Chapter 4 - Network Layer

Wednesday, March 23, 2016 2:12 PM

Network Layer provides host-to-host communication
- ↳ unlike transport & application layers all hosts and all routers use network layer protocols
- ↳ one of most complex protocol stack layers

IP (internet protocol)
- ↳ most famous network layer protocol
- ↳ best-effort/unreliable service

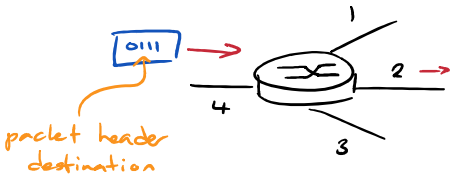
→ all hosts have IP address (network layer address)

NETWORK LAYER FUNCTIONS

Routing → determine route for packets on network-wide scale
→ source to destination

Forwarding → determine route for packet from one link to next link
→ contained within one router from input to output
↳ tables determined by routing algorithms

Internet protocol
Report datagram errors



FORWARDING TABLE	
header	output link
0100	3
0101	2
0111	2
1001	1

data is forwarded from input to output link using PACKET-SWITCHES

type 1. link-layer switches (layer 2 device, forward using frame data)

type 2. routers (layer 3 device, forward using datagram data)

Network service model

possible services: guaranteed delivery
bounded delay
in-order delivery
minimum bandwidth
maximum jitter (variation in time b/w packets)
security services

	Guaranteed Bandwidth	No-loss Guarantee	In-order Guarantee	Timing Maintained	Congestion Control	
IP	None	None	None	No	None	← minimal
CBR	Constant	Yes	Yes	Yes	Yes	
ABR	Guaranteed minimum	None	Yes	No	Indication provided	

→ available bit rate
→ constant bit rate
} network services

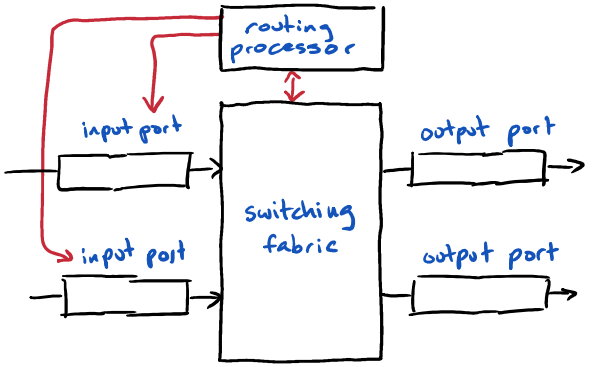
CONNECTION, CONNECTIONLESS SERVICE

→ virtual circuit network (VC)
→ provides network layer connection service
→ evolved from telephony
→ complexity managed inside network

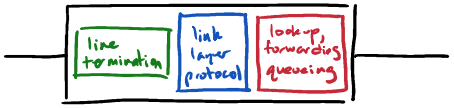
→ datagram network
→ IP is an example
→ connectionless
→ no setup or state maintained
→ complexity managed at hosts

INSIDE A ROUTER

↳ forwarding table maps prefix of packet's destination address to output link
↳ if multiple matches, router uses longest prefix (in bits)

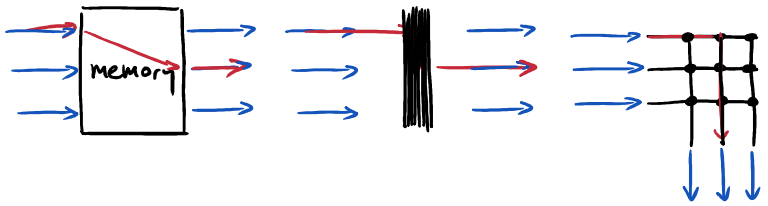


Input ports
↳ where lookup is performed
↳ updated by routing processor
↳ performs queueing if congested
↳ ie. head-of-the-line blocking
if fabric slower than input ports combined



Switching fabric
↳ transfers packets from input buffer to output

ex. memory bus crossbar

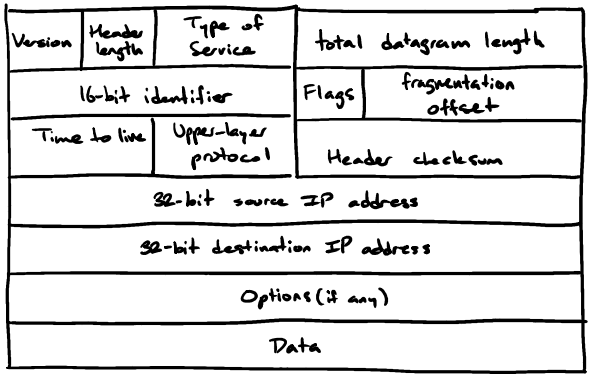


Output ports
↳ provides buffering when data arrives faster than it can be sent (transmission rate)

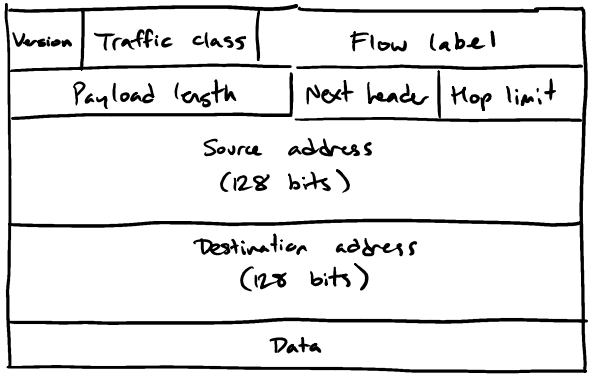
Routing processor
↳ executes routing protocols, updates forwarding table

THE INTERNET'S SWEETHEART, IP

IPv4 datagram format



IPv6 datagram



Version → IP version #

* Header length → variable in IPv4 b/c options → fixed to 40-bytes in IPv6
→ if no options (typically) 20-byte

Type of service → diff. types need high reliability, low delay, high throughput, etc. → similar to IPv6 traffic class

Datagram length → header plus data, max is $2^{16}-1$, typically ~1500 bytes → only payload length in IPv6

* Identifier, flags, frag. offset → fragmentation discontinued in IPv6
↳ used for fragmentation

* Checksum

TTL → # hops before packet dropped, max 255 → similar to Hop Limit
→ decremented by one at each router

Upper-layer protocol → similar to Next Header
↳ indicates if should be passed to UDP or TCP, etc.
↳ analogous to port # field in transport

* Header checksum
↳ detects bit-errors

Source/destination

* Options → rarely used

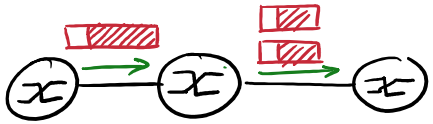
Data → usually contains TCP or UDP segment
→ sometimes carries ICMP messages

* discontinued in IPv6

IP FRAGMENTATION & REASSEMBLY

↳ network links have maximum transfer size, MTU

if datagram length > MTU, datagram must be fragmented
- ↳ router breaks up datagram into smaller packets
- ↳ reassembly is left for the recieving end-system



typically each datagram sent gets an incremented "identification number"
- ↳ fragmented datagrams get same identification number
- ↳ fragmentation flag set to 1 except for final datagram
- ↳ fragment offset identifies order to reconstruct
  - ↳ offset = $\frac{\text{amount of bytes already sent}}{8}$

Ex) 4000 Bytes datagram must be sent through 1500 Byte MTU link
original identification number is 30. what are the # Bytes, ID, Offset, and Flag of fragmented datagrams?

$$\#\text{ fragments} = \left\lceil \frac{4000 \text{ bytes}}{1500 - 20 \text{ bytes}} \right\rceil = 3$$

each fragment needs 20 bytes for header, can only transmit 1480 bytes in payload

fragment #	bytes of data	ID	flag	offset
1	1480	30	1	$\frac{0}{8} = 0$
2	1480	30	1	$\frac{1480}{8} = 185$
3	1020	30	0	$\frac{1480 + 1480}{8} = 370$

original amount of data = 4000-20

stays same

final flag = 0

IPv6 doesn't support fragmentation and it is generally not recommended anymore. It makes end systems vulnerable

by sending overlapping/misaligned fragments or those without a zero offset that the host can't reassemble and may cause it to crash. (Jolt2)

↳ instead of fragmenting, recommended practice is to drop

IP Addressing

↳ boundary b/w host & link and router & link is called an interface

↳ routers have multiple interfaces, hosts typically have only one

→ IP address are associated with interface

dotted-decimal notation

xxx . xxx . xxx . xxx ← 4 bytes

byte byte byte byte

0-255 0-255 0-255 0-255

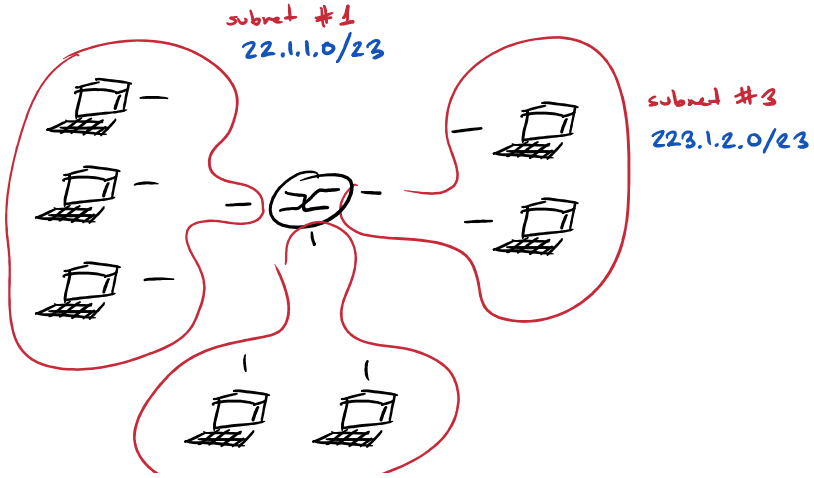
↳ IPv4 gives 32 bits, $2^{32} \simeq 3$ billion possible addresses

↳ depending on the class, the first N bits of an IP address are devoted to the interfaces subnet

↳ interconnecting hosts/routers form subnet

↳ N defined by subnet mask, ex.

223.1.1.0/23 ← first 23 bits define subnet address



subnet #2

223.1.3.0/23

a.b.c.d/N format is called CIDR

↳ classless inter-domain routing

Ex) 200.23.16.0/23

↳ 11001000.00010111.00010000.00000000

23 bits subnet part

9 bits host part

only this part is considered by routers outside subnet

there may even be additional subnets within this portion

→ Before CIDR, class A = /8
class B = /16
class C = /24

→ IP addressed globally managed by ICANN
↳ they provide blocks of IPs to ISPs who can subsequently manage those blocks
↳ assigning domain names

DHCP, Dynamic Host Configuration Protocol

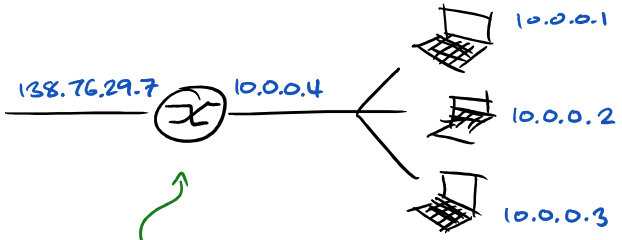
↳ automatically assigns IPs within subnet as people move in and out of networks

↳ client-server protocol, arriving hosts request IP from DHCP server to join network

↳ 1. sends DHCP discovery message to 255.255.255.255 from 0.0.0.0
2. DHCP server offers IP to 255.255.255.255
3. client request that IP to 255.255.255.255
4. server ACKs request

NAT (network address translation)

→ allows local network to publically use one IP
→ within network each host gets a private IP
→ traffic b/w inside and outside goes through one router that changes IPs and maps internal hosts to specific ports
→ provides privacy for network ↳ 16 bits for port ∴ max ~66000 hosts
→ provides additional IPs



outgoing datagrams

src	10.0.0.1, 3345	→	src 138.76.29.7, 5001
dest	outside world		dest outside world

incoming datagrams

src	outside world	→	src outside world
	138.76.29.7 5001		

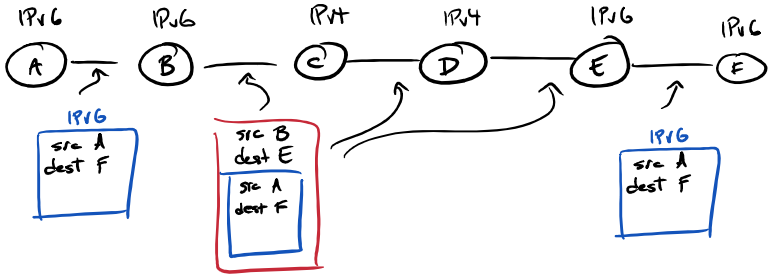
dest 10.0.0.1, 3345

→ NAT is widely used but controversial
  → port #s not meant for host addressing
  → routers not supposed to process packets at layer 4, ports
  → violates end-to-end argument, ie. direct communication
  → IPv6 should be used to solve IP address shortage

→ also adds overhead

ICMP - internet control message protocol ↳ network layer information
  → mainly for hosts & routers to send error messages
  → traceroute program uses ICMP

Tunneling
  ↳ used to transfer IPv6 through IPv4 routers
  ↳ encapsulate IPv6 datagram with IPv4

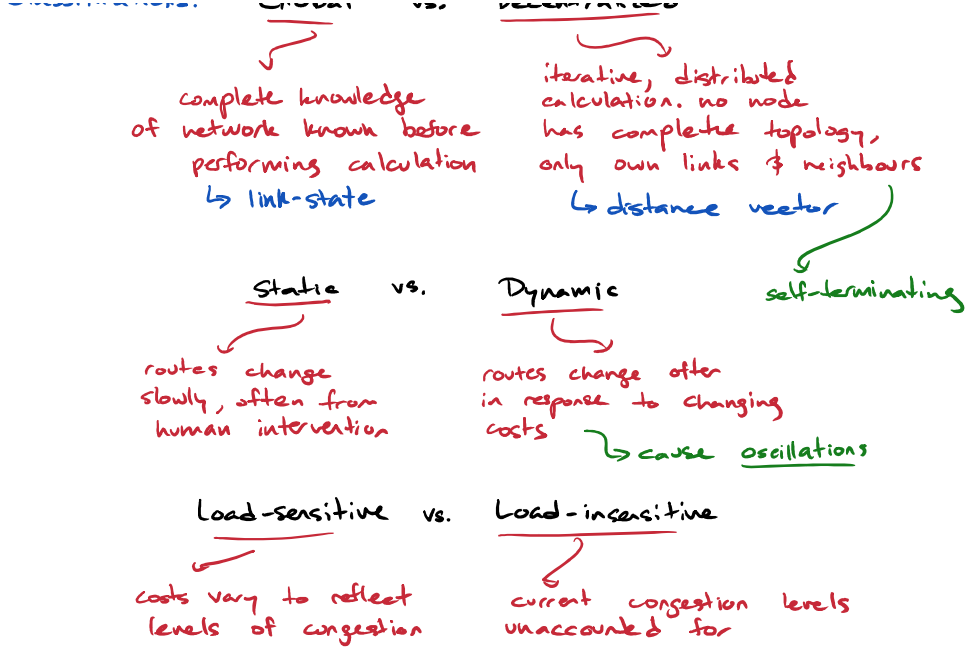


ROUTING ALGORITHMS

→ determines end-to-end path through network
→ creates routers' forwarding tables
→ main network-layer job

→ goal: find lowest cost path from src to destination
  ↳ can be link length, speed, or literal monetary cost

Classifications: Global vs. Decentralized
  ↳ complete knowledge
  ↳ iterative, distributed



Link State

↳ global knowledge of network required

↳ each node broadcasts its connection costs to all other links

↳ each node has complete & identical view of network

→ then each node runs link-state algorithm to determine their forwarding tables

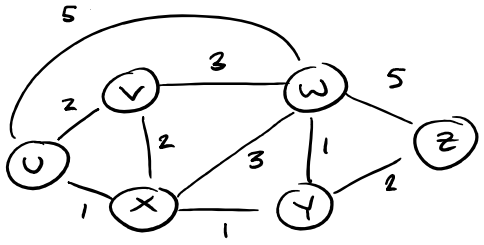
DIJKSTRAs ALGORITHM, $O(n^2)$

$D(v)$ → least cost path cost from source to v
$p(v)$ → previous node along current least-cost path to v
N' → subset of nodes investigated
$c(x,y)$ → cost from x to y

```
For source node, u
  //initialization
  N' = {u}
  for v in all nodes:
    if v is a neighbour of u:
```

```
        D(v) = c(u,v)
    else
        D(v) = ∞

// rest
while N' != N:
    find w not in N' with lowest D(w)
    add w to N'
    for each neighbour, v, of w
        if v not in N'
            D(v) = min( D(v), D(w) + c(w,v) )
            //new cost is either old cost to v
            // or known least path cost to
            // w plus cost of w to v
```

Ex) Compute forwarding table for u using Dijkstra's algorithm



=

N'	D(v) p(v)	D(x) p(x)	D(w) p(w)	D(y) p(y)	D(z) p(z)
u	2,u	1,u	5,u	∞	∞
ux	2,u		4,x	2,x	∞
uxy	2,u		3,y		4,y
uxyv			3,y		4,y
uxyvw					4,y
uxyvwz					

D(x) was lowest, so add x to N'

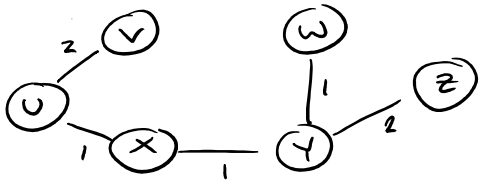
ex. $D(w) = \min(D(w), D(x) + c(x,w))$
$= \min(5, 1+3)$
$= 4$

lowest cost of previous line

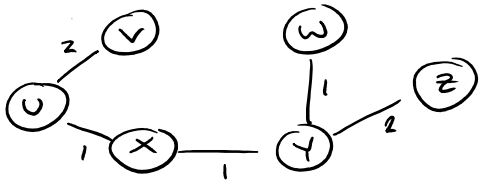
each row should only have the p() of something in that row's N'

Forwarding table of u

destination	link
x	(u,x)
y	(u,x)
v	(u,v)
w	(u,x)
z	(u,x)



destination	link
x	(v, x)
y	(v, x)
v	(v, v)
w	(v, x)
z	(v, x)



Distance Vector

↳ nodes don't know entire topology, just their own costs and the costs of their direct neighbours

↳ asynchronous, distributed, iterative, self-terminating

↳ nodes recieve information from direct neighbours, perform calculation, distribute result to neighbours

↳ ex. RIP, BGP, ISO IDRP, ARPAnet

Each node knows:

- cost to each neighbour
- own distance vector
  ↳ estimate of cost to all destinations in network
- distance vectors of immediate neighbours

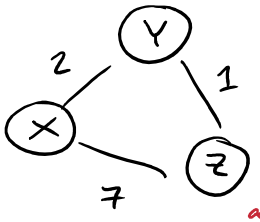
↳ if there are changes, each node updates its neighbours, who recalculate and update their neighbours and this ripples out as everyone recalculates until everyone's stable

DV ALGORITHM for each node, x

```
//initialization
for all destinations, y, in N
    Dx(y) = c(x,y)

for each neighbour, w
    for all destinations, y, in N
        Dw(y) = ?

for each neighbour, w
    send distance vector Dx = [Dx(y) : y in N] to w
```

```
//loop
while true
    wait until link cost change or
          until updated distance vector from neighbour, w

    for each neighbour, y, in N
        Dx(y) = min_v { c(x,v) + Dv(y) }

    if Dx(y) changed for any destination, y
        send distance vector D⃗x = [Dx(y) : y in N] to neighbours
```

Ex)



Show steps in how these nodes would compute their routing tables

=

initial

Node X

from \ cost to	x	y	z
x	∅	2	7
y			
z			

Node Y

from \ cost to	x	y	z
x			
y	2	∅	1
z			

Node Z

cost to	x	y	z

after each node has shared info and calculated distance vector

Node X

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

only recalculates own vector

Node Y

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

Node Z

cost to	x	y	z

after X & Z reshare info b/c changes

Node X

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

because own distance vector change, it would rebroadcast

instead of z to x (7) it can go z to y, y to x (2+1=3)

Node Y

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

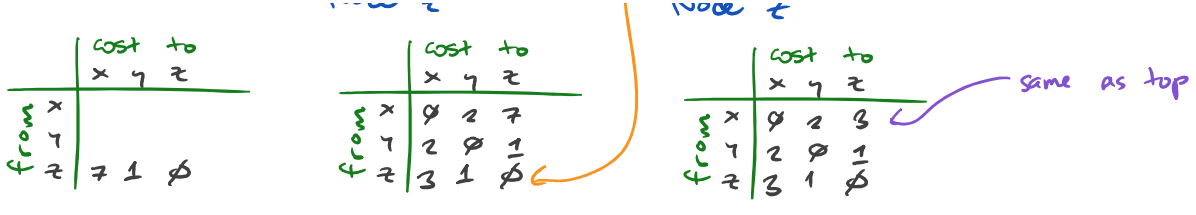
own vector didn't change so wouldn't rebroadcast

note symmetry

Node Z

cost to	x	y	z

same as top



=

DV & link cost changes (routers break, go offline, etc.)

↳ decreased cost is easy & recomputation fast
"good news travels fast"

↳ increased cost causes a problem where links with contradictory information forward the packet in a routing loop

↳ slowly recomputed b/c of TTL

"bad news travels slow"

↳ fixed with "Poisoned Reverse"

→ after forwarding a datagram a router lies to the recieving router and tells it the cost to get to the destination from the forwarding router is ∞

→ ensures recieving router will not forward datagram back to sending router

LS vs. DV

DV → communication only b/w adjacent routers

LS → communication b/w all routers

speed of convergence

LS → oscillations, slow
DV → possible routing loops
→ possible slow recomputation

robustness

LS → router can broadcast incorrect link cost
↳ only computes own table if mistake
DV → router can broadcast incorrect path cost
↳ mistake ripple through entire network

HIERARCHICAL ROUTING

↳ internet isn't flat network, it is hierarchical network of networks

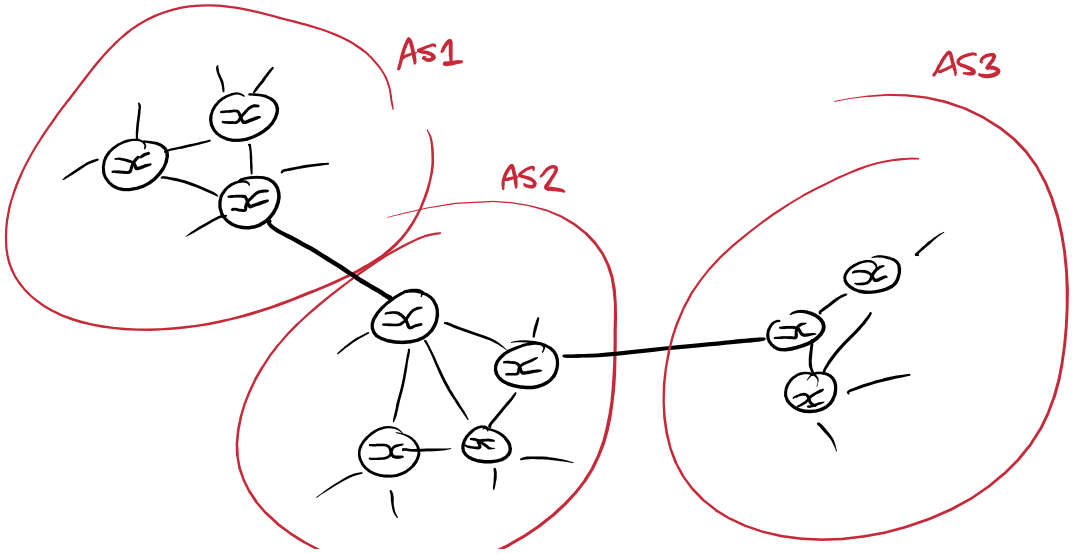
↳ Autonomous Systems (AS) managed under same administrative control (ISP, company, etc.)

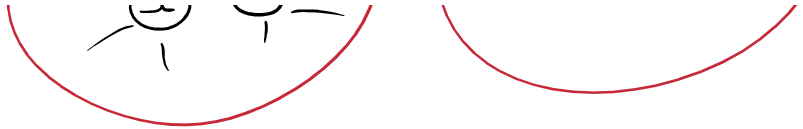
↳ different subnetworks use different routing protocols

↳ intra-AS routing protocols

↳ the scale is huge, 600 million hosts
↳ can't store all in routing tables
↳ routing algorithms would never terminate

↳ ASs connected by gateway routers





Inter-AS routing protocols

↳ ie. which AS to send information through to other network

→ usually simpler than intra-AS routing

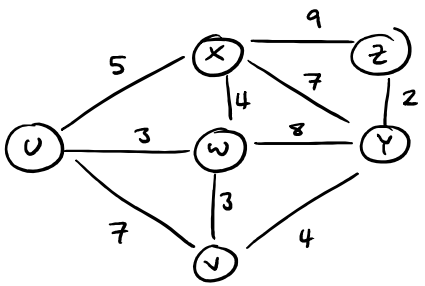
→ often "hot-potato" routing

↳ send to gateway route that has the lowest intra-AS cost

↳ move packet as inexpensively as possible

→ usually policy dictates more than efficiency

Ex)

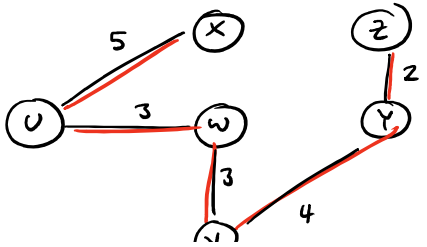


Find least cost paths of u using Dijkstra's algorithm

=

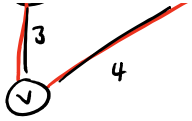
N'	D(x),p(x)	D(v),p(v)	D(w),p(w)	D(y),p(y)	D(z),p(z)
u	5,u	7,u	3,u	∞	∞
uw	5,u	6,w		11,w	∞
uwx		6,w		11,w	14,x
uwxv				10,v	14,x
uwxvy					12,y
uwxvyz					

highlight all routes used and erase all not used



forwarding table of u

destination	link
x	(u,x)
w	(u,w)
v	(u,w)
y	(u,w)



x	(v,x)
u	(v,u)
v	(v,v)
y	(v,w)
z	(v,w)

Ex) a class c subnet can accomodate how many hosts?

class C is /24, meaning first 24 bits devoted to subnet addressing.

32 - 24 bits = 8 bits remaining for hosts

2^8 possible addresses, 256

however, "255" and "0" are reserved for broadcast & network

∴ **254 addresses**

$2^0 = 1$
$2^1 = 2$
$2^2 = 4$
$2^3 = 8$
$2^4 = 16$
$2^5 = 32$
$2^6 = 64$
$2^7 = 128$
$2^8 = 256$

Ex) The CIDR subnet mask for a network is /24, what is the subnet mask in dotted decimal notation?

In dotted-decimal, the subnet bits are all set to 1 and the host bits are set to 0.

↳ 11111111 11111111 11111111 00000000

↳ **255.255.255.0**

Ex) Consider IP subnet with prefix 129.17.129.96/27.

a) Provide range of IP addresses (in D.D. notation) that can be assigned to this subnet.

b) Suppose an organization wants to create 4 subnets (of same size) within this block. What are the prefixes of the four subnets?

==

$96 = 64 + 32 = 2^6 + 2^5$

00000000 00000000 00000000 01100000 — 96

belong to subnet

01111111 — $96+1+2+4+8+16 = 127$

a) 129.17.129.96 → 129.17.129.127

need to index an additional four subnets w/ equal number of hosts

↳ for 4 subnets, prefix needs 2 additional bits, ie. /29

Last 8 bits — hosts from 000 to 111

.... 01100000 subnet 1 129.17.129.96 → 129.17.129.103 (96+7)

initial subnet; sub-subnet, 00→11

01101000 subnet 2 129.17.129.104 → 129.17.129.111 (103+7)

01110000 subnet 3 129.17.129.112 → 129.17.129.119

01111000 subnet 4 129.17.129.120 → 129.17.129.127

Ex) Consider datagram using 8-bit host addresses, suppose router uses longest prefix-matching w/ given forwarding table.

Ex) Consider datagram using 8-bit host addresses, suppose router uses longest prefix-matching w/ given forwarding table.

Give associated range of destinations & host addresses.

Prefix Match	Interface
00	0
010	1
011	2
10	2
11	3

=

00 gets 0000 0000 → 0011 1111 → 64 addresses
010 gets 0100 0000 → 0101 1111 → 32 addresses
011 gets 0110 0000 → 0111 1111 } 96 addresses
10 gets 1000 0000 → 1011 1111 }
11 gets 1100 0000 → 1111 1111 → 64 addresses

↑ check for no overlap in any ranges, checks out

Ex) Consider a router connecting 3 subnets. Suppose all subnet interfaces must have prefix 223.1.17/24. Also suppose subnet 1 must get at least 60 interfaces, subnet 2 must get at least 90 interfaces, and subnet 3 must get at least 12.

Provide 3 network addresses that satisfy these conditions.

=

3 subnets, ∴ identify with 2 additional prefix bits 00, 01, 10 (11 spare)

0000 0000
6 left for hosts

don't have room. identify w/ just prefix of 1

subnet 1 must have 60, so give 6 bits (64)
use additional 2 bits for subnet prefix

00 00 0000 → 0011 1111

↳ 223.1.17.0/26

subnet 2 must have 90, so give 7 bits (128)
use additional 1 bit for subnet prefix

1000 0000 → 1111 1111

↳ 223.1.17.128/25

subnet 3 must have 12, so give 4 bits (16)
use additional four bits for subnet prefix

↳ for subnet prefix use 01 so it doesn't overlap w/ 00 (subnet 1) or 1 (subnet 2)

0100 0000 → 0100 1111

↳ 223.1.17.64/28

Ex) a) What does it mean that NAT allocates private IP addresses?

b) What does a NAT router look like to the rest of the world?

c) How does a NAT router obtain an IP and how does it provide IPs to the hosts within private networks?

=

a) they're private in that they don't have meaning outside the NAT network

b) it looks like a single host with one IP

c) most likely they are both assigned with DHCP.
↳ for the NAT router, probably the ISP's DHCP server
↳ for the hosts in the NAT, probably the NAT's own DHCP server

Ex) Consider a datagram network using 32-bit host addresses. Suppose a router has four links, numbered 0 through 3, and packets are to be forwarded to the link interfaces as follows:

Destination Address Range	Link Interface
11100000 00000000 00000000 00000000 through 11100000 00111111 11111111 11111111	0
11100000 01000000 00000000 00000000 through 11100000 01000000 11111111 11111111	1
11100000 01000001 00000000 00000000 through 11100001 01111111 11111111 11111111	2
otherwise	3

a. Provide a forwarding table that has five entries, uses longest prefix matching, and forwards packets to the correct link interfaces.

=

prefix	output link
11100000 00	0
11100000 0100 0000	1
11100000	2
11100001 1	3
otherwise	3

=

RIP, Routing Information Protocol

→ distance vector algorithm
→ cost measured in # hops
→ maximum allowable hops = 15
→ distance vectors rebroadcasted every 30s
↳ if no advertisement heard after 180s, router assumed dead

with UDP

→ new DV advertised

→ implements poison reverse to prevent ping-pong loops

→ routing tables managed at application level

OSPF (Open Shortest Path First)

→ link-state algorithm (Dijkstra)

→ advertisements only use IP (not TCP/UDP)

→ adds security over RIP

↳ messages authenticated so false advertisements protected against

→ multiple same cost paths allowed (multipath)

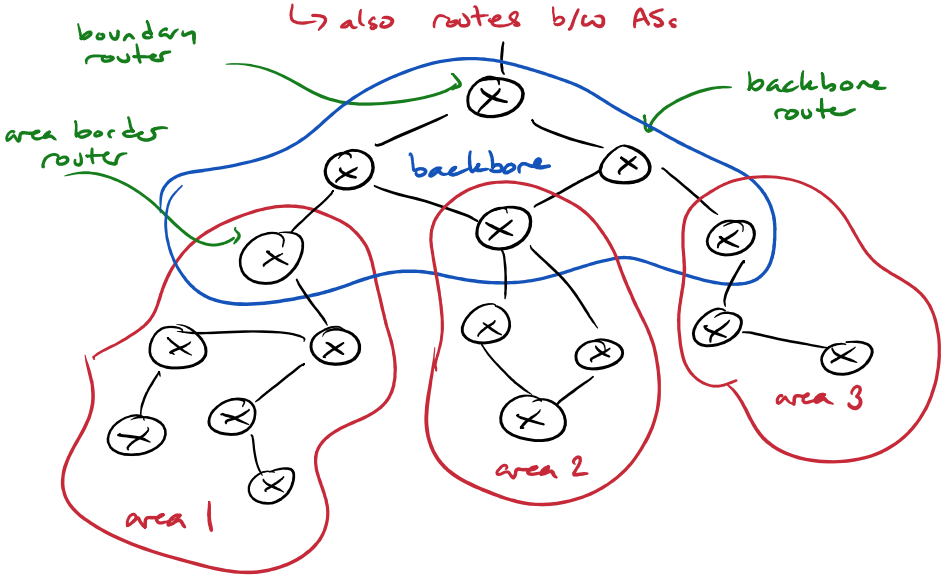
→ hierarchical

↳ broken into areas, which each compute own OSPF link-state routing algorithm

↳ areas connected by one "backbone area"

↳ backbone area routes traffic b/w others

↳ also routes b/w ASs



BGP, Border Gateway Protocol

→ inter AS routing protocol
→ provides ASs information about neighbouring ASs reachability, existence
→ uses semi-permanent TCP connections b/w ASs

→ largely determined by policy over efficiency
↳ ie. if two different ASs are communicating, you don't want to route for them if you aren't getting anything out of it