Lecture 19: Distance Vector Routing

COMP 332, Spring 2024

Victoria Manfredi

Acknowledgements: materials adapted from Computer Networking: A Top Down Approach 7th edition: ©1996-2016, J.F Kurose and K.W. Ross, All Rights Reserved as well as from slides by Abraham Matta at Boston University, and some material from Computer Networks by Tannenbaum and Wetherall.

Today

1. Announcements
   - hw7 due next Wed.
   - what's a virtual machine?
   - run the traceroute command and look at traffic in wireshark
     - compare with pkts you're generating
   - socket.inet_aton, socket.ntoa_inet()
     - to convert string address to/from 32-bit packed address
2. Control plane
   - Distance vector routing
   - Link state vs. distance vector routing
   - Learning to route
3. Internet Control Message Protocol (ICMP)
4. Homework 7 help: virtual machine, coding

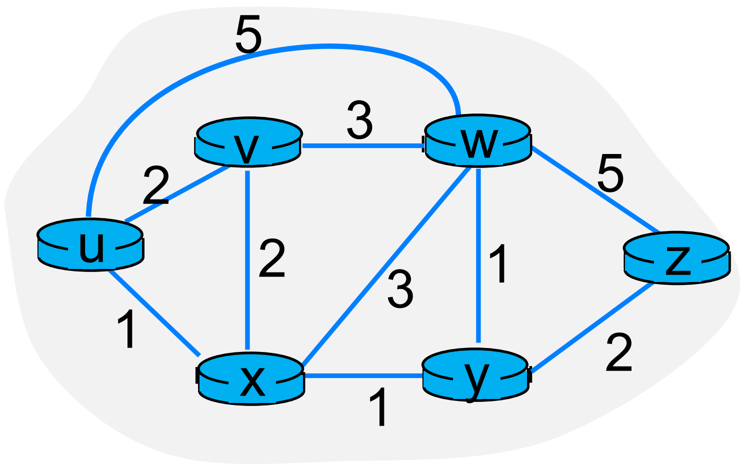
Takeaways from last time

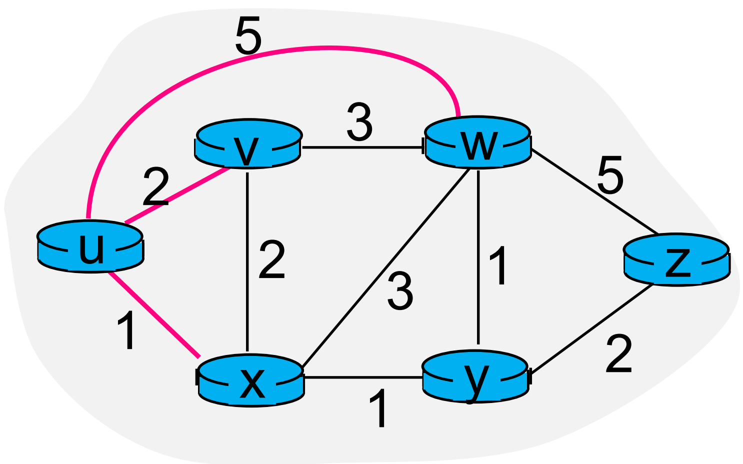
Global information

- global link state algorithms
- all routers have complete topology, link cost info
- *exchange info onLy about neighbors but with all nodes*

Local/decentralized information

- decentralized distance vector algorithms
- router knows only physically-connected neighbors, link costs to neighbors
- iterative computation
- *exchange info about all nodes but only with neighbors*





Both are used on Internet. First cover abstractly and then talk about specific Internet protocols (OSPF, BGP, RIP, …)

Control Plane

DISTANCE VECTOR ROUTING

Distance vector routing

Distance vector (DV)

- vector of best known costs from router to each dst and link to use

Each node x maintains

- Link cost from x to each neighbor v
  - $c(x,v)$
- x's own DV
  - $D_x(y)$: estimate of least cost path from x to node y
  - $D_x = [D_x(y): y \in N]$
- DV for each nbr v
  - $D_v(y)$: estimate of least cost path from neighbor v to node y
  - $D_v = [D_v(y): y \in N]$

Each node periodically sends its own DV to neighbors

- rather than link state costs

Bellman-Ford equation to update DV estimates

Uses dynamic programming

- break problem into simpler sub-problems
- solve each sub-problem once and store solution

Bellman-Ford equation

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

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

cost from neighbor v to destination y

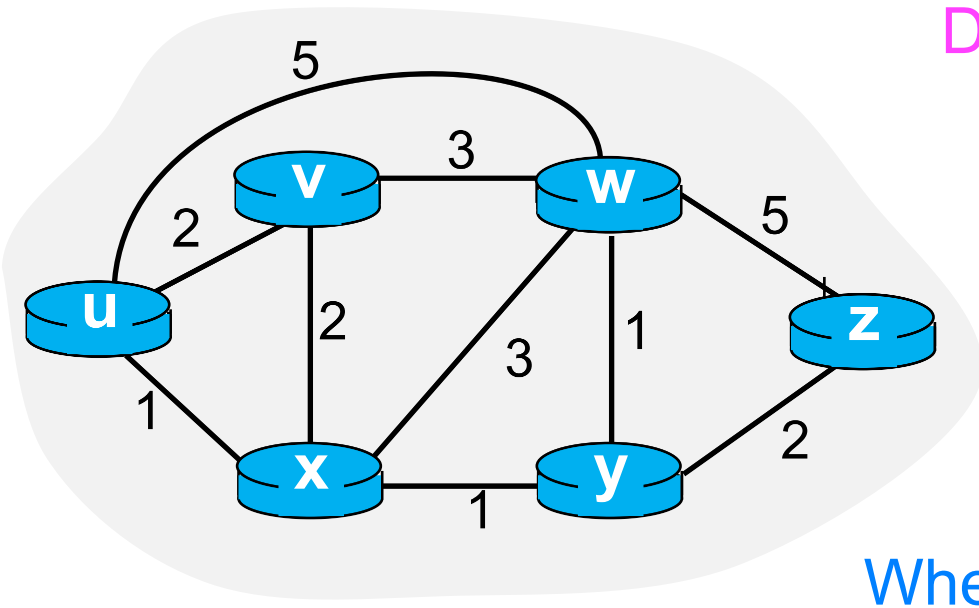
cost to neighbor v

min taken over all neighbors v of x

When x receives new DV estimate from neighbor

- x updates its own DV using B-F equation

Example: compute min cost path from u to z



Bellman-Ford equation

$$
\begin{aligned}
D_u(z) &= \min \{c(u,v) + D_v(z), \\
&\qquad c(u,x) + D_x(z), \\
&\qquad c(u,w) + D_w(z)\} \\
&= \min \{2 + 5, \\
&\qquad 1 + 3, \\
&\qquad 5 + 3\} \\
&= 4
\end{aligned}
$$

Where

$D_v(z) = 5$, $D_x(z) = 3$, $D_w(z) = 3$

Node achieving minimum is next hop in shortest path

- put in forwarding table

Distance vector algorithm run at each node x

Initialization

For all dst y є N

if y is nbr of x

$D_x(y) = c(x, y)$

else

$D_x(y) = \infty$

For each nbr w and dst y є N

$D_w(y) = \infty$

Send x's DV to all nbrs w

$D_x = [D_x(y) : y \in N]$

Loop

x waits for change in local link cost or DV msg from neighbor

recompute estimates

$D_x(y) = \mathit{min}\ v\ \{ c(x,v) + D_v(y) \}$

if x's DV to any dst has changed, *notify* neighbors

Q: when does loop terminate?

Distance vector algorithm run at each node x

Initialization

For all dst y ϵ N

if y is nbr of x

$D_x(y) = c(x, y)$

else

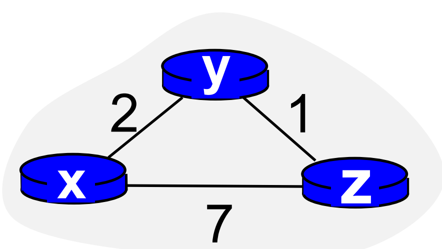
$D_x(y) = \infty$

For each nbr w and dst y ϵ N

$D_w(y) = \infty$

Send x's DV to all nbrs w

$D_x = [D_x(y) : y \in N]$



\# of nodes

\# of neighbors

Node x

cost to

from	x	y	z
x	0	2	7
y	∞	∞	∞
z	∞	∞	∞

Node y

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

Node z

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

Node x

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

Node y

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

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

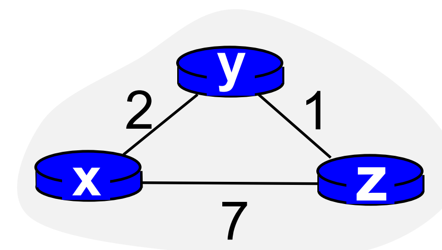
Node z

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	7	1	0



time

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

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

3 min: What is $D_x(z)$ update?

Node x

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

Node y

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

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

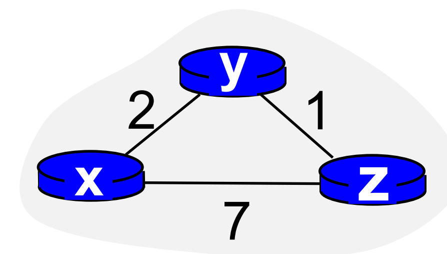
Node z

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



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

time

Node x

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

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	x	y	z

No change: don't send out DV

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

Node z

cost to

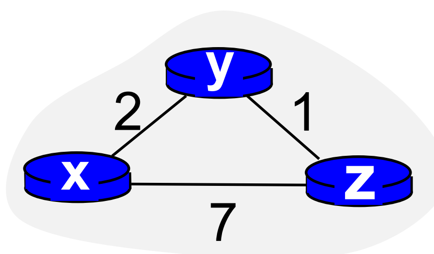
from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

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

Node x

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

No change: don’t send out DV

Node y

cost to

from	x	y	z
x	∞	∞	∞
y	2	0	1
z	∞	∞	∞

from	x	y	z

No change: don’t send out DV

from	x	y	z

No change: don’t send out DV

Node z

cost to

from	x	y	z
x	∞	∞	∞
y	∞	∞	∞
z	7	1	0

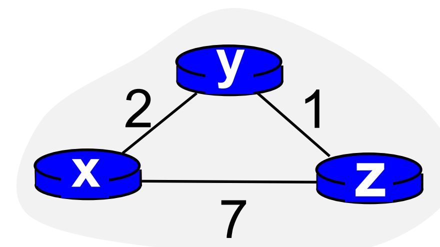
cost to

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

cost to

from	x	y	z

No change: don’t send out DV

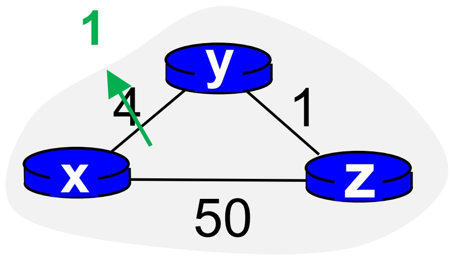


DONE

time

Good news travels fast

1. Updates routing info
2. Recalculates DV
3. If DV changes, notify neighbors



Node detect local link cost change

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*

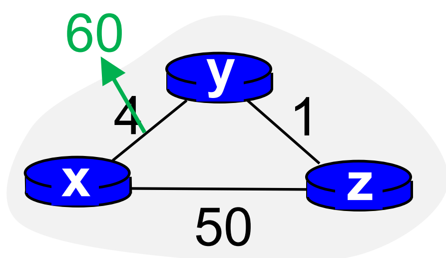
Bad news travels slow

Count to infinity problem

- 44 iterations before algorithm stabilizes

Intuitively

- when z tells y it has a path to x, y has no way of knowing that z is using y on its path



cost to

Y (from)	x	y	z
x	0	4	3
y	4	0	1
z	5	1	0

cost to

Y (from)	x	y	z
x	0	4	3
y	6	0	1
z	5	1	0

3 min: Compute new $D_y(x)$ and $D_z(x)$ after change

$D_y(x) = \min\{c(y,x)+D_x(x),\ c(y,z) + D_z(x)\}$
$= \min\{60+0,\ 1+5\} = 6$

→ Routing Loop

$D_z(x) = \min\{c(z,x) + D_x(x),\ c(z,y) + D_y(x)\}$
$= \min\{50+0\ ,\ 1+6\} = 7$

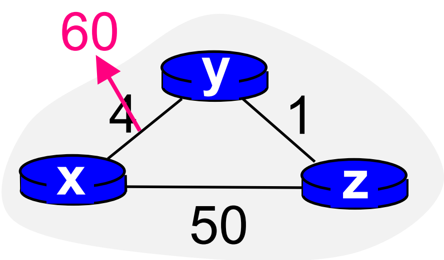
→ Count-to-infinity

Problem arises because y still expects z can get to x with cost of 5

A proposed solution: 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)



Y (from \ cost to)	x	y	z
x	0	4	5
y	4	0	1
z	∞	1	0

$$D_y(x) = \min\{c(y,x)+D_x(x),\ c(y,z)+D_z(x)\} = \min\{60+0,\ 1+\infty\} = 60$$

Q: Will this completely solve count to infinity problem?

- no, only for 2 node loops

Another proposed solution: hold time

- don't process route updates for period of time after route retraction
- ameliorates problem but does not solve

Distance vector routing summary

Easy to implement

- you will implement for hw9 :-)

Distributed

- x doesn't compute paths in isolation
- requires route info (path costs) computed by neighbors

Iterative

- x updates its DV whenever
  - local link costs change
  - DV update received from nbr

Asynchronous

- updates, exchanges happen asynchronously

Self-terminating

- x stops updating DV when no more changes received

Control Plane

LINK STATE VS. DISTANCE VECTOR ROUTING

Comparison

Link state routing

- every node exchanges with every other node in network information about its links to neighbors
- then each node runs Dijkstra's knowing complete graph

Distance vector routing

- every node exchanges with neighbors only its distance estimates to every other node in network
- then each node updates its distance estimates using new estimates from neighbors, then sends its own new estimates to neighbors

Given min cost paths

- can directly compute forwarding table
- forwarding table is used by routers to find next hops for packets
- these min cost paths will need to be periodically recomputed, which can introduce problems

Message complexity

n nodes
E links

Link state

- O(nE) messages sent
  - every node floods its link state message out over every link in network to reach every node
- smaller messages sent to every node
  - message size depends on the number of neighbors a node has
  - any link change requires a broadcast

Distance vector

- # of messages depends on convergence time which varies
  - nodes only exchange messages between neighbors
- larger messages sent only to neighbors
  - message size is proportional to the number of nodes in the network
  - if link changes don't affect shortest path, no message exchange

Speed of convergence

n nodes
E links

Link state

- $\Sigma_{i=1}^{n-1} i = n(n+1)/2 = O(n^2)$
  - search through n-1 nodes to find min, recompute routes
  - search through n-2 nodes to find min, recompute routes
  - …
- converges quickly but may have oscillations
  - route computation is centralized
  - a node stores a complete view of the network

Distance vector

- slow to converge and convergence time varies
  - route computation is distributed
- may be routing loops, count-to-infinity problem

What happens if router malfunctions?

n nodes
E links

Link state

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

Distance vector

- DV node can advertise incorrect path cost
- each node's DV used by others: errors propagate through network

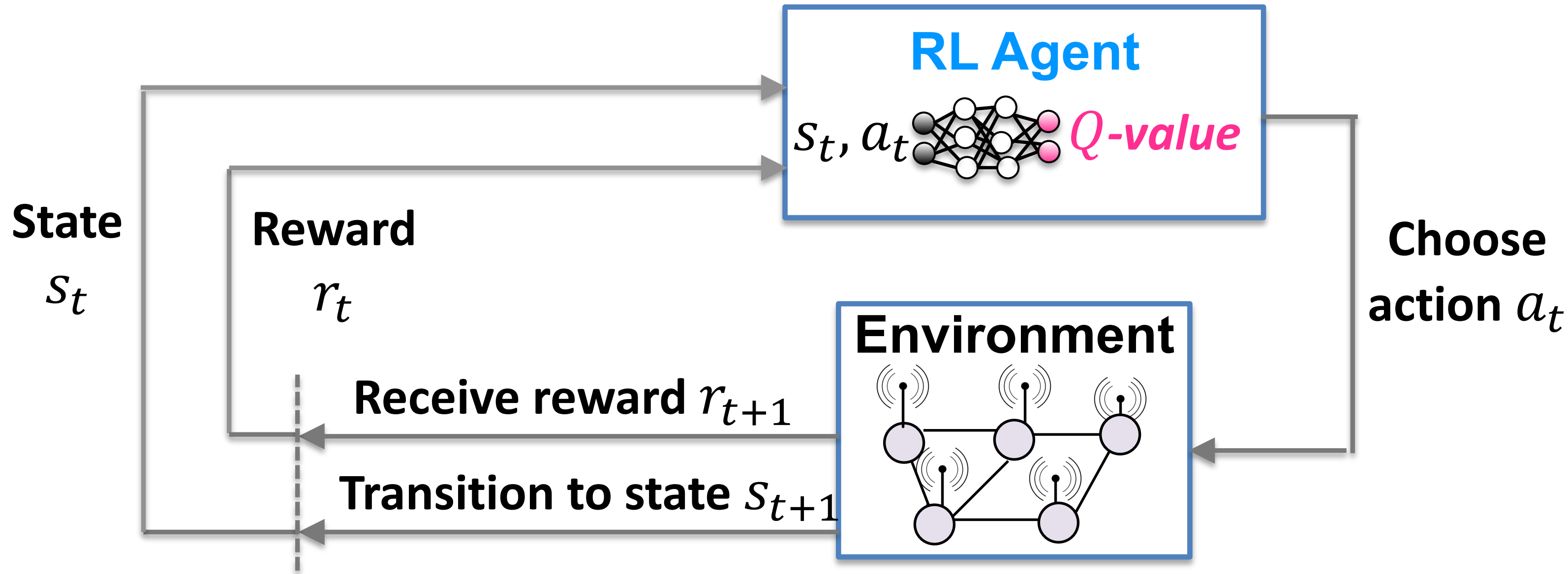
Both have strengths and weaknesses.
One or the other is used in almost every network

Control Plane

OTHER APPROACHES TO MAKE ROUTING DECISIONS

Reinforcement learning to make routing decisions

RL agent learns to choose actions to maximize expected future reward



Define RL agent for routing. Requires us to define *states*, *actions*, and *rewards* useful for routing

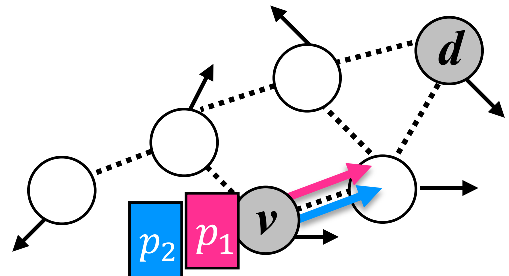
Given trained model install at routers using Software-Defined Networking

Key ideas

1. Packet-centric decisions

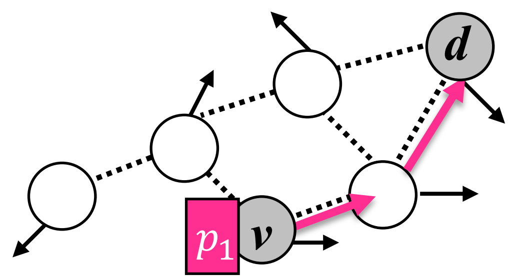
2. Relational features

Problem: Normally a device chooses a packet's next hop … but a device's state doesn't track what happens to the packet



Device v chooses next hop for each outgoing packet

Solution: Use packet agents to simplify s, a, s', r experience sequence and define reward



Packet p_1 chooses its next hop at each device

Key ideas

Problem: How to define generalizable states and actions?

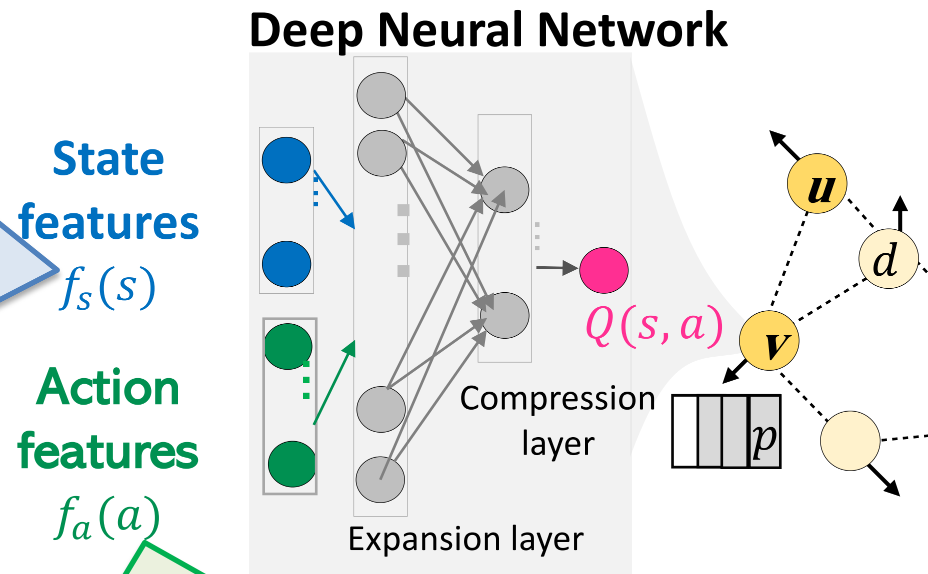
Solution: Use relational features that model the relationship between devices instead of describing a specific device

1. Packet-centric decisions

2. Relational features

For packet p at device v with 1-hop neighbors $Nbr(v)$

- **Packet features** $f_{packet}(p)$: p**'s TTL**
- **Device features** $f_{device}(v, d)$: v**'s** queue length, queue length for packets to d, node degree, node density
- **Neighbor features,** $f_{neighbor}(Nbr(v), p, t)$: summarize varying # of neighbors using min, mean, max of $f_{device}(Nbr(v), p, t)$
- **Path features** $f_{path}(v, d)$: distance or delay from v to d



Packet p separately considers each action u

- **device features** $f_{device}(u, d)$,
- **neighborhood features** $f_{nbrhood}(u, d)$,
- **device features** $f_{device}(u, d)$
- **context features** $f_{context}(p, u)$ which indicate whether p has recently visited u

Internet Control Message Protocol (ICMP)

OVERVIEW

Internet Control Message Protocol (ICMP)

Used by hosts & routers to communicate network-level information

- error reporting
  - unreachable host, network, port, protocol
- echo request/reply
  - used by ping)
- network-layer above IP
  - ICMP msgs carried in IP pkts

Type	Code	Description
0	0	echo reply (ping)
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)
8	0	echo request (ping)
9	0	route advertisement
10	0	router discovery
11	0	TTL expired
12	0	bad IP header

ICMP message

- type, code plus first 8 bytes of IP pkt causing error

Traceroute and ICMP

Source sends series of segments or packets to destination

- first set has TTL =1
- second set has TTL=2, etc.
- unlikely port number

When *n*th set arrives to nth router

- router discards and sends source ICMP message (type 11, code 0)
- ICMP message includes name of router & IP address

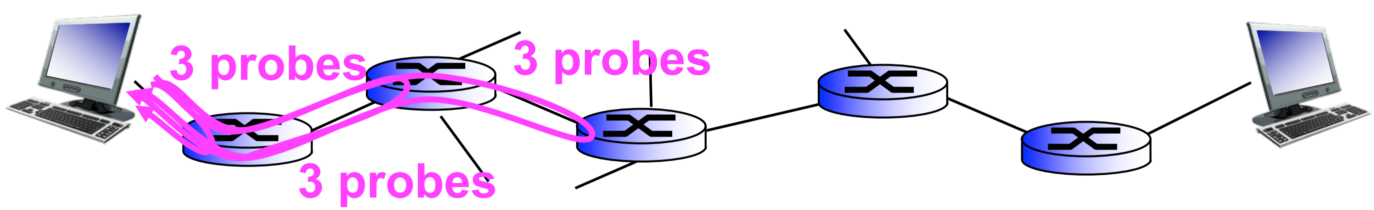
When ICMP msg arrives

- source records RTTs

Stopping criteria

TCP segment or UDP datagram eventually arrives at dst host

- dst returns ICMP "port unreachable" message
- source stops



Q: why can traceroute work with segments, datagrams, or packets?

ICMP traceroute

We're generating an ICMP echo request

Intermediate routers

– respond with ICMP TTL expired

Final destination

– responds with ICMP echo reply