# The Network Layer: Routing 1

Smith College, CSC 249
March 2, 2018

1

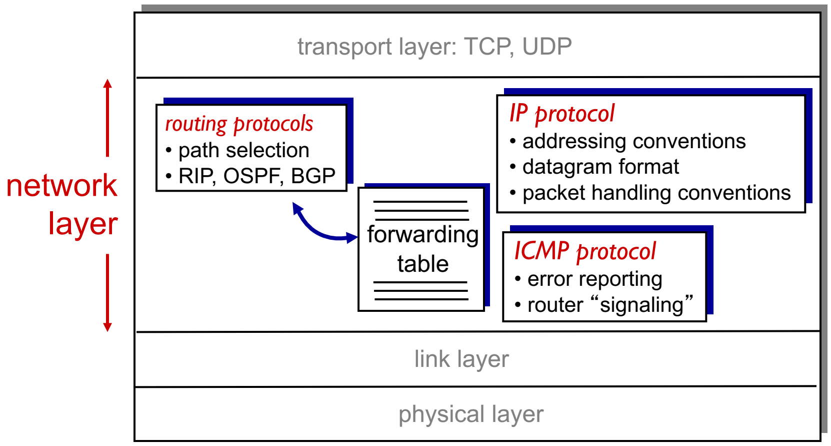
## Summary Chapter 4

- IP Addressing
  - Network prefixes and Subnets
  - IP datagram format
- DHCP – dynamic addressing
  - Obtain: own IP address
  - Subnet mask, DNS server & first-hop router IP address
- NAT – network address translation… at end of class today

2

# Overview of the Network Layer

Network layer functions & protocols:




# Smith College IP Addressing

Possible **QUESTIONS**:

1) Given a mask of 255.255.254.0
   - What is the "/__" notation for this?

2) are the machines with IP addresses 131.229.22.50 and 131.229.23.243 on the same subnet?
   - How many hosts are supported in the range 131.229.22.00/23 ?

## IP addresses: how to get one?

Q: How does *network* get subnet part of IP address?

A: Is allocated a portion of its provider ISP's address space, which gets that from ICANN (Internet Corp. for Assigned Names and Numbers)

Q: How does a *host* get an IP address?

- hard-coded by system administrator in a file, ***or***
- DHCP: Dynamic Host Configuration Protocol: dynamically get address from as server
  - "plug-and-play"

5

## DHCP: Dynamic Host Configuration Protocol

Goal: allow host to *dynamically* obtain its IP address from network server when it joins a network

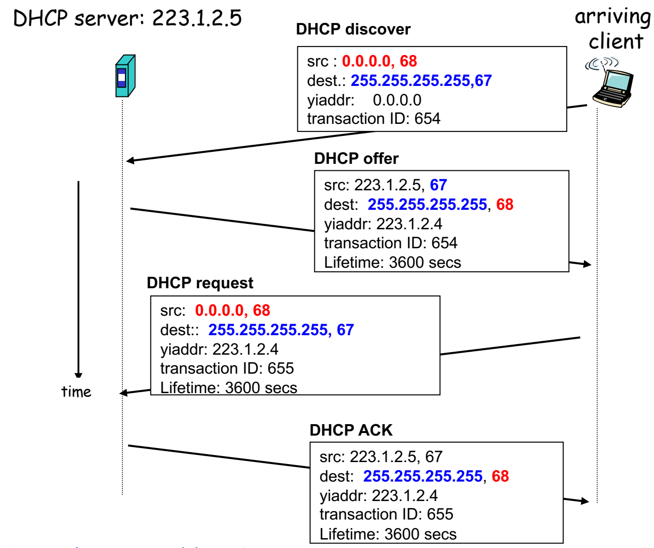
- Can renew its lease on the IP address it is using
- Allows reuse of addresses once one host leaves
- Support for mobile users to join networks

DHCP overview:

1) host broadcasts "DHCP discover" msg
2) DHCP server responds with "DHCP offer" msg
3) host requests IP address: "DHCP request" msg
4) DHCP server sends address: "DHCP ack" msg

6

## DHCP client-server scenario




yiaddr = 'your internet address'
**broadcast** address, 255.255.255.255 → sent to every host *in the subnet*



## NAT: Network Address Translation

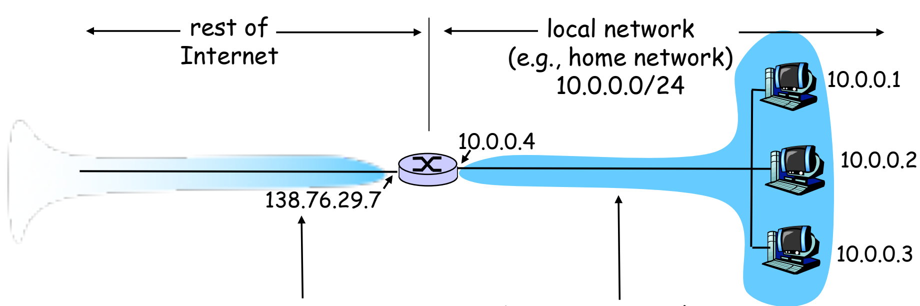
- Motivation: local (home) network uses just one IP address as far as outside world view:
  - range of addresses not needed from ISP: just one IP address for all devices
  - can change addresses of devices in local network without notifying outside world
  - can change ISP without changing addresses of devices in local network
  - devices inside local net not explicitly addressable, visible by outside world (a security plus)
- Range of addresses within: 10.0.0.0/24

## Standard Reserved IP Address Blocks for Private Network Use

- 10.0.0.0/8
- 172.16.0.0/12
- 192.168.0.0/16

9

## NAT: Network Address Translation




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

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

10

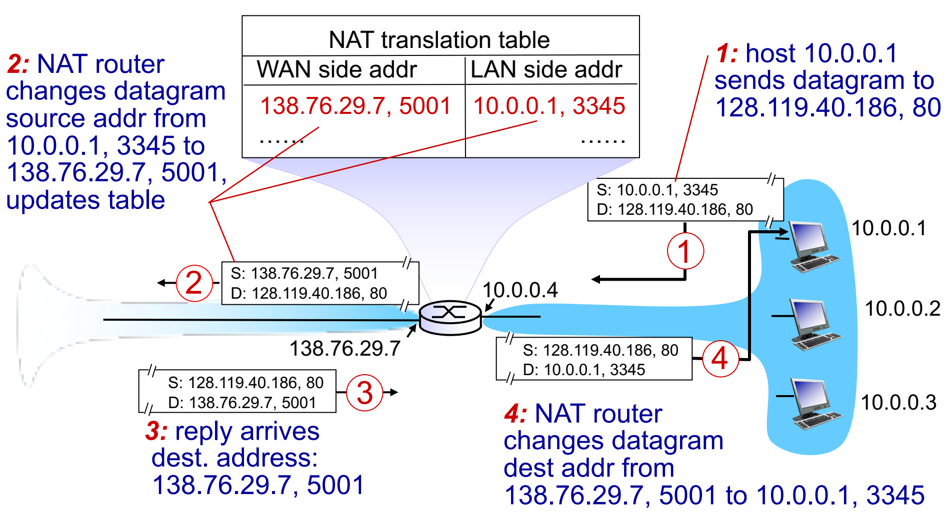
# NAT Router Tasks

Implementation: NAT router must:

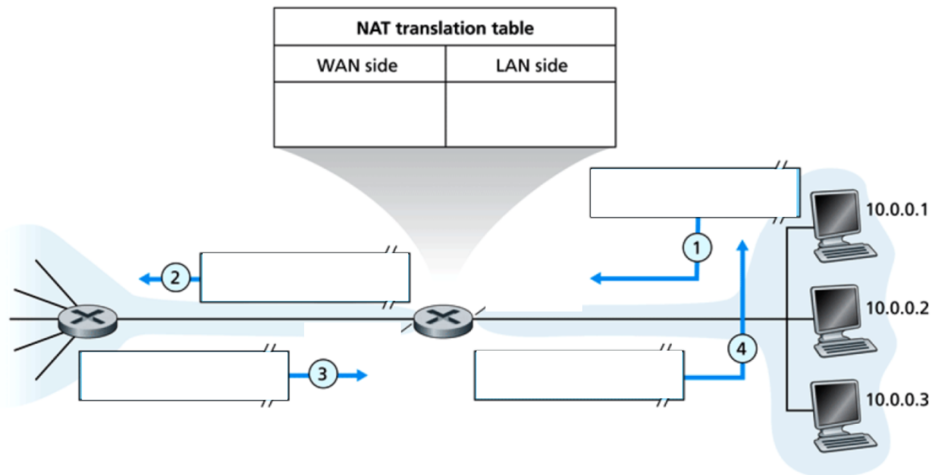
- *for outgoing datagrams: replace* (source IP address, port #) of every outgoing datagram with (NAT IP address, new port #)
  - remote clients/servers will respond using (NAT IP address, new port #) as destination address
- *remember (in NAT translation table)* every (source IP address, port #) to (NAT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



# NAT: network address translation



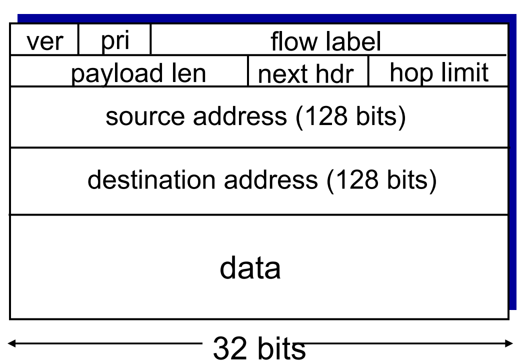
# NAT Question on Handout






# IPv6 datagram format

*priority:* identify priority among datagrams in flow
*flow Label:* identify datagrams in same "flow."
(concept of "flow" not well defined).
*next header:* identify upper layer protocol for data



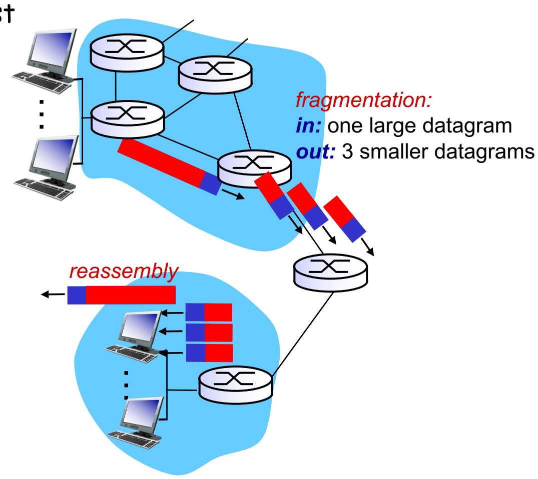
## NAT Controversies?

- Port numbers are used by NAT to identify hosts (and the process) within the local network – but ports are for addressing *processes only* not *hosts*
- Routers should only process packets up to layer 3 (ports associated with app socket)
- Violates end-to-end argument
  - NAT possibility must be taken into account by application designers, *e.g.*, P2P applications
  - Interfering nodes should not modify IP addresses and port numbers
- Address shortage should instead be solved by IPv6

15

## IP fragmentation & reassembly

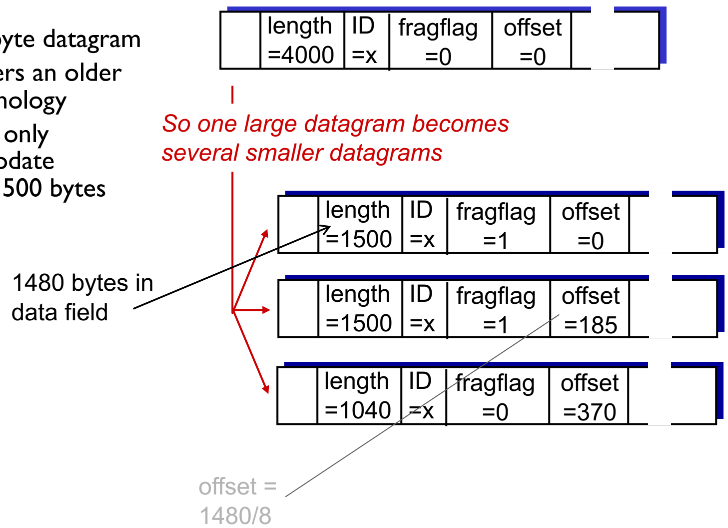
- network links have MTU (max. transfer size) - largest possible link-level frame
  - different link technologies have different MTUs
- large IP datagram may be divided ("fragmented") within a network when the link technology changes
  - one datagram becomes several datagrams
  - "reassembled" only at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



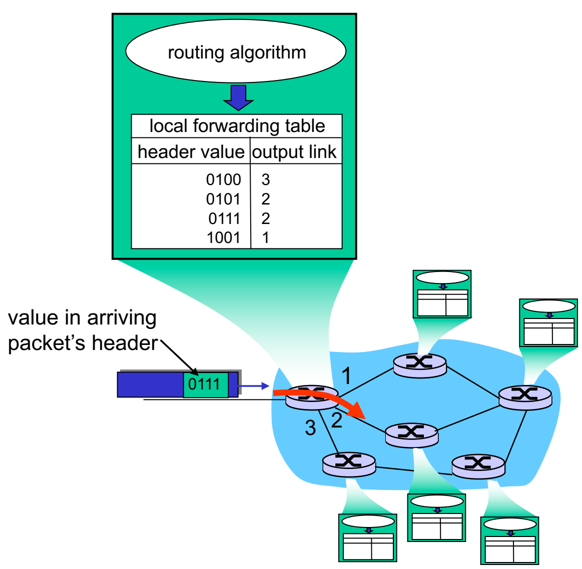
# IP fragmentation & reassembly

*example:*

- A 4000 byte datagram
- Encounters an older link technology
- That can only accommodate MTU = 1500 bytes



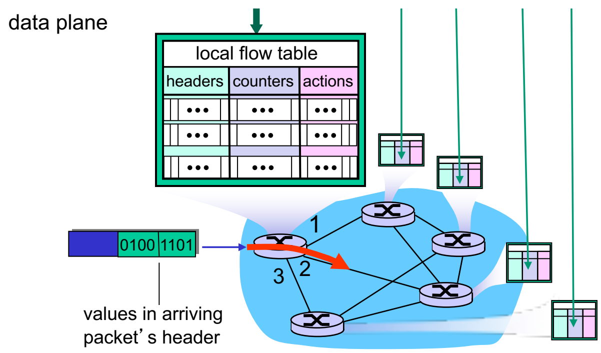

# Recap: Routing v. Forwarding




18

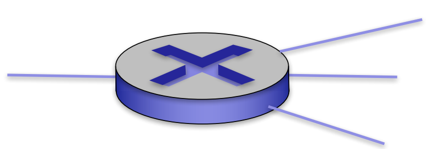
# Generalized Forwarding and SDN

Each router contains a *flow table* that is computed and distributed by a *logically centralized* routing controller




# OpenFlow data plane abstraction

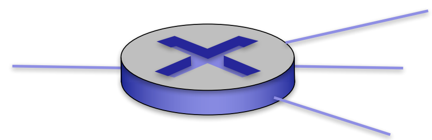
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



*Flow table in a router (computed and distributed by controller) define router's match+action rules*

# OpenFlow data plane abstraction

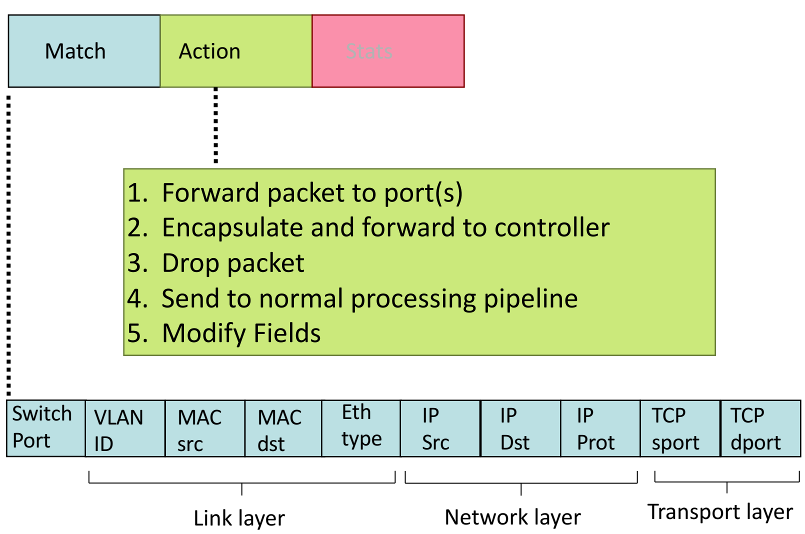
- generalized forwarding: simple packet-handling rules
  - *Pattern:* match values in packet header fields
  - *Actions: for matched packet:* drop, forward, modify, matched packet or send matched packet to controller
  - *Priority*: disambiguate overlapping patterns
  - *Counters:* #bytes and #packets



* : wildcard

1. src=1.2.*.*, dest=3.4.5.* → drop
2. src = *.*.*.*, dest=3.4.*.* → forward(2)
3. src=10.1.2.3, dest=*.*.*.* → send to controller

# OpenFlow: Flow Table Entries



# Examples

Destination-based forwarding:

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Proto | TCP sport | TCP dport | Action |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | 51.6.0.8 | * | * | * | port6 |

*IP datagrams destined to IP address 51.6.0.8 should be forwarded to router output port 6*

Firewall:

| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Proto | TCP sport | TCP dport | Forward |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | * | * | * | * | 22 | drop |

*do not forward (block) all datagrams destined to TCP port 22*

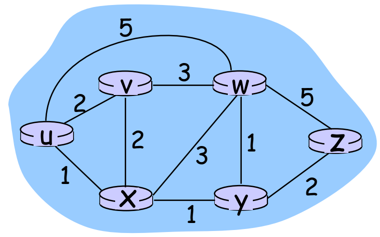
| Switch Port | MAC src | MAC dst | Eth type | VLAN ID | IP Src | IP Dst | IP Prot | TCP sport | TCP dport | Forward |
|---|---|---|---|---|---|---|---|---|---|---|
| * | * | * | * | * | 128.119.1.1 | * | * | * | * | drop |

*do not forward (block) all datagrams sent by host 128.119.1.1*

# Overview of Routing

- The "control plane"
- What is the objective of routing?
- Does routing occur between hosts or routers?
- What are differences between centralized (global) and decentralized algorithms?
  - What are examples of each?
  - Amount of information initially
  - How information is shared/spread
  - Synchronous or asynchronous?
    - (see pathologies as well)

# Routing Notation




Graph: G = (N,E)

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

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

25

# A Link-State Routing Algorithm

Dijkstra's algorithm

- Computes the shortest paths in a graph by using weights on edges as a measure of distance.
  - Starts with complete information
  - A path with the least number of edges may not be the path with the least weight / least cost.
- Each node has global information on network topology and edge weights
- A 'Greedy' algorithm
  - Makes the locally optimum choice, with objective of finding the global optimum

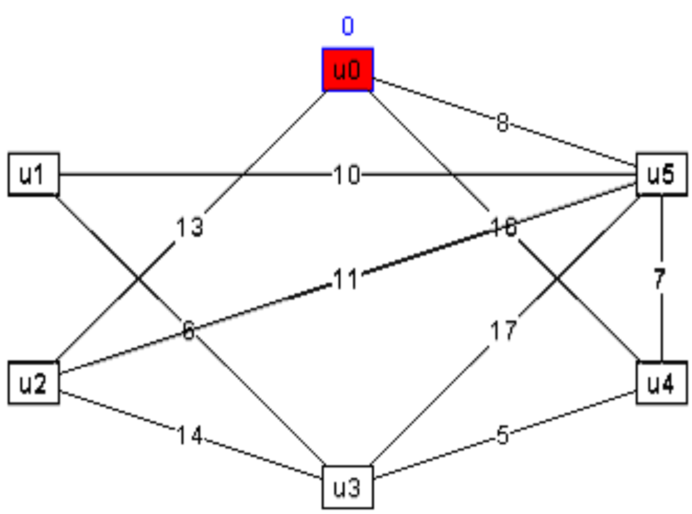
26

## Dijkstra 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 dest. v
- p(v): predecessor node along path from source to v
- N': set of nodes whose least cost path definitively known

27

## A Link-State Routing Algorithm




28

# A Link-State Routing Algorithm

Dijkstra's algorithm

- computes least cost paths from one node ('source') to all other nodes
  - Determines the forwarding table for that node
- The network topology and link costs are known to all nodes
  - accomplished via "link state broadcast"
  - all nodes have the same information
- The algorithm is iterative: after k iterations, the least cost paths to k destinations are known

http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/DijkstraApp.shtml?demo1

http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/DijkstraApp.shtml?demo7

http://www-b2.is.tokushima-u.ac.jp/~ikeda/suuri/dijkstra/DijkstraApp.shtml?demo8



# Dijsktra's Algorithm for node 'u'

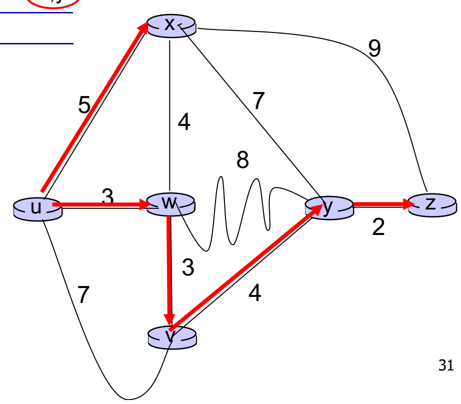
```
Initialization:
  N' = {u}
  for all nodes v
    if v is neighbor to u
      then D(v) = c(u,v)   (D(v): current value of cost of path from source to dest. v)
    else D(v) = ∞

Loop
   find some w not yet 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 are in set N'
```

# Dijkstra's algorithm: example

| Step | N' | D(v) p(v) | D(w) p(w) | D(x) p(x) | D(y) p(y) | D(z) p(z) |
|---|---|---|---|---|---|---|
| 0 | u | 7,u | 3,u | 5,u | ∞ | ∞ |
| 1 | uw | 6,w | | 5,u | 11,w | ∞ |
| 2 | uwx | 6,w | | | 11,w | 14,x |
| 3 | uwxv | | | | 10,v | 14,x |
| 4 | uwxvy | | | | | 12,y |
| 5 | uwxvyz | | | | | |

*notes:*

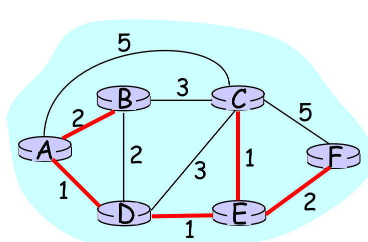
- Construct shortest path tree by tracing predecessor nodes
- Construct the forwarding table by recording the *next hop* to the destination node
- What is the forwarding table??




31

# Dijkstra's algorithm: example

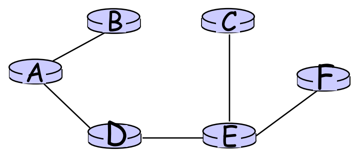
| Step | start N' | D(B),p(B) | D(C),p(C) | D(D),p(D) | D(E),p(E) | D(F),p(F) |
|---|---|---|---|---|---|---|
| 0 | A | 2,A | 5,A | 1,A | infinity | infinity |
| 1 | | | | | | |
| 2 | | | | | | |
| 3 | | | | | | |
| 4 | | | | | | |
| 5 | | | | | | |




32

## Dijkstra's algorithm: example

Resulting shortest-path tree from A:




Resulting forwarding table in A:

| destination | link |
|---|---|
| B | (A, B) |
| D | (A, D) |
| E | (A, D) |
| C | (A, D) |
| F | (A, D) |

33

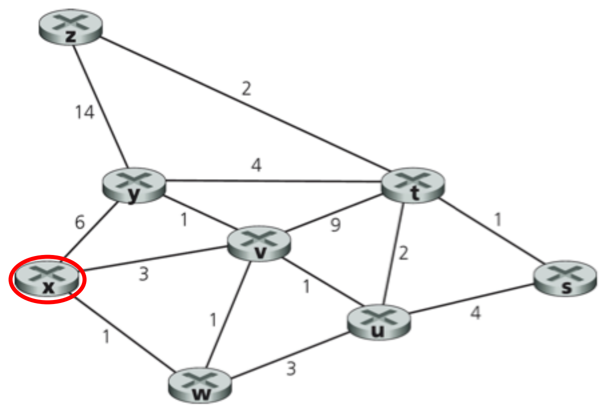
## Routing Activity

- Each pair, or table, be a different router
- Fill in table on handout using Dijkstra's algorithm, for your router letter (IP address)
- Create the forwarding table (back side of handout)
- Send datagrams to a distant destination, forwarding the datagrams to the appropriate "next-hop" using your forwarding table.

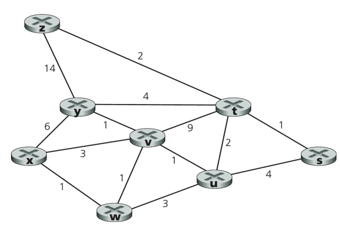
34

# Link State Example

Use Dijkstra's algorithm to **compute** the least-cost-path table for node x, and the forwarding table for x's router




35




| Step | N' | D(s),p(s) | D(t),p(t) | D(u),p(u) | D(v),p(v) | D(w),p(w) | D(y),p(y) | D(z),p(z) |
|---|---|---|---|---|---|---|---|---|
| 0 | X | ∞ | ∞ | ∞ | | | | ∞ |
| 1 | x | | | | | | | |
| 2 | x | | | | | | | |
| 3 | x | | | | | | | |
| 4 | x | | | | | | | |
| 5 | x | | | | | | | |
| 6 | x | | | | | | | |
| 7 | x | | | | | | | |

## Final Step: The Forwarding Table

| Destination | Link |
|---|---|
| S | |
| T | |
| U | |
| V | |
| W | |
| Y | |
| Z | |

37

## Algorithm 2: Distance Vector

Rather than using global information, a distance vector algorithm is:

- distributed:
  - each node communicates only with directly-attached neighbors
- iterative:
  - continues until no nodes exchange info.
  - self-terminating: no "signal" to stop
- asynchronous:
  - nodes need *not* exchange information or iterate in lock step!

38

## Distance Vector Algorithm

Bellman-Ford Equation, an important relationship among costs of least-cost paths

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

39

## Summary

Forwarding:

- Leads to questions of addressing
  - Assignment of IP addresses
  - NAT, IPv6 …

Routing:

- Routing objectives
- Routing notation
- Routing classification
- Link state v. Distance Vector
- Hierarchical structure

40