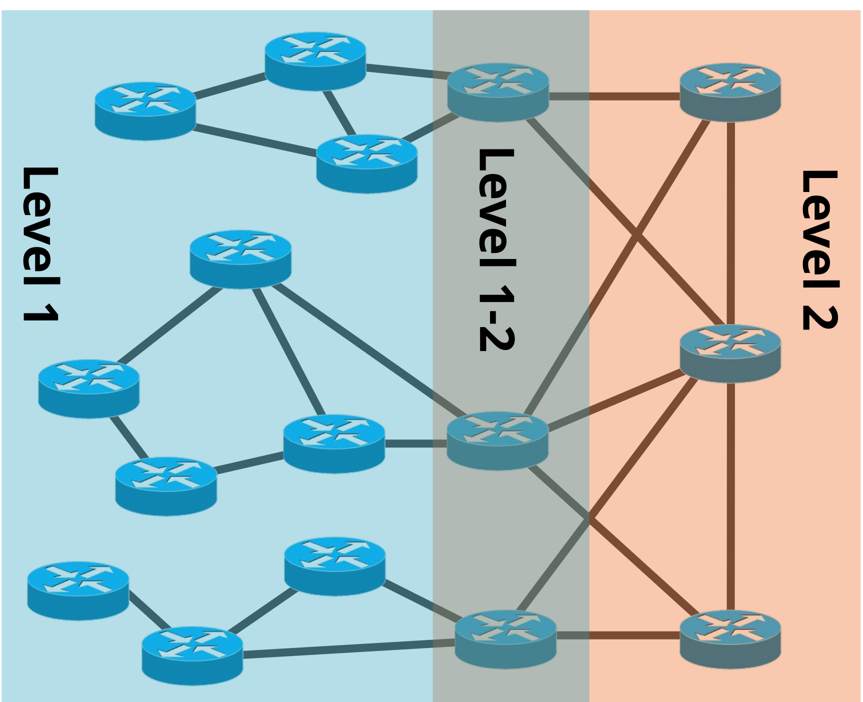
- Two different implementations of link-state routing

OSPF	IS-IS
Favored by companies, datacenters	Favored by ISPs
More optional features	Less "chatty" • Less network overhead • Supports more devices
Built on top of IPv4 • LSAs are sent via IPv4 • OSPFv3 needed for IPv6	Not tied to IP • Works with IPv4 or IPv6

Different Organizational Structure

OSPF	IS-IS
Organized around overlapping areas	Organized as a 2-level hierarchy
Area 0 is the core network	Level 2 is the backbone





Link State vs. Distance Vector

	Link State	Distance Vector
Message Complexity	$O(n^2*e)$	$O(d*n*k)$
Time Complexity	$O(n*\log n)$	$O(n)$
Convergence Time	$O(1)$	$O(k)$
Robustness	• Nodes may advertise incorrect link costs • Each node computes	• Nodes may advertise incorrect path cost • Errors propagate due to

- Which is best?
- In practice, it depends.
- In general, link state is more popular.