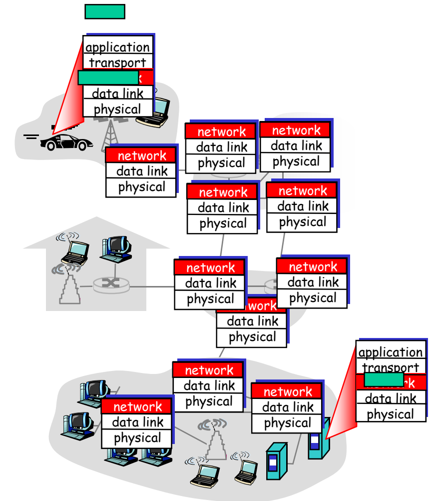
Network Layer

Computer Networks

Network layer

- transport segment from sending to receiving host
- on sending side encapsulates segments into *datagrams*
- on receiving side, delivers segments to transport layer
- network layer protocols in *every* host, router
- Router examines header fields in all IP datagrams passing through it

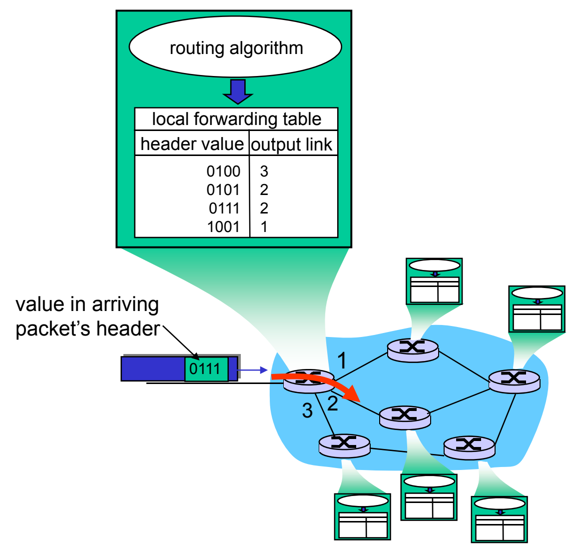


Two Key Network-Layer Functions

- *Forwarding Function:* move packets from router's input to appropriate router output
- *Routing Function:* determine route taken by packets from source to destination
  - *Routing Algorithms*

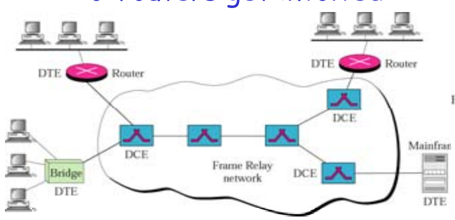
- analogy:
- routing: process of planning trip from source to destination
- forwarding: process of getting through single interchange

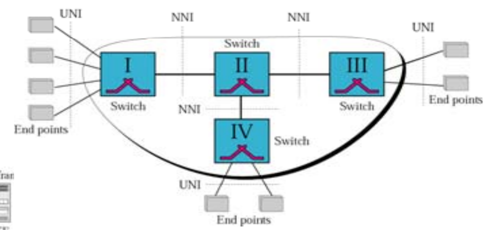
Interplay between routing and forwarding



Connection setup

- 3rd important function in *some* network architectures:
  - Asynchronous Transfer Mode (ATM), Frame Relay, X.25
  - Backbone networks in Internet are Switched WAN
  - Switched WAN : wide area network that cover large area and provide access at several point to users
- before datagrams flow, two end hosts *and* intervening routers establish virtual connection
  - routers get involved





DTE : Data Terminating Equipment
DCE : Data Circuit Equipment

UNI : User-to-Network Interface
NNI : Network-to-Network Interface

Network layer connection and connection-less service

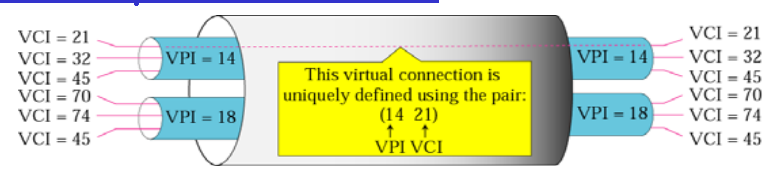
- Network layer Connection service
  - Virtual Circuit
    - Connection setup
    - Forwarding
    - Routing
- Network layer Connection-less service
  - Datagram network
    - Forwarding
    - Routing
- Analogous to transport-layer services, but:
  - service: host-to-host
  - no choice: network provides one or the other
  - implementation: in network core

Virtual circuits (VC)

> "source-to-dest path behaves much like telephone circuit"
> - performance-wise
> - network actions along source-to-dest path

- call setup, teardown for each call *before* data can flow
- each packet carries VC identifier (VCI) (not destination host address)
- *every* router on source-dest path maintains "state" for each passing connection
- link, router resources (bandwidth, buffers) may be *allocated* to VC (dedicated resources = predictable service)

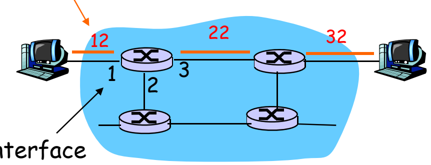
VC implementation



- VC consists of:
  - Path from source to destination (Virtual Path: VP)
  - VC numbers, one number for each link along path
  - entries in forwarding tables in routers along path
- packet belonging to VC carries VC number (rather than destination address)
- VC number can be changed on each link.
  - New VC number comes from forwarding table

Note that : virtual connection is defined by a pair of numbers: VPI and VCI.

Forwarding table



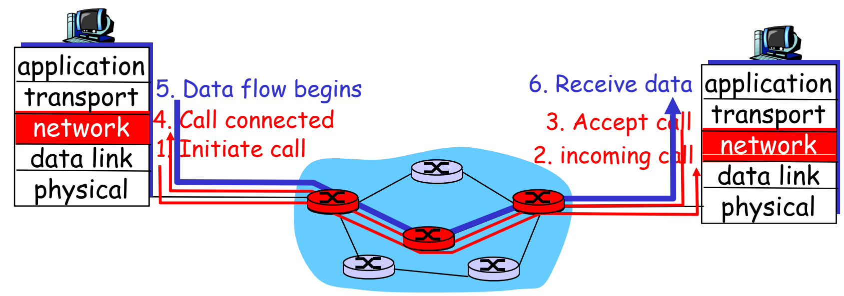
Forwarding table in northwest router:

Incoming interface	Incoming VC #	Outgoing interface	Outgoing VC #
1	12	3	22
2	63	1	18
3	7	2	17
1	97	3	87
…	…	…	…

Routers maintain connection state information!

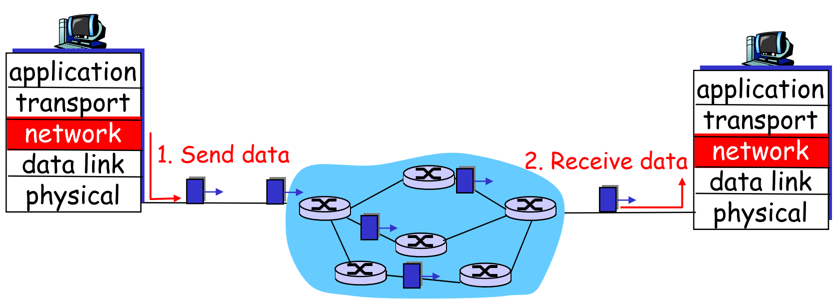
Virtual circuits: signaling protocols

- used to setup, maintain teardown VC
- used in ATM, Frame-Relay, X.25
- not used in today's Internet



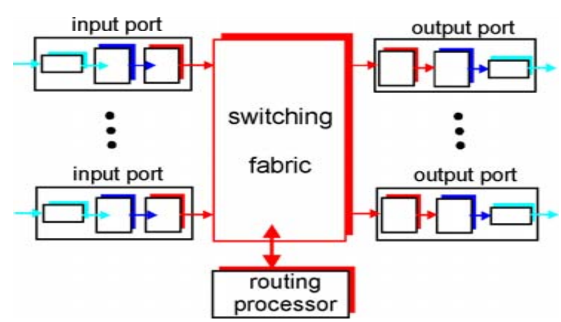
Datagram networks

- no call setup at network layer
- routers: no state about end-to-end connections
  - no network-level concept of "connection"
- packets forwarded using destination host address
  - packets between same source-dest pair may take different paths

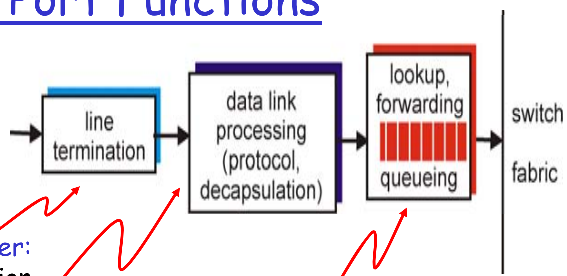


What is inside the router? : Router Architecture Overview

- Two key router functions:
- run routing algorithms/protocol
  - Route Information Protocol (RIP),
  - Open Shortest Path First (OSPF),
  - Border Gateway Protocol (BGP)
- *forwarding* datagrams from incoming to outgoing link



Input Port Functions

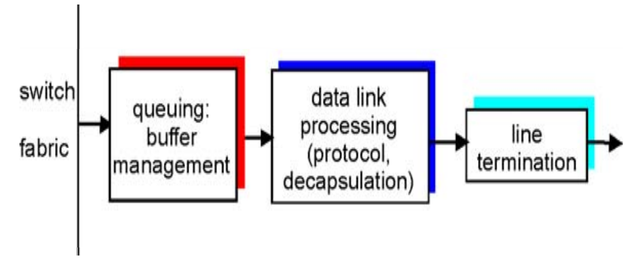


Physical layer:
bit-level reception

Data link layer:
e.g., Ethernet

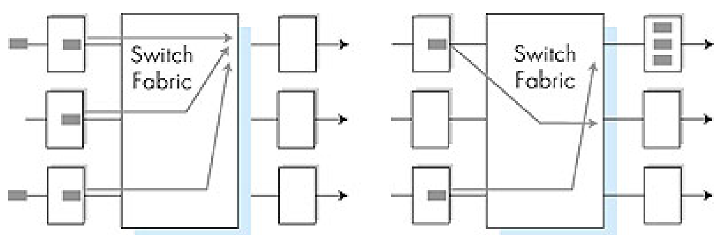
- **Decentralized switching**:
- given datagram dest., lookup output port using forwarding table in input port memory
- Goal: complete input port processing at 'line speed'
- Queuing: if datagrams arrive faster than forwarding rate into switch fabric

Output Ports



- *Buffering* required when datagrams arrive from fabric faster than transmission rate
- *Scheduling discipline* chooses among queued datagrams for transmission

Output port queueing



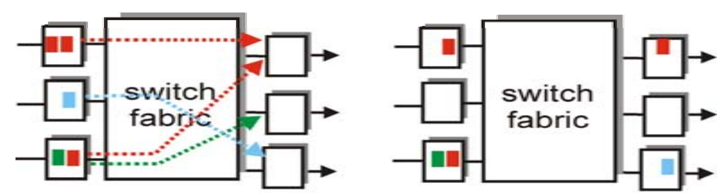
Output Port Contention at Time *t*

One Packet Time Later

- buffering when arrival rate via switch exceeds output line speed
- *queueing (delay) and loss due to output port buffer overflow!*

Input Port Queuing

- Fabric slower than input ports combined -> queueing may occur at input queues
- Head-of-the-Line (HOL) blocking: queued datagram at front of queue prevents others in queue from moving forward
- *queueing delay and loss due to input buffer overflow!*



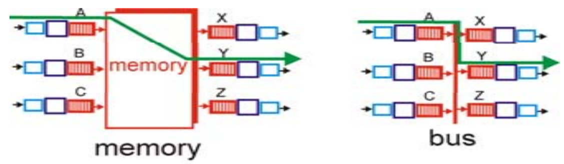
output port contention at time t - only one red packet can be transferred

green packet experiences HOL blocking

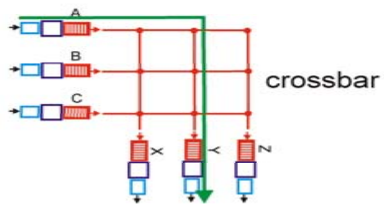
Three types of switching fabrics

- Packet was copied from input port into processor memory
- routing processor extracted destination address, looked up appropriate output port in forwarding table
- Cisco's Catalyst 8500 series switches (10 Gbps)

- Input port transfer packet directly to output port over shared bus
- One packet at a time can be transferred over bus
- switching bandwidth of router is limited to bus speed
- Cisco 5600 – switches packets over 32 Gbps bus

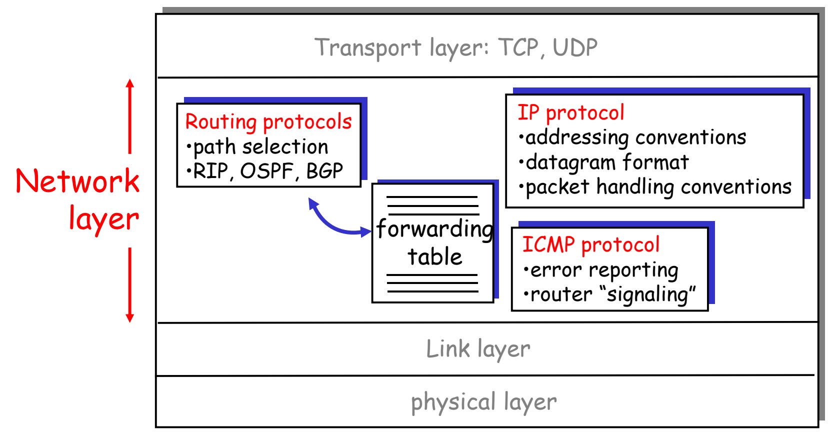


- Crossbar switch is interconnection network Consisting of 2n buses
- Cisco 12000 family switches use interconnection network, providing up to 60 Gbps through switching fabric

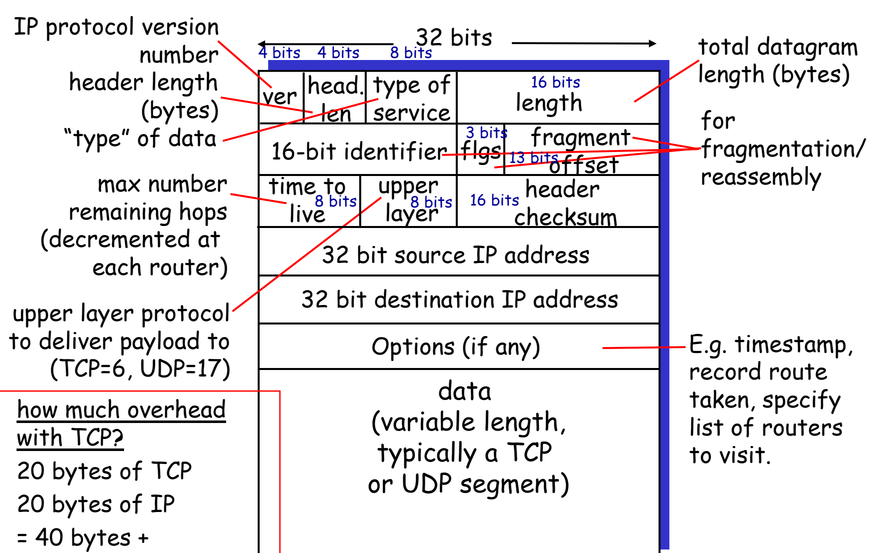


The Internet Network layer

Host, router network layer functions:



IP datagram format

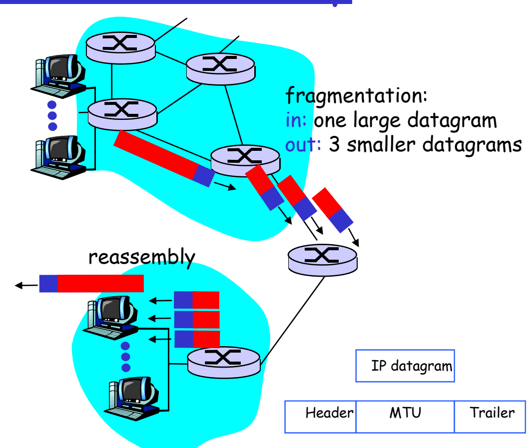


- how much overhead with TCP?
- 20 bytes of TCP
- 20 bytes of IP
- = 40 bytes + application layer overhead

MTU = Maximum Transmission Units

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

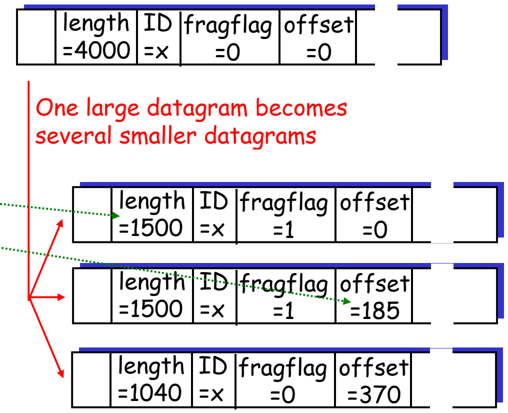


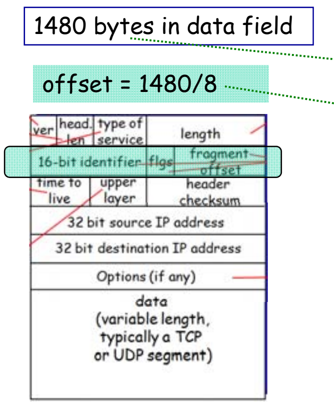
Token Ring (16 Mbps) – 17,914 bytes
Token Ring (4 Mbps) - 4,464 bytes
Ethernet - 1,500 bytes
Point-to-Point Protocol (PPP)–296 bytes

IP Fragmentation and Reassembly

Example

- 4000 byte datagram
- MTU = 1500 bytes





Network Layer : Logical Addressing

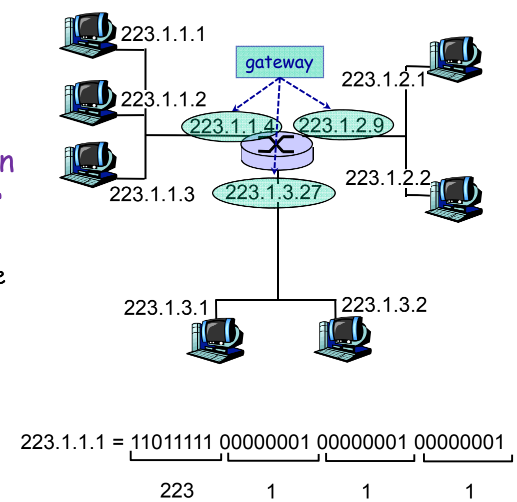
- Communication at network layer is host-to-host
- Computer somewhere in the world need to communicate with another computer somewhere else in the world through Internet
- Packet transmitted by sending computer may pass through several LANs or WANs before reaching destination computer
- We need *global addressing scheme* called *logical addressing*
- Today, we use the term *IP address* to mean a logical address in network layer of TCP/IP protocol suite

IP Addresses

- The Internet address are 32 bits in length
  - Address space is 2^{32} or 4,294,967,296
  - These addresses are referred to as IPv4 (IP version 4) addresses or simply IP address
- The need for more addresses motivated a new design of the IP layer called new generation of IP or IPv6 (IP version 6)
  - The Internet uses 128-bit addresses that give much greater flexibility in address location
  - These addresses are referred to as IPv6 (IP version 6) address

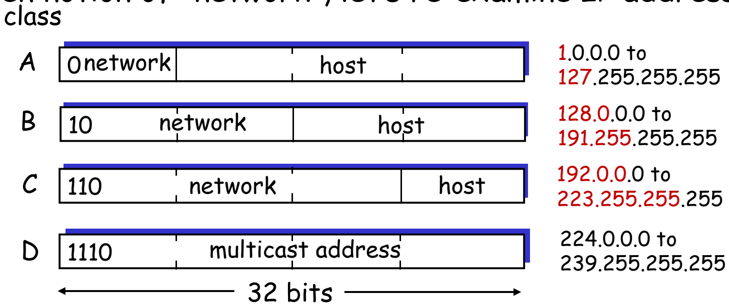
IPv4 Addressing: introduction

- IPv4 address: 32-bit identifier for host, router *interface*
- *Interface:* connection between host/router and physical link
  - router's typically have multiple interfaces
  - host typically has one interface
  - IP addresses associated with each interface



IP Addresses "class-full" addressing:

given notion of "network", let's re-examine IP addresses:



	First byte	Second byte	Third byte	Fourth byte
Class A	0			
Class B	10			
Class C	110			
Class D	1110			
Class E	1111			

a. Binary notation

	First byte	Second byte	Third byte	Fourth byte
Class A	0–127			
Class B	128–191			
Class C	192–223			
Class D	224–239			
Class E	240–255			

b. Dotted-decimal notation

Class	Number of Blocks	Block Size	Application
A	128	16,777,216	Unicast
B	16,384	65,536	Unicast
C	2,097,152	256	Unicast
D	1	268,435,456	Multicast
E	1	268,435,456	Reserved

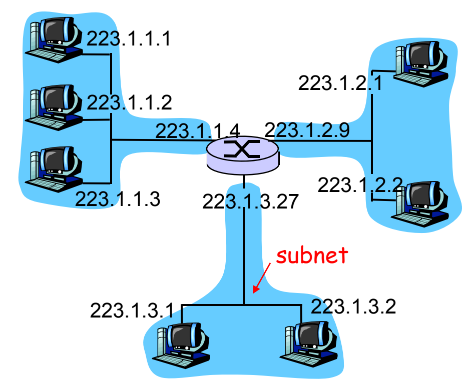
Class	Network Octets (blanks in the IP address are used for octets identifying hosts)	Total Number of Possible Networks or Licenses	Host Octets (blanks in IP address are used for octets identifying networks)	Total Number of Possible IP Addresses in Each Networks
A	0.__.__.__ to 127.__.__.__	128	__.0.0.1 to __.255.255.254	16,777,214
B	128.0.__.__ to 191.255.__.__	64x256 16,384	__.__.0.1 to __.__.255.254	65,534
C	192.0.0.__ to 223.255.255.__	32x256x256 2,097,152	__.__.__.1 to __.__.__.254	254

Address for Private Networks

	Range			Total
Class A	10.0.0.0	to	10.255.255.255	2^{24}
Class B	172.16.0.0	to	172.31.255.255	2^{20}
Class C	192.168.0.0	to	192.168.255.255	2^{16}

Subnets

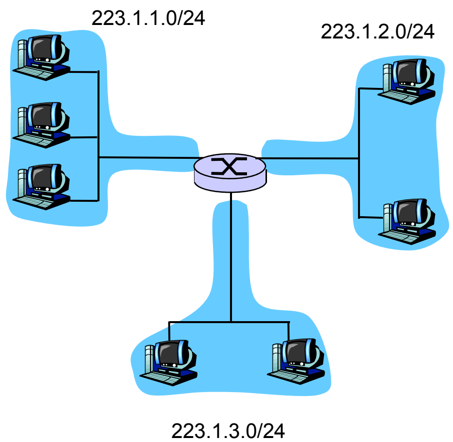
- IP address:
  - subnet part (high order bits)
  - host part (low order bits)
- *What's a subnet ?*
  - device interfaces with same subnet part of IP address
  - can physically reach each other without intervening router



network consisting of 3 subnets

Class C

Subnets



Recipe

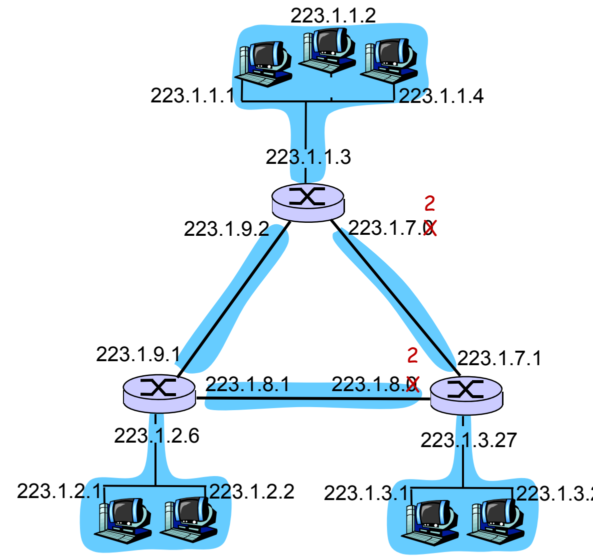
- To determine the subnets, detach each interface from its host or router, creating islands of isolated networks.
- Each isolated network is called a subnet.

Subnet mask: /24

Class C

Subnets

How many?



IP addressing: CIDR

CIDR: Classless InterDomain Routing

- Subnet portion of address of arbitrary length
- Address format: a.b.c.d/x, where x is # bits in subnet portion of address

subnet part | host part

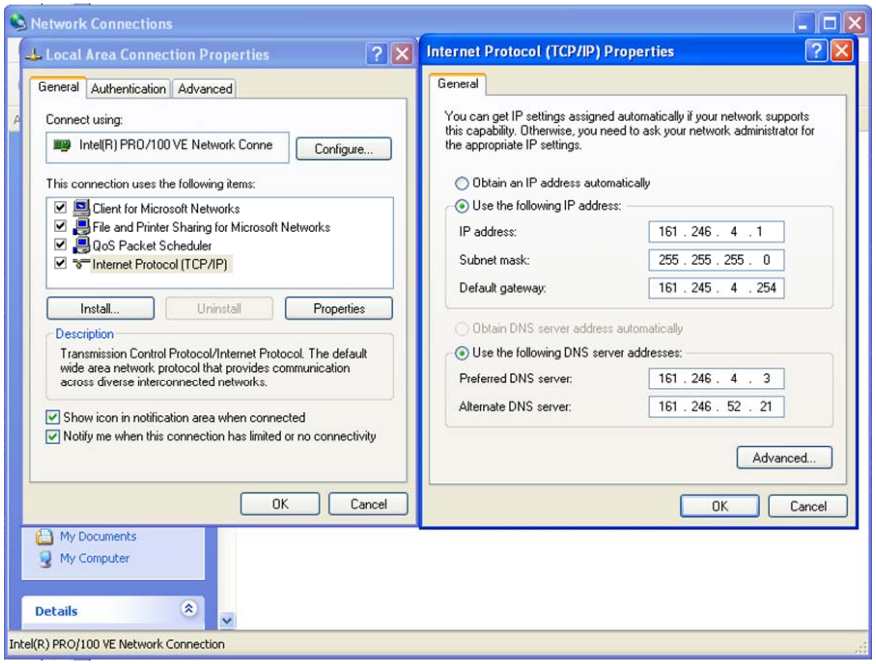
11001000 00010111 00010000 00000000

200.23.16.0/23

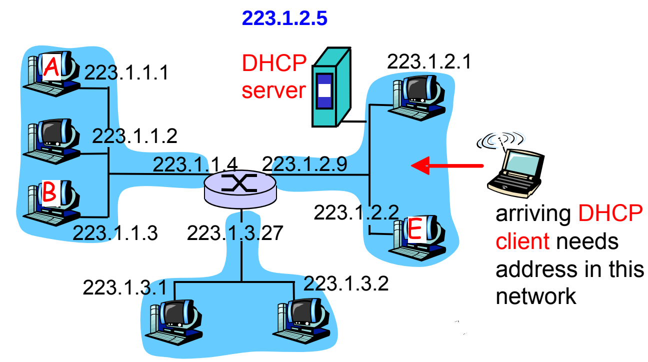
ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000 00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010 00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100 00000000	200.23.20.0/23
...	....	....
Organization 7	11001000 00010111 00011110 00000000	200.23.30.0/23

IP addresses: how to get one?

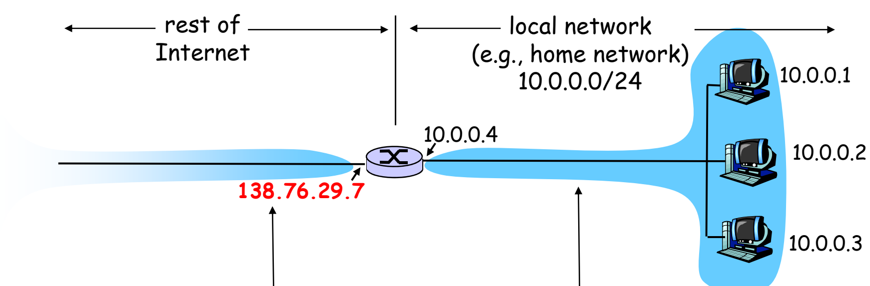
- Q: How does *host* get IP address?
- hard-coded by system admin in a file
  - Windows:
    - control-panel->network connections->properties ->Internet Protocol (TCP/IP)
  - UNIX: /etc/rc.config
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"
  - allow host to *dynamically* obtain its IP address from network server when it joins network



DHCP client-server scenario



NAT: Network Address Translation



All datagrams *leaving* local network have same single source NAT IP address: 138.76.29.7, different source port numbers

Datagrams with source or destination in this network have 10.0.0.0/24 address for source, destination (as usual)

	Range			Total
Class A	10.0.0.0	to	10.255.255.255	2^{24}
Class B	172.16.0.0	to	172.31.255.255	2^{20}
Class C	192.168.0.0	to	192.168.255.255	2^{16}

NAT: Network Address Translation

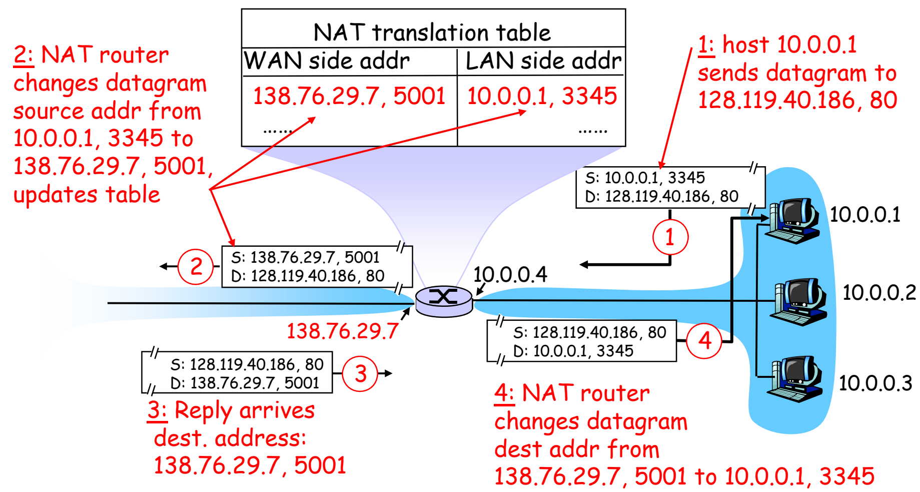
- Motivation: local network uses just *one IP address* as far as outside world is concerned:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can *change addresses of devices* in local network without notifying outside world
  - can *change ISP* without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus).

NAT: Network Address Translation

Implementation: NAT router must:

- *outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
  . . . remote clients/servers will respond using (NAT IP address, new port #) as destination address.
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT IP address, new port #) translation pair
- *incoming datagrams: replace* (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT: Network Address Translation

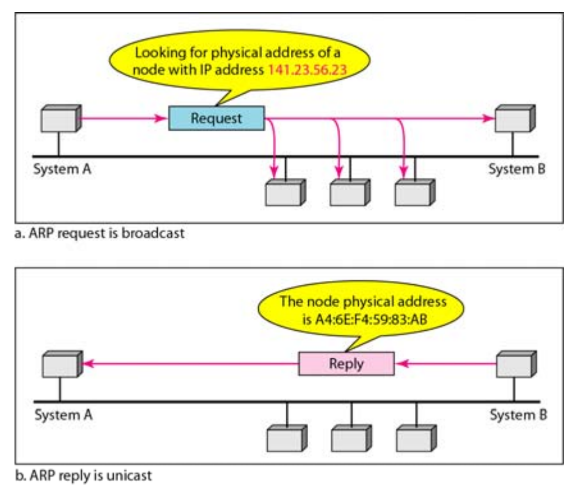


Address Mapping

- Delivery of packet to host or router requires two levels of addressing: logical address and physical address
- We need to be able to map a logical address to its corresponding physical address and vice versa.
- Mapping Logical Address to Physical Address can be done by Address Resolution Protocol (ARP) RFC 826

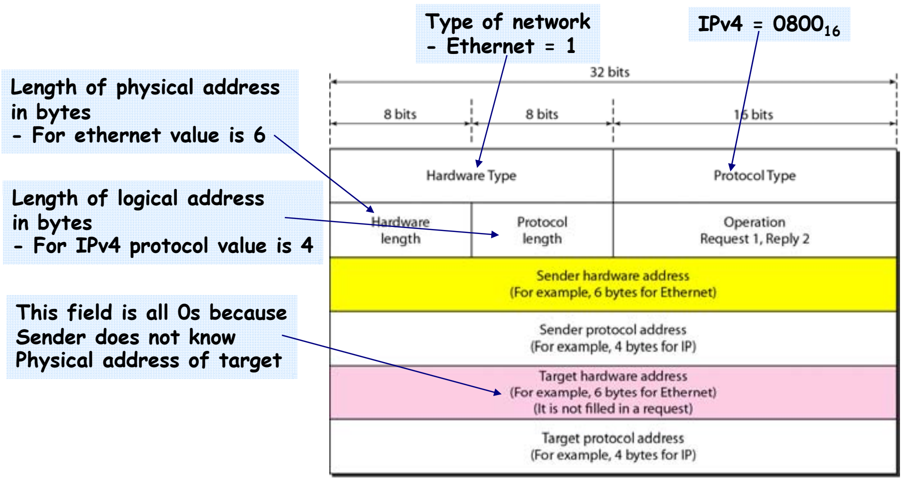
Logical address → Physical address

Address Resolution Protocol (ARP)

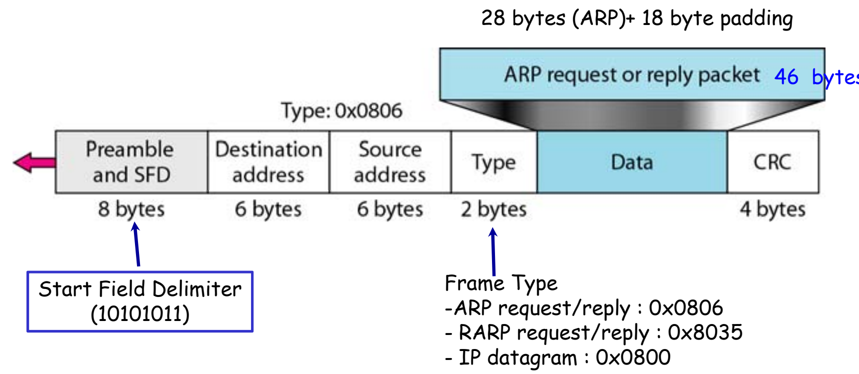


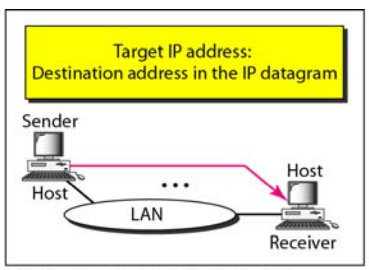
ARP Operation

ARP Packet

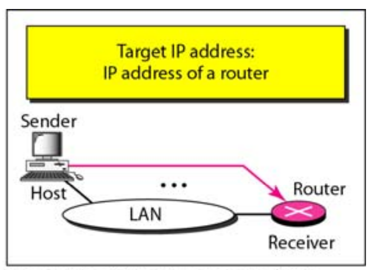


Encapsulating of ARP packet

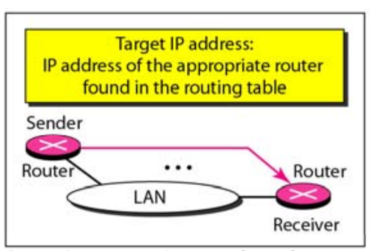




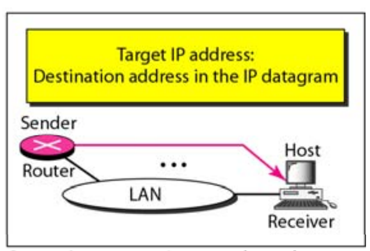
Case 1. A host has a packet to send to another host on the same network.



Case 2. A host wants to send a packet to another host on another network. It must first be delivered to a router.

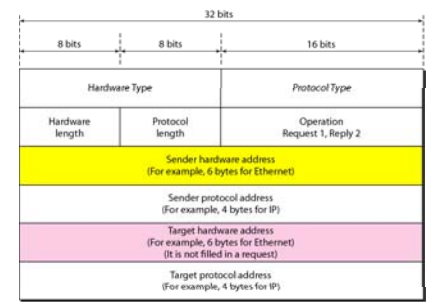


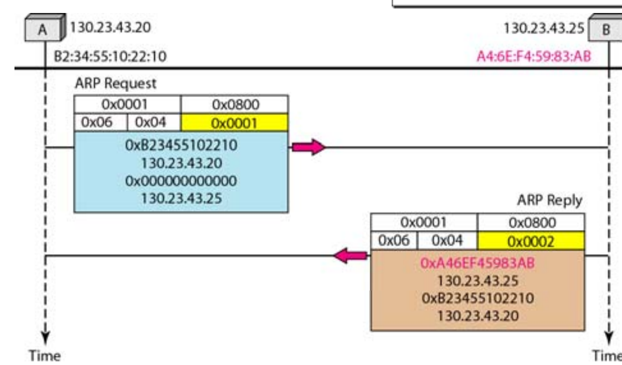
Case 3. A router receives a packet to be sent to a host on another network. It must first be delivered to the appropriate router.



Case 4. A router receives a packet to be sent to a host on the same network.

Example





Internet Control Message Protocol (ICMP)

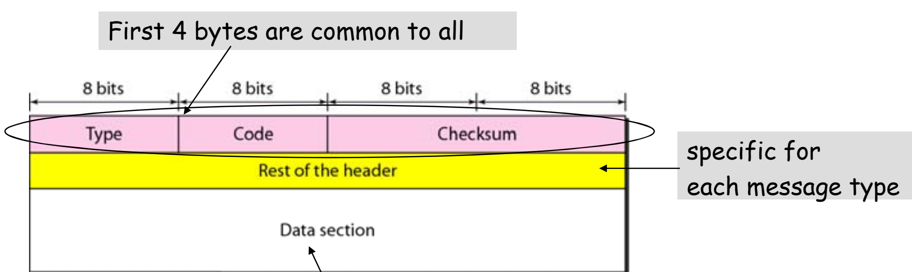
- IP provides unreliable and connectionless datagram delivery
- IP protocol is a best-effort delivery service that delivers datagram from original source to final destination
- Two deficiencies
  - **Lack of error control**
    - No error-reporting or error-correcting mechanism
  - **Lack of assistance mechanism for host and management queries**
    - Host sometimes needs to determine if router or another host is alive
    - Sometimes a network administrator needs information from another host or router

ICMP : Type of Messages

- ICMP message are divided into two broad categories
  - **Error-reporting message**
    - Report problems that router or host (destination) may encounter when it processes IP packet
  - **Query message**
    - Help host or network manager get specific information from router or another host
      - Ex. Nodes can discover their neighbors
      - Hosts can discover and learn about routers on their network

ICMP : Message Format

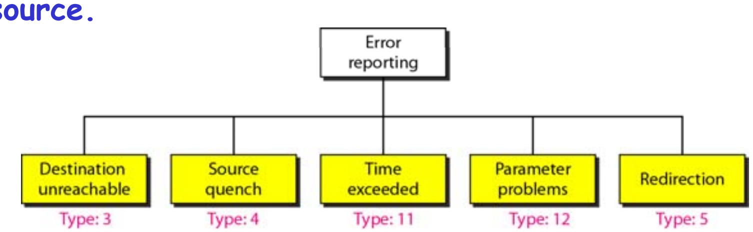
- ICMP message has 8-byte header and variable-size data section



- In **error messages** carries information for finding original packet that had error
- In **query messages** data section carries extra information based on type of query

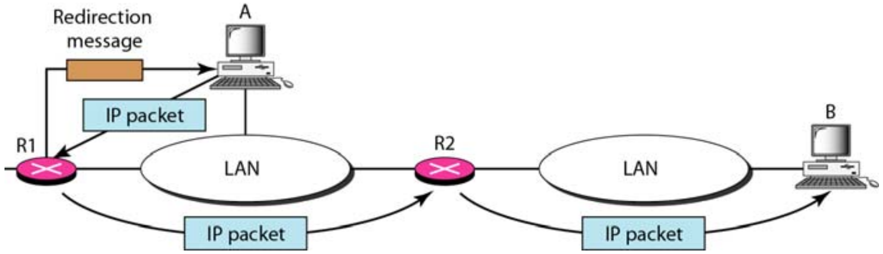
ICMP : Error Reporting

- **ICMP always reports error messages to original source.**



Type	Code	description
3	0	dest. network unreachable
3	1	dest host unreachable
3	2	dest protocol unreachable
3	3	dest port unreachable
3	6	dest network unknown
3	7	dest host unknown
4	0	source quench (congestion control - not used)
11	0	TTL expired
12	0	bad IP header

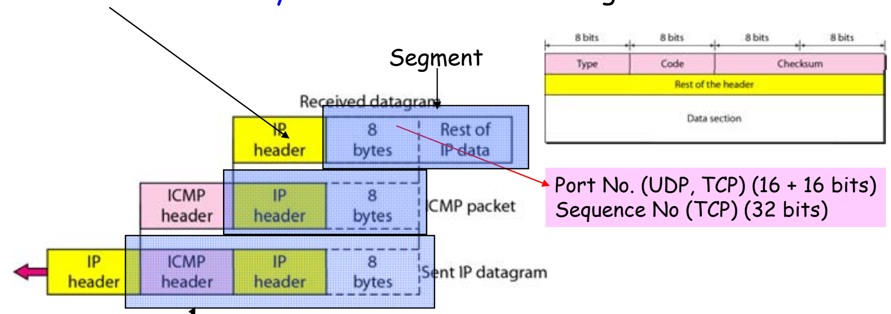
Redirection Concept



- Host A wants to send a datagram to host B
- Router R2 is obviously most efficient routing choice, but host A did not choose router R2
- Datagram goes to R1 instead
- Router R1, after consulting its table, finds that packet should have gone to R2
- R1 sends packet to R2 and,at the same time, sends a redirection message to host A
- Host A's routing table can now be updated

ICMP : Error Reporting (continued)

- All error messages contain data section that includes
  - IP header of the original datagram plus
  - the first 8 bytes of data in that datagram



- network-layer "above" IP:
  - ICMP messages carried in IP datagrams

Type: 3	Code: 0 to 15	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Destination Unreachable

Type: 4	Code: 0	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Source Quench

0 : Time to Live exceeded in Transit
1 : Fragment Reassembly Time Exceeded

Type: 11	Code: 0 or 1	Checksum
Unused (All 0s)		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

TTL Expired

Code 0 : Error in one of header field, value in pointer field points to byte with problem
Code 1 : required part of option is missing

Type: 12	Code: 0 or 1	Checksum
Pointer	Unused (All 0s)	
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

Parameter Problem

Type: 5	Code: 0 to 3	Checksum
IP address of the target router		
Part of the received IP datagram including IP header plus the first 8 bytes of datagram data		

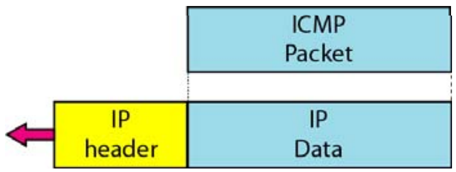
Redirection message format

Code 0 – Redirection for network (or Subnet) -specific route
Code 1 – Redirection for host-specific route
Code 2 – same as code 0 , based on specified type of service
Code 3 – same as code 1, based on specified type of service

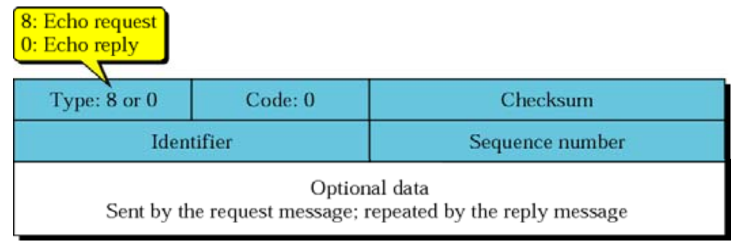
ICMP : Query

Type	Code	description
0	0	echo reply (ping)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery

- Query message is encapsulated in IP packet, which in turn is encapsulated in data link layer frame
- In this case, no bytes of original IP are included in message

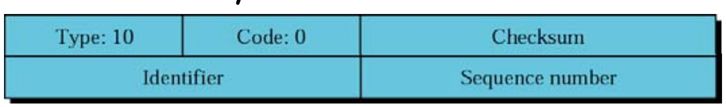


Echo-request and echo-reply messages



Route discovery and Route Advertisement

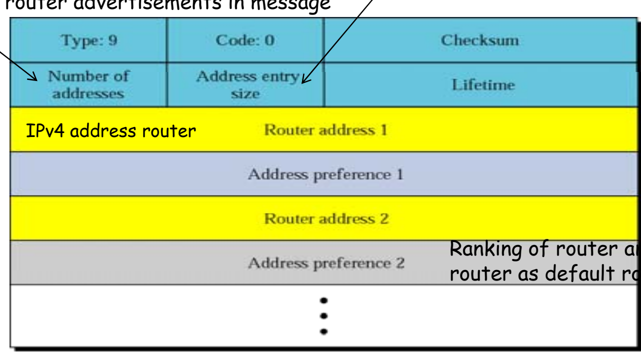
- Route discovery



- Route advertisement

information for each router address entry in the list. The value is normally set to 2 (router address + preference level).

number of router advertisements in message



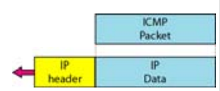
Ranking
0 – default
80000000_{16}

Ranking of router and used to select router as default router

Ping

- Ping program is used to find whether a host is alive or not

56+8 bytes of ICMP header + 20 bytes of IP header



56+8 bytes of ICMP header

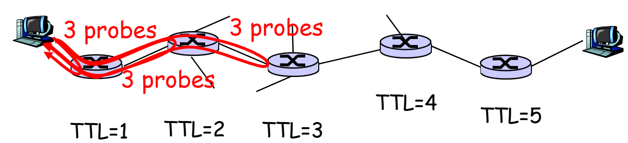
```
$ ping fhda.edu
PING fhda.edu (153.18.8.1)    56 (84) bytes of data.
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=0    ttl=62    time=1.91 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=1    ttl=62    time=2.04 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=2    ttl=62    time=1.90 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=3    ttl=62    time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=4    ttl=62    time=1.93 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=5    ttl=62    time=2.00 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=6    ttl=62    time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=7    ttl=62    time=1.94 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=8    ttl=62    time=1.97 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=9    ttl=62    time=1.89 ms
64 bytes from tiptoe.fhda.edu (153.18.8.1): icmp_seq=10   ttl=62    time=1.98 ms

--- fhda.edu ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10103ms
    rtt min/avg/max = 1.899/1.955/2.041 ms
```

Traceroute

- Source sends series of UDP segments to dest
  - First has TTL =1
  - Second has TTL=2, etc.
  - Unlikely port number
- When nth datagram arrives to n^{th} router:
  - Router discards datagram
  - And sends to source an ICMP message (type 11, code 0) : TTL expired
  - Message includes name of router& IP address
- When ICMP message arrives, source calculates RTT
- Traceroute does this 3 times
- Stopping criterion
- UDP segment eventually arrives at destination host
- Destination returns ICMP "port unreachable" packet (type 3, code 3 : Destination Port Unreachable)
- When source gets this ICMP, stops.

"Real" Internet delays and routes

- What do "real" Internet delay & loss look like?
- Traceroute program (tracert for windows) : provides delay measurement from source to router along end-end Internet path towards destination. For all *i*:
  - sends three packets that will reach router *i* on path towards destination
  - router *i* will return packets to sender
  - sender times interval between transmission and reply.



"Real" Internet delays and routes

traceroute: gaia.cs.umass.edu to www.eurecom.fr

Three delay measurements from gaia.cs.umass.edu to cs-gw.cs.umass.edu

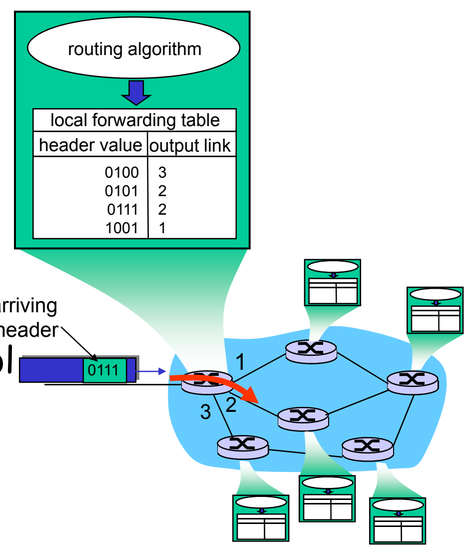
```
1  cs-gw (128.119.240.254)  1 ms  1 ms  2 ms
2  border1-rt-fa5-1-0.gw.umass.edu (128.119.3.145)  1 ms  1 ms  2 ms
3  cht-vbns.gw.umass.edu (128.119.3.130)  6 ms 5 ms 5 ms
4  jn1-at1-0-0-19.wor.vbns.net (204.147.132.129)  16 ms 11 ms 13 ms
5  jn1-so7-0-0-0.wae.vbns.net (204.147.136.136)  21 ms 18 ms 18 ms
6  abilene-vbns.abilene.ucaid.edu (198.32.11.9)  22 ms  18 ms  22 ms
7  nycm-wash.abilene.ucaid.edu (198.32.8.46)  22 ms  22 ms  22 ms
8  62.40.103.253 (62.40.103.253)  104 ms 109 ms 106 ms
9  de2-1.de1.de.geant.net (62.40.96.129)  109 ms 102 ms 104 ms
10  de.fr1.fr.geant.net (62.40.96.50)  113 ms 121 ms 114 ms
11  renater-gw.fr1.fr.geant.net (62.40.103.54)  112 ms  114 ms  112 ms
12  nio-n2.cssi.renater.fr (193.51.206.13)  111 ms  114 ms  116 ms
13  nice.cssi.renater.fr (195.220.98.102)  123 ms  125 ms  124 ms
14  r3t2-nice.cssi.renater.fr (195.220.98.110)  126 ms  126 ms  124 ms
15  eurecom-valbonne.r3t2.ft.net (193.48.50.54)  135 ms  128 ms  133 ms
16  194.214.211.25 (194.214.211.25)  126 ms  128 ms  126 ms
17  * * *
18  * * *
19  fantasia.eurecom.fr (193.55.113.142)  132 ms  128 ms  136 ms
```

trans-oceanic link

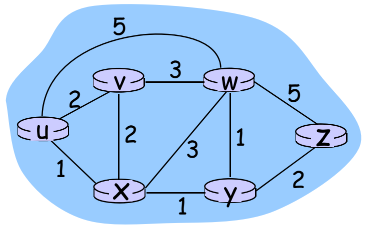
* means no response (probe lost, router not replying)

Routing Algorithms

- Link state
  - Ex. Open Shortest Path First (OSPF)
- Distance Vector
  - Ex. Routing Information Protocol (RIP)



Graph abstraction



Graph: G = (N,E)

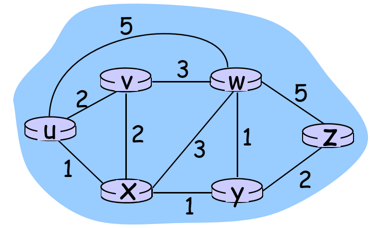
N = set of routers = { u, v, w, x, y, z }

E = set of links ={ (u,v), (u,x), (v,x), (v,w), (x,w), (x,y), (w,y), (w,z), (y,z) }

Remark: Graph abstraction is useful in other network contexts

Example: P2P, where N is set of peers and E is set of TCP connections

Graph abstraction: costs



• c(x,x') = cost of link (x,x')

  - e.g., c(w,z) = 5

• cost could always be 1, or inversely related to bandwidth, or inversely related to congestion

Cost of path $(x_1, x_2, x_3,\ldots, x_p) = c(x_1,x_2) + c(x_2,x_3) + \ldots + c(x_{p-1},x_p)$

Question: What's the least-cost path between u and z ?

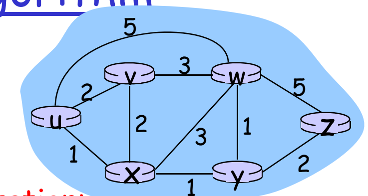
Routing algorithm: algorithm that finds least-cost path

Routing Algorithm classification

- Global or decentralized information?
- Global:
  - all routers have complete topology, link cost information
  - "link state" algorithms
- Decentralized:
  - router knows physically-connected neighbors, link costs to neighbors
  - iterative process of computation, exchange of information with neighbors
  - "distance vector" algorithms

- Static or dynamic?
- Static:
  - routes change slowly over time
- Dynamic:
  - routes change more quickly
    - periodic update
    - in response to link cost changes

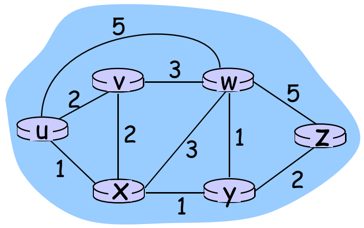
A Link-State Routing Algorithm



- Dijkstra's algorithm
- network topology, link costs known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have same information
- computes least cost paths from one node ('source") to all other nodes
  - gives forwarding table for that node
- iterative: after k iterations, know least cost path to k destination's

- Notation:
- c(x,y): link cost from node x to y; = ∞ if not direct neighbors
- D(v): current value of cost of path from source to destination v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

Dijsktra's Algorithm



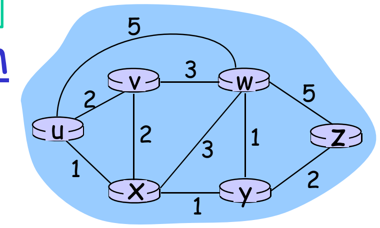

```
Initialization:
  N' = {u}
  for all nodes v
    if v adjacent to u
       then D(v) = c(u,v)
    else D(v) = ∞

Loop
   find w not in N' such that D(w) is a minimum
   add w to N'
   update D(v) for all v adjacent to w and not in N' :
      D(v) = min( D(v), D(w) + c(w,v) )
   /* new cost to v is either old cost to v or known
    shortest path cost to w plus cost from w to v */
until all nodes in N'
```

หา Cost จาก U ไปยัง Node ข้างเคียงก่อน

เลือก node ที่ให้ cost น้อยที่สุด เป็นสมาชิกของ N'

P : โหนดใดๆ ในเครือข่ายที่ไม่เป็นสมาชิกของ N'

Dijsktra's Algorithm




```
Initialization:
  N' = {u}
  for all nodes P
    if P adjacent to u
       then D(P) = c(u, P)
    else D(P) = ∞

Loop
   find Q not in N' such that D(Q) is a minimum
   add Q to N'
   update D(P) for all P adjacent to Q and not in N' :
      D(P) = min( D(P), D(Q) + c(Q, P) )
   /* new cost to P is either old cost to P or known
    shortest path cost to Q plus cost from Q to P */
until all nodes in N'
```

หา Cost จาก U ไปยัง Node ข้างเคียงก่อน

เลือก node ที่ให้ cost น้อยที่สุด เป็นสมาชิกของ N'

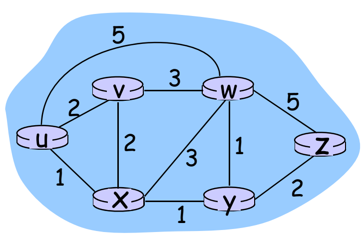
Q : โหนดใดๆ ในเครือข่ายที่ไม่เป็นสมาชิกของ N' และมีค่า Cost น้อยที่สุด

Dijkstra's algorithm: example

D(P): current value of cost of path from source to destination P

p(P): predecessor node along path from source to P

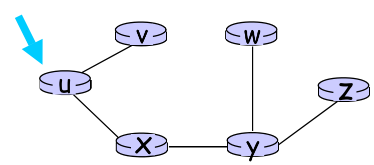
Step	N'	D(v),p(v)	D(w),p(w)	D(x),p(x)	D(y),p(y)	D(z),p(z)
0	u	2,u	5,u	1,u	∞	∞
1	ux	2,u	4,x		2,x	∞
2	uxy	2,u	3,y			4,y
3	uxyv		3,y			4,y
4	uxyvw					4,y
5	uxyvwz					



ต้องการหา **Shortest Path** จาก **Node u** ไปยังทุก **Node**

Dijkstra's algorithm: example (2)

Resulting shortest-path tree from u:



Resulting forwarding table in u:

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

Distance Vector Algorithm

Bellman-Ford Equation (dynamic programming)

Define

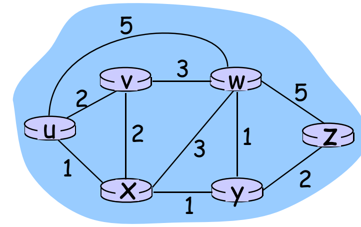
$d_x(y)$:= cost of least-cost path from x to y

Then

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

where min is 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:

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

Node that achieves minimum is **next hop in shortest path** ➔ forwarding table

Distance Vector Algorithm

- Iterative, asynchronous: 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 messg from neighbor)

recompute estimates

if DV to any destination 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

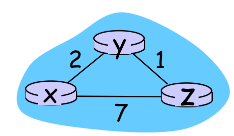
node y table

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

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	7	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node y table

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	7	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

node z table

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

cost to

from	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

cost to

from	x	y	z
x	0	2	3
y	2	0	1
z	3	1	0

time