L8 – Routing Protocols and IP

50.012 Networks

Jit Biswas

Cohort 1: TT7&8 (1.409-10)

Cohort 2: TT24&25 (2.503-4)

Introduction

- Todays lecture:
  - More on routing
    - RIP
    - OSPF
    - BGP
  - The internet protocol
    - IPv4 packet format
    - Datagram fragmentation
    - NAT
    - DHCP
- Note: large parts of this slide set are based on Kurose & Ross chapter 4 slide set

Overview of LO Progress so far

Touched so far:

- LO1: Explain fundamental network protocols
- LO2: Paraphrase the organization of computer networks
- LO3: Solve standard problems in interconnections between AS
- LO7: Design and implement a server-client architecture based on sockets

Outstanding LOs:

- LO4: Model the Internet structure and derive operational parameters
- LO5: Design optimized network topology for given problem settings
- LO6: Judge and evaluate a provided network setup

Routing Continued

Link state routing

- Dijkstra’s algorithm was an example for link state algorithms
- We start from one node, and iteratively include knowledge of all other nodes
- Messages that need to be exchanged: O(|E|*|V|)
  - Each router sends cost information for all neighbors
- This does not scale on the Internet
  - |E| and |V| is huge -> product is bigger
  - We would have to run Dijkstra’s algorithm for each router
- We need an algorithm that builds on local information

Distance vector algorithm

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Bellman-Ford equation (dynamic programming)

let

$d_x(y)$:= cost of least-cost path from x to y

then

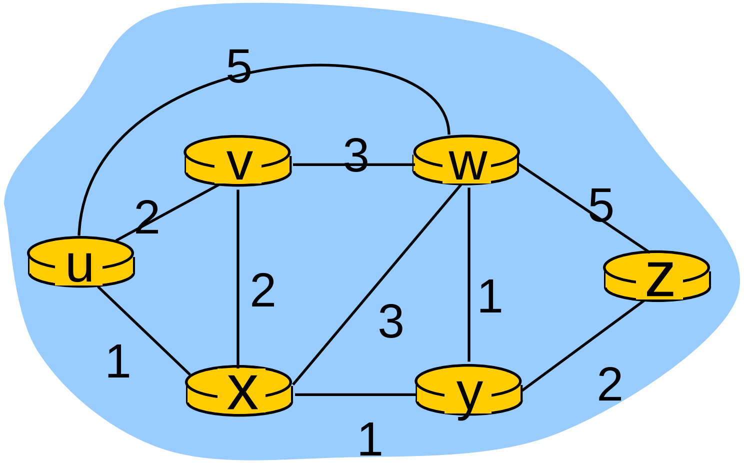
$$d_x(y) = \min_v \{c(x,v) + d_v(y)\}$$

cost from neighbor v to destination y

cost to neighbor v

min taken over all neighbors v of x

Bellman-Ford example



clearly, $d_v(z) = 5$, $d_x(z) = 3$, $d_w(z) = 3$

B-F equation says:

$$
\begin{aligned}
d_u(z) = \min \{ & c(u,v) + d_v(z), \\
& c(u,x) + d_x(z), \\
& c(u,w) + d_w(z) \} \\
= \min \{ & 2 + 5, \\
& 1 + 3, \\
& 5 + 3\} = 4
\end{aligned}
$$

node achieving minimum is next hop in shortest path, used in forwarding table

Distance vector algorithm

- $D_x(y)$ = estimate of least cost from x to y
  - x maintains distance vector $\mathbf{D}_x = [D_x(y): y \in N]$
- node x:
  - knows cost to each neighbor v: $c(x,v)$
  - maintains its neighbors' distance vectors. For each neighbor v, x maintains
    $\mathbf{D}_v = [D_v(y): y \in N]$

Distance vector algorithm

key idea:

- from time-to-time, each node sends its own distance vector estimate to neighbors
- when x receives new DV estimate from neighbor, it updates its own DV using B-F equation:

$$D_x(y) \leftarrow \min_v\{c(x,v) + D_v(y)\} \quad \text{for each node } y \in N$$

- under minor, natural conditions, the estimate $D_x(y)$ *converge to the actual least cost* $d_x(y)$

Distance vector algorithm

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

iterative, asynchronous, self-terminating

each local iteration caused by:

- local link cost change
- DV update message from neighbor

distributed

- each node notifies neighbors *only* when its DV changes
  - neighbors then notify their neighbors if necessary

each node:

wait for (change in local link cost or msg from neighbor)

recompute estimates

if DV to any dest has changed, *notify* neighbors

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\}$$
$$= \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\}$$
$$= \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

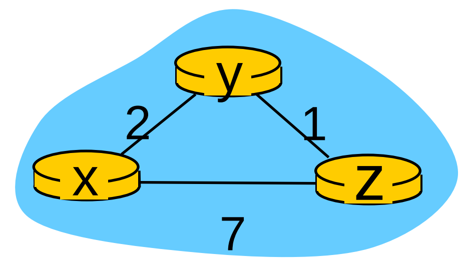
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0



time

$$D_x(y) = \min\{c(x,y) + D_y(y),\ c(x,z) + D_z(y)\} = \min\{2+0\ ,\ 7+1\} = 2$$

$$D_x(z) = \min\{c(x,y) + D_y(z),\ c(x,z) + D_z(z)\} = \min\{2+1\ ,\ 7+0\} = 3$$

node x table

from \ cost to	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

from \ cost to	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

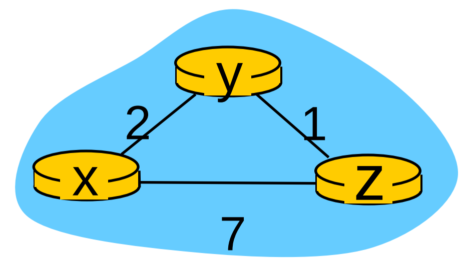
from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

from \ cost to	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

from \ cost to	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

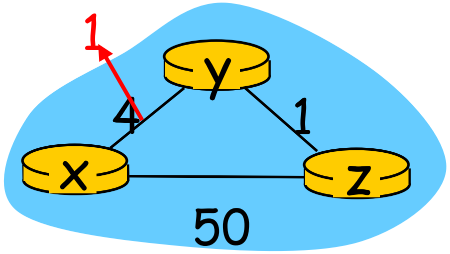


time

Distance vector: link cost changes

link cost changes:

- node detects local link cost change
- updates routing info, recalculates distance vector
- if DV changes, notify neighbors



"good news travels fast"

t_0 : *y* detects link-cost change, updates its DV, informs its neighbors.

t_1 : *z* receives update from *y*, updates its table, computes new least cost to *x* , sends its neighbors its DV.

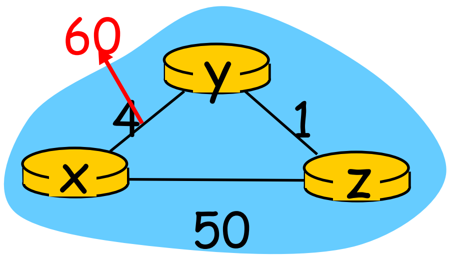
t_2 : *y* receives *z*'s update, updates its distance table. *y*'s least costs do *not* change, so *y* does *not* send a message to *z*.

Distance vector: link cost changes

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

link cost changes:

- node detects local link cost change
- *bad news travels slow* - "count to infinity" problem!
- 44 iterations before algorithm stabilizes: see text



poisoned reverse:

- If Z routes through Y to get to X :
  - Z tells Y its (Z's) distance to X is infinite (so Y won't route to X via Z)
- will this completely solve count to infinity problem?

Comparison of LS and DV algorithms

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

message complexity

- ***LS:*** with n nodes, E links, O(nE) msgs sent
- ***DV:*** exchange between neighbors only
  - convergence time varies

speed of convergence

- ***LS:*** $O(n^2)$ algorithm requires O(nE) msgs
  - may have oscillations
- ***DV:*** convergence time varies
  - may be routing loops
  - count-to-infinity problem

robustness: what happens if router malfunctions?

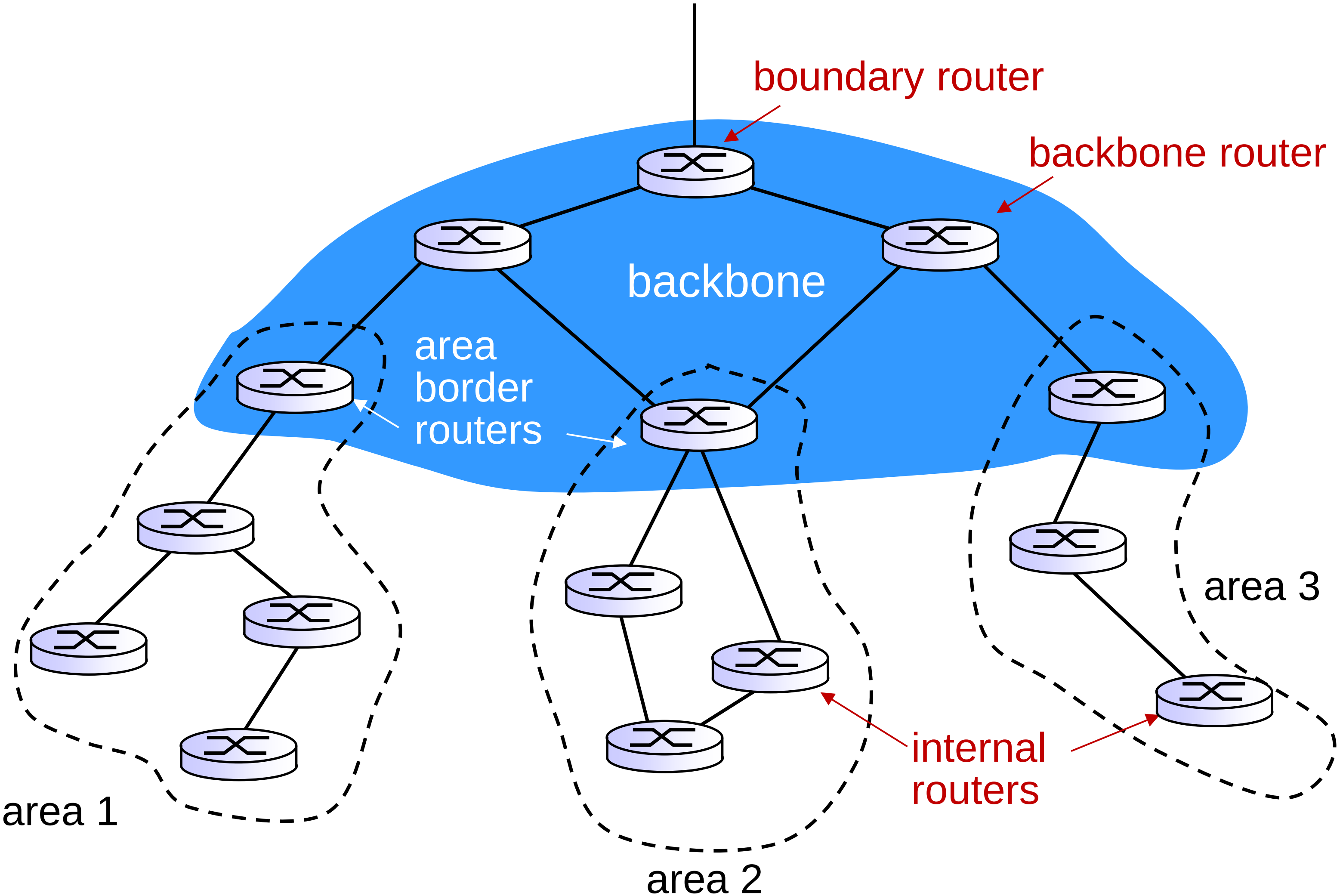
LS:

- node can advertise incorrect *link* cost
- each node computes only its *own* table

DV:

- DV node can advertise incorrect *path* cost
- each node's table used by others
  - error propagate thru network

Hierarchical Routing



Intra-/Inter-AS routing

- Different routing algorithms are used within and outside an AS
- Within an AS:
  - Limited number of links and routers
  - Link state algorithms can be feasible
- Common Intra-AS protocols
  - RIP: Routing Information Protocol (DV)
  - OSPF: Open Shortest Path First (LS)
- Inter-AS routing:
  - LS infeasible, DV (or PV) needed
  - BGP is the most common PV protocol

Routing Information Protocol (RIP)

- Simple Distance Vector protocol
  - Based on Bellman-Ford
- Distance metric always 1
  - Cost of path is #hops
  - Paths lead to CIDR networks
- Exchange of distance vectors between neighbors every 30s
- Size of distance vectors exchanged is limited to 25
- Local routing table contains
  - List of CIDR networks reachable + path cost + interface
- Technically application layer protocol as it uses UDP

Link Failure in RIP

- Regular advertisements are expected every 30 seconds
- If no advertisement exchanged for 180 seconds: declared dead
- Router will delete local routing table entries via dead neighbor
  - Advertise new (reduced) table next
- This will propagate link failure through AS
- Maximum limit on route cost (16) to prevent count-to-infinity

Open-Shortest-Path-First (OSPF)

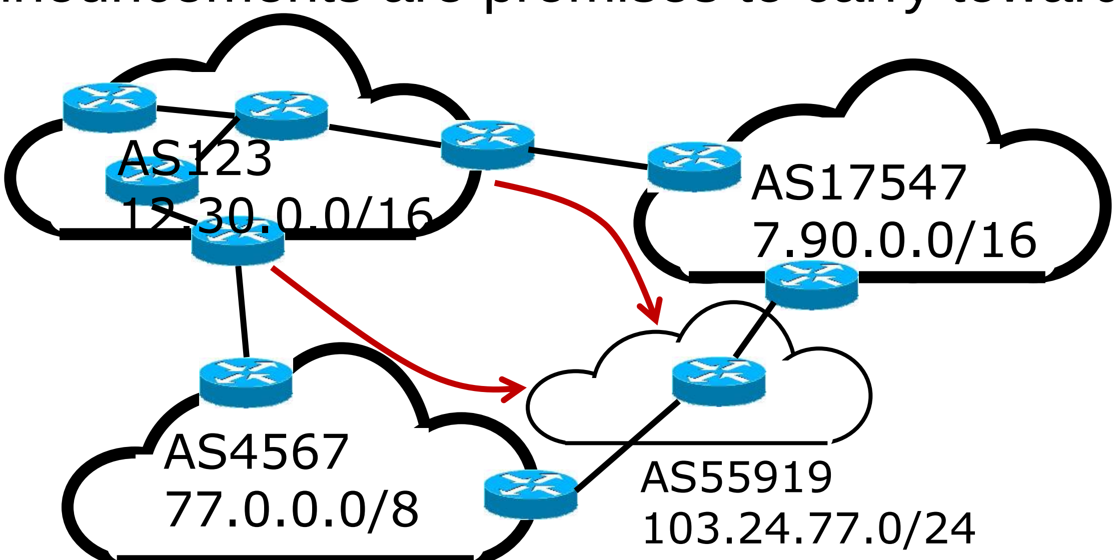
- OSPF uses LS algorithm
  - Essentially Dijkstra's algorithm
- OSPF is network layer protocol
  - Based on local broadcasts of link tables
- Each router has to have full view of AS network
- Each router then runs Dijkstra's algorithm

OSPF vs. RIP

- OSPF allows more complex link cost
  - e.g. different cost for real-time traffic
- RIP still has update problem, but mitigated
- OSPF scales worse for larger AS
- There is a hierarchical OSPF version for large AS
  - Network is fragmented into areas connected by border routers
  - Broadcasts contained to areas, solve areas individually
- OSPF messages use cryptographic signatures

Border Gateway Protocol (BGP)

- BGP is a path vector protocol
  - Paths are announced, without distances
  - Paths actually include sequence of nodes on the path
- Two variants:
  - eBGP for communication between AS
  - iBGP to propagate routes internally within AS
- eBGP is based on TCP connections between routers
- eBGP announcements are promises to carry towards target prefix



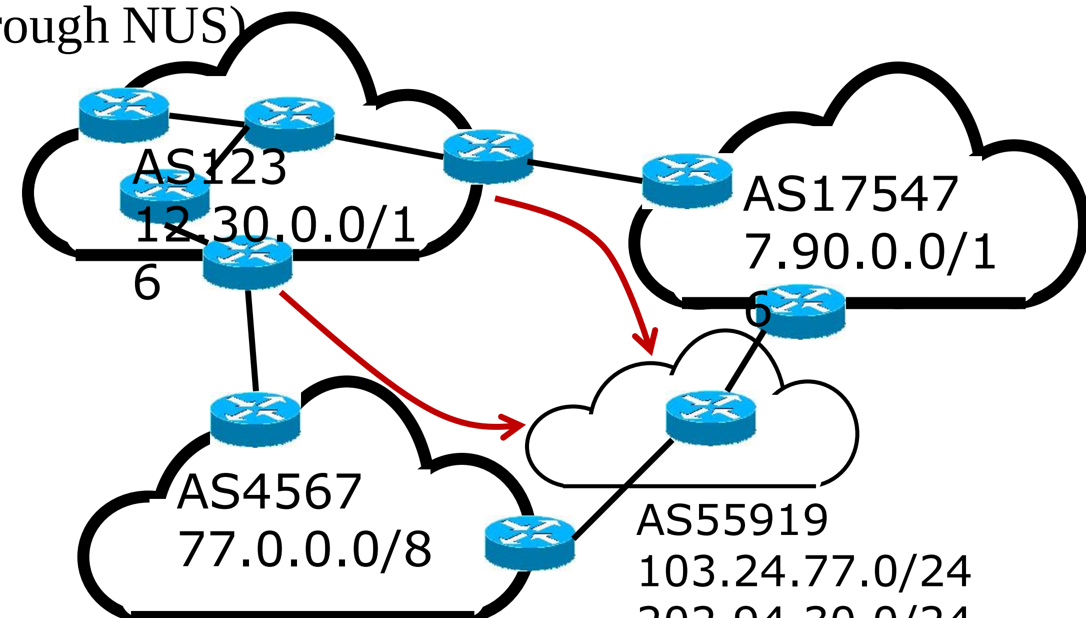
http://bgp.he.net/AS55919#_asinfo

eBGP announcements

- Advertised route contains:
  - Target prefix (CIDR network)
  - BGP attributes
- Two main attributes:
  - AS-PATH: list of AS that are on path
  - NEXT-HOP: IP address of router to reach current AS
- The receiver of announcement can decide to import route
  - Depends on local *policies*
- Example for SUTD AS
  - 103.24.77.0/24
  - AS-PATH AS55919
  - NEXT-HOP 103.24.77.1

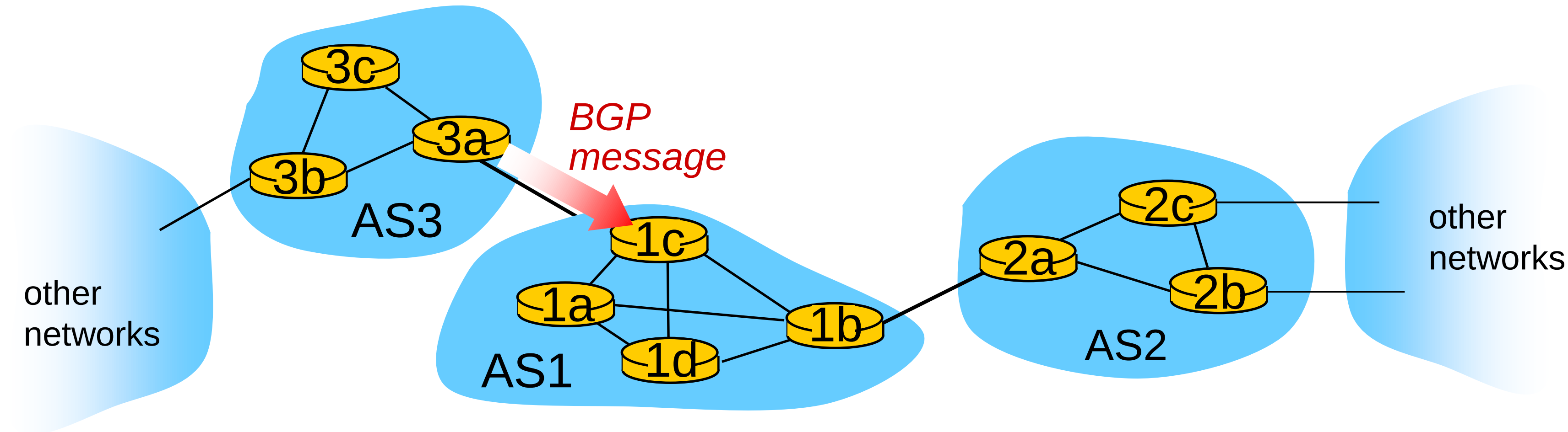
eBGP Route Selection

- Receiver of BGP announcement might already have route entries to some announced prefix
- There should only be one entry to every prefix
  - How to select which one?
- Selection based on:
  - Length of AS-PATH (not directly hops, number of AS)
  - Local policies (do not route through NUS)
  - *Hot Potato Routing*: least hops to NEXT-HOP of route



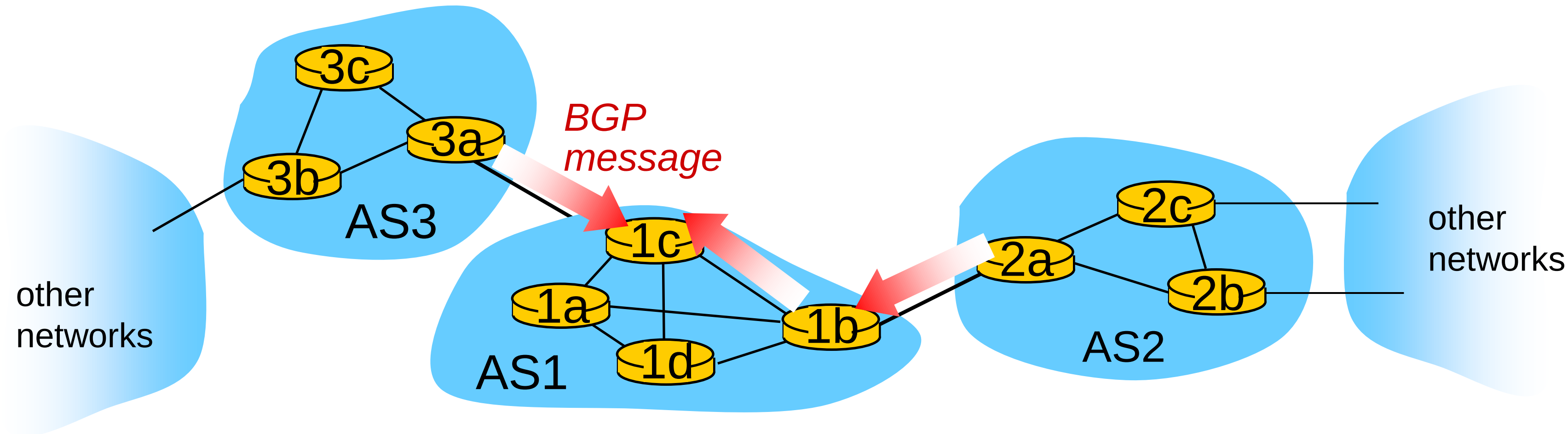
Example – BGP Routing

Router becomes aware of prefix



- BGP message contains "routes"
- "route" is a prefix and attributes: AS-PATH, NEXT-HOP,…
- Example: route:
  - Prefix:138.16.64/22 ; AS-PATH: AS3 AS131 ; NEXT-HOP: 201.44.13.125

Router may receive multiple routes



- Router may receive multiple routes for <u>same</u> prefix
- Has to select one route

Select best BGP route to prefix

Router selects route based on shortest AS-PATH

- Example:

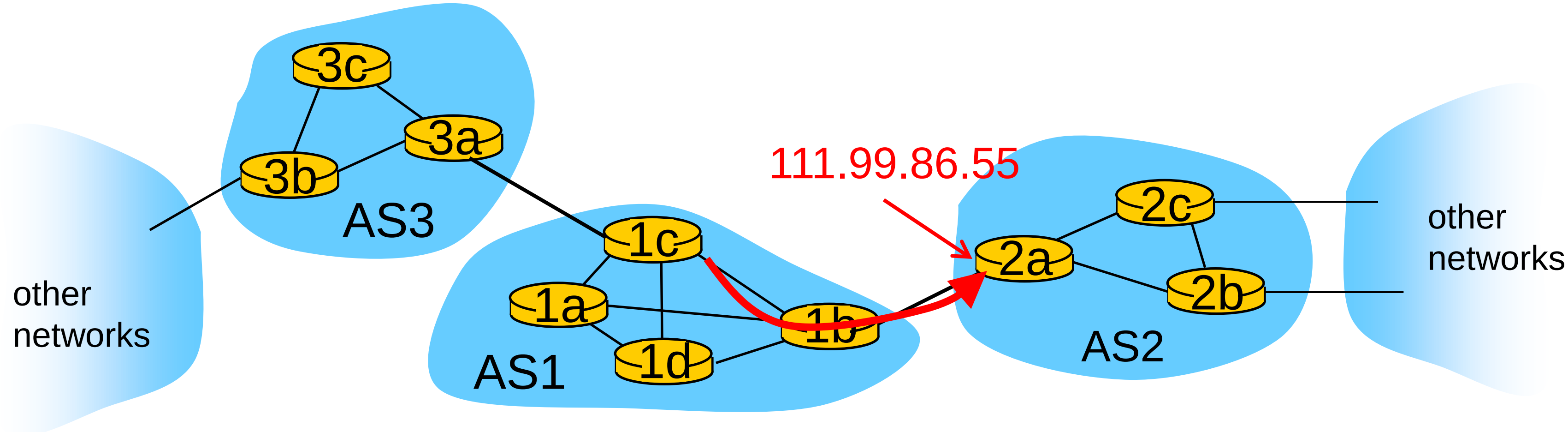
select

  - AS2 AS17 to 138.16.64/22
  - AS3 AS131 AS201 to 138.16.64/22

- What if there is a tie? We'll come back to that!

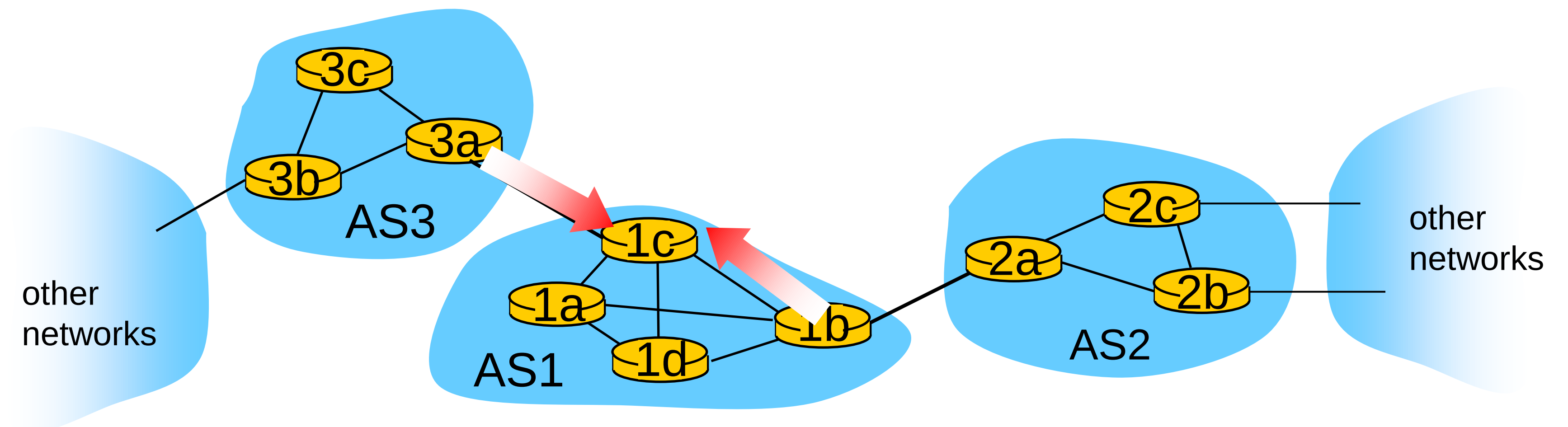
Find best intra-route to BGP route

- Use selected route's NEXT-HOP attribute
  - Route's NEXT-HOP attribute is the IP address of the router interface that begins the AS PATH.
- Example:
  - AS-PATH: AS2 AS17 ; NEXT-HOP: 111.99.86.55
- Router uses OSPF to find shortest path from 1c to 111.99.86.55



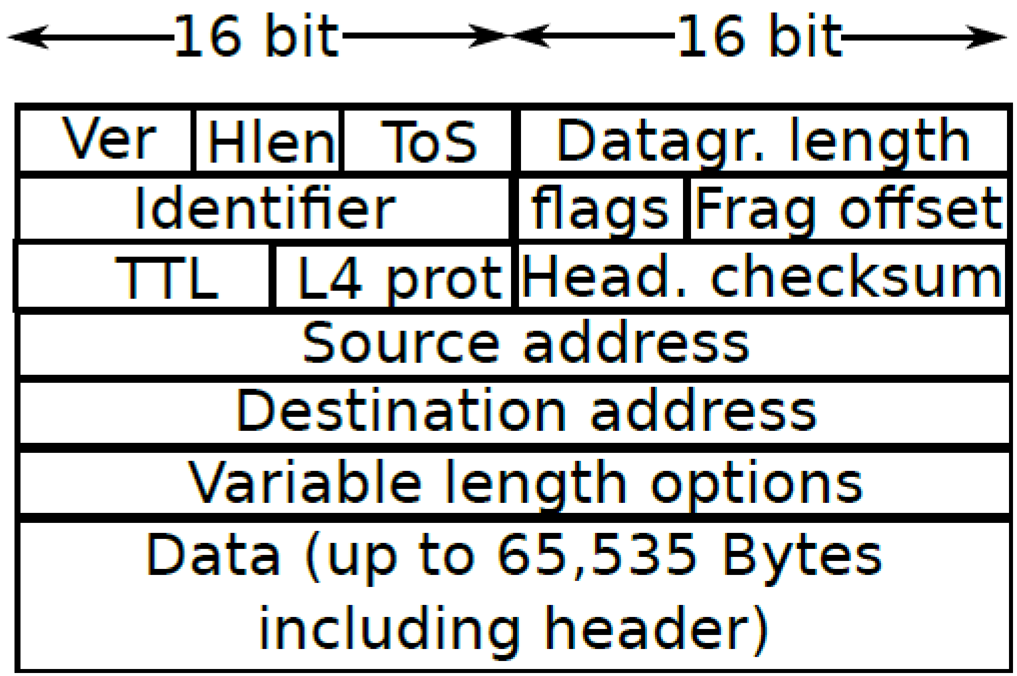
Hot Potato Routing

- Suppose there two or more best inter-routes.
- Then choose route with closest NEXT-HOP
  - Use OSPF to determine which gateway is closest
  - Q: From 1c, chose AS3 AS131 or AS2 AS17?
  - A: route AS3 AS201 since it is closer



The Internet Protocol

IPv4 Format



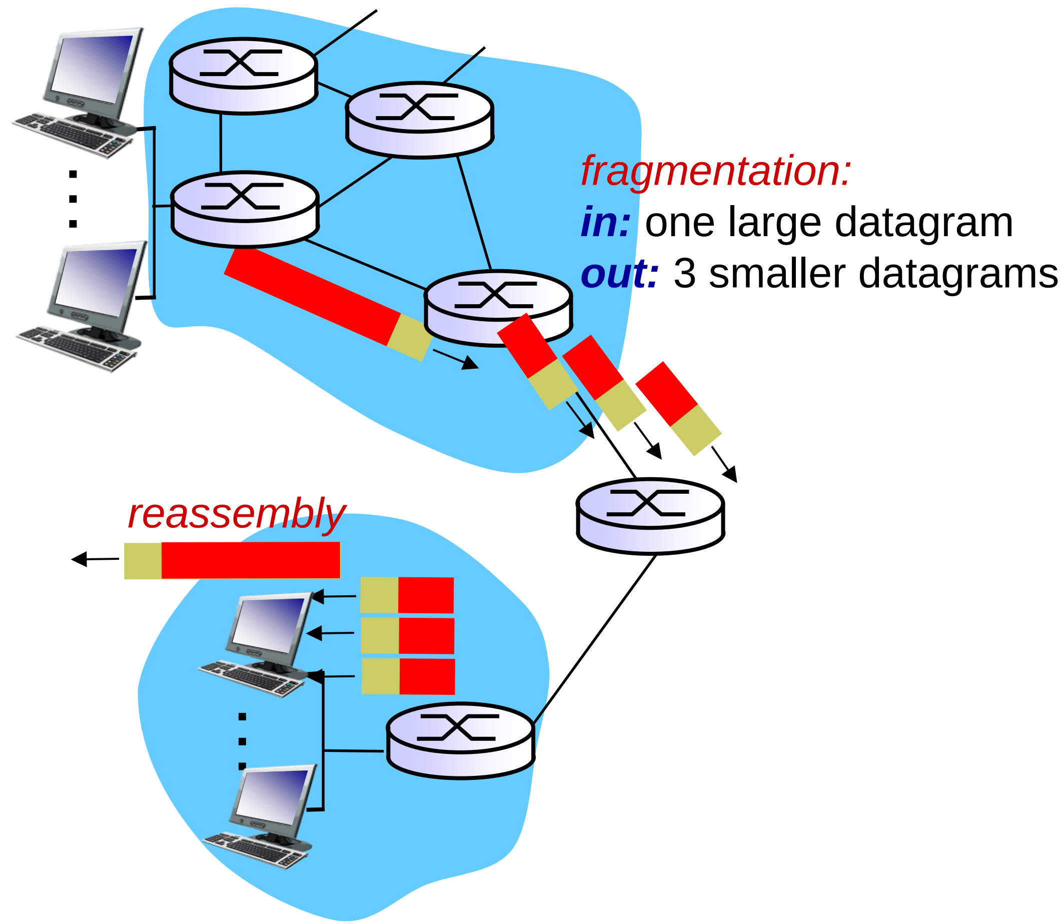
The IP packet format

IP fragmentation

- First layer that single links are considered
- Links can have different maximal transmission unit/size (MTU)
- Transport layer segments might have to be fragmented
- Segments are split into *fragments*, re-assembled at receiver
- Identifier, flags, *Fragment* offset part of IP header used to keep track of network layer fragmentation of single datagram
- MTU is usually lower in link layer (~1.5kB)

IP fragmentation, reassembly

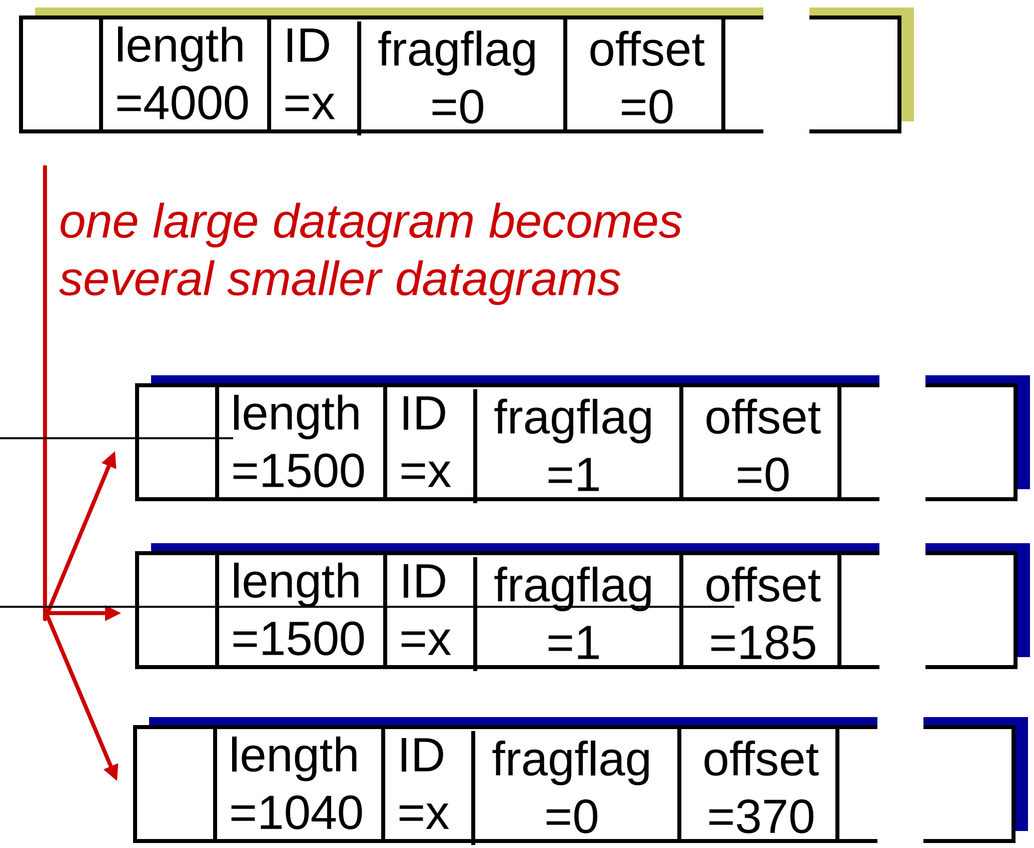
- network links have MTU (max.transfer size) - largest possible link-level frame
  - different link types, different MTUs
- large IP datagram divided ("fragmented") within net
  - one datagram becomes several datagrams
  - "reassembled" only at final destination
  - IP header bits used to identify, order related fragments



IP fragmentation, reassembly

example:

- 4000 byte datagram
- MTU = 1500 bytes



1480 bytes in data field

offset = 1480/8

TTL

- The time-to-live (TTL) field is used to prevent infinite loops
- Its maximum initial value is 255, typical value is 64
- Every hop/router reduces the TTL by 1
- If TTL reaches 0, the packet is discarded
- This prevents a packet from being caught in infinite routing loops

Network Address Translation

Network Address Translation

- Private IP addresses cannot receive traffic from the internet
- How can we enable networks with only private addresses to *connect* to the Internet?

Network Address Translation

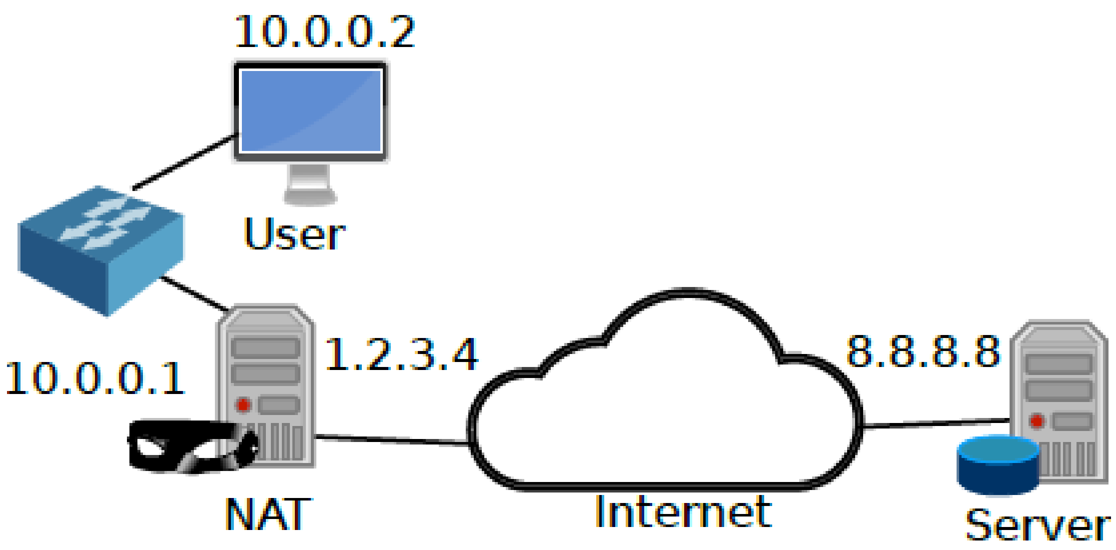
- Private IP addresses cannot receive traffic from the internet
- How can we enable networks with only private addresses to *connect* to the Internet?

- An intermediate gateway-like device with public IP and private IP can forward traffic from private IP users
  - Gateway uses its public IP as source of traffic
  - Gateway uses own source port for connection
- When responses are received by Gateway, it translates back
  - To private source/destination IP
  - To original source/destination port

Network Address Translation (NAT)

- This technique is called NAT, and it is a fundamental aspect of modern networking
- NAT'ing only works for connections initiated from the private IPs
  - Private IPs cannot be directly reached from internet
- The NAT will set up a tuple for each forwarded connection
  - [sourceIP,destIP, sourcePort,destPort, natPort]
- Using this tuple, response messages can be translated back
- NAT'ing can also translate between public networks, but is not often used that way
- NAT'ing works on Transport layer, not Application layer
  - The NAT does not care about application layer protocol

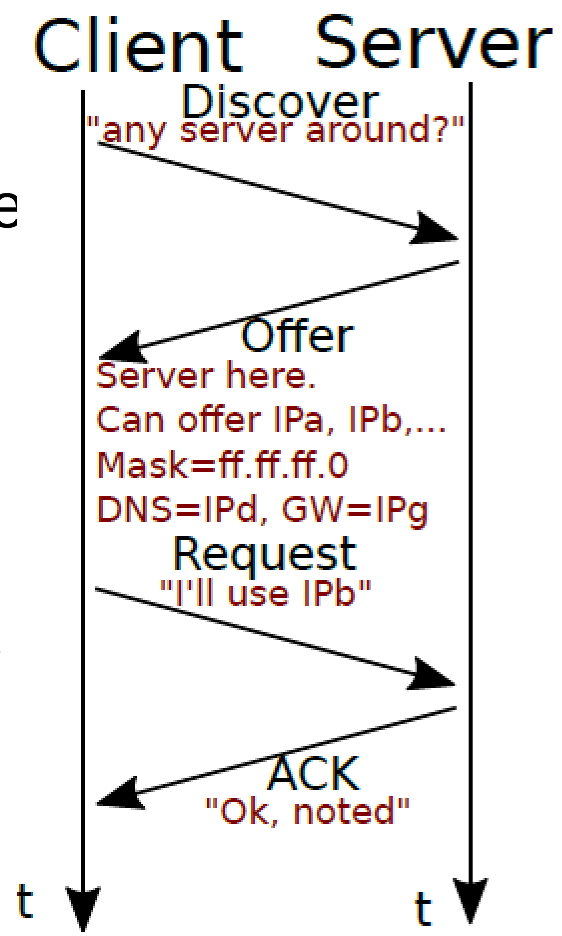
Example NAT Use Case



DHCP

Dynamic Host Configuration Protocol

- More commonly known as DHCP
- Protocol to provide Network Layer and above configuration
- Managed by DHCP server running on the same Link layer *Broadcast domain*
  - As clients do not have an IP address, no sender IP is possible, no gateway is known
  - Four messages are exchanged in total
  - To allow DHCP requests to reach a server in another broadcast domain, *DHCP forwarders* can be used



DHCP Leases

- In the discover message, the client can optionally request a certain IP and a special lease length
- The server can consider these, or not
- The DHCP server offer will include one or more IP addresses, and a lease period
- The client will the choose one, and request it
- Check your currently valid leases like this:
  - If using *dhclient* directly

```
less /var/lib/dhcp/dhclient.leases
```

- If using network-manager

```
grep dhclient /var/log/syslog
```

DHCP Relay

- DHCP relays can be used to extend a DHCP server's range beyond its Link layer broadcast domain
- The DHCP relay will listen for local DHCP discover and request messages
- The DHCP relay will then forward these message to the known IP of the DHCP server
  - The server can reside in different subnet, can be reached via routing
- The server will then send the reply to the relay, relay re-broadcasts the reply
- Typically, the DHCP relay role will be played by the local router/gateway

Activity 8: DHCP

Open the following URL and answer the questions below:

https://www.cloudshark.org/captures/0009d5398f37

1. What is the application that launches the DHCP request?
2. What are the source and destination port numbers for the request? What processes do these ports correspond to?
3. What are the source and destination IP addresses of the corresponding network layer datagram?
4. What are the source and destination MAC addresses of the corresponding link layer frame?
5. Submit on eDimension

Ref: https://www.quora.com/Why-would-DHCP-be-considered-a-layer-7-application-layer-protocol-when-the-sole-purpose-of-the-DHCP-is-to-manage-IP-addresses

Conclusion

- Two main classes of low-cost path finding algorithms:
  - Link State (Dijkstra's)
  - Distance Vector (Bellman-Ford)
- LS is handling negative changes better, but does not scale
- DV scales much better, but has problems handling link loss/increased cost
- RIP is an Intra-AS DV protocol
- OSPF is an Intra-AS LS protocol
- BGP is an Inter-AS Path Vector protocol
  - Cost is "number of AS" on path
- The IP protocol itself is quite simple
  - IP addresses + possible fragmentation