Routing (part 2)

Lecture 24

http://www.cs.rutgers.edu/~sn624/352-F24

Srinivas Narayana

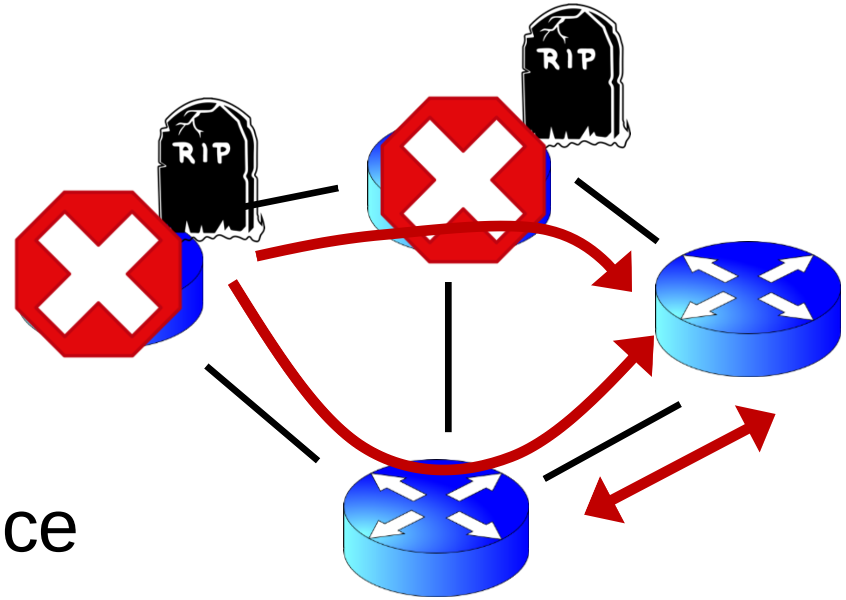
RUTGERS
UNIVERSITY | NEW BRUNSWICK

Routing: Google Maps for the Internet?

Goals of routing:

#1 Good paths

#2 Failure resilience

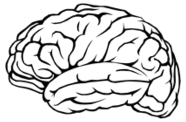


Routing protocols

Link state protocols

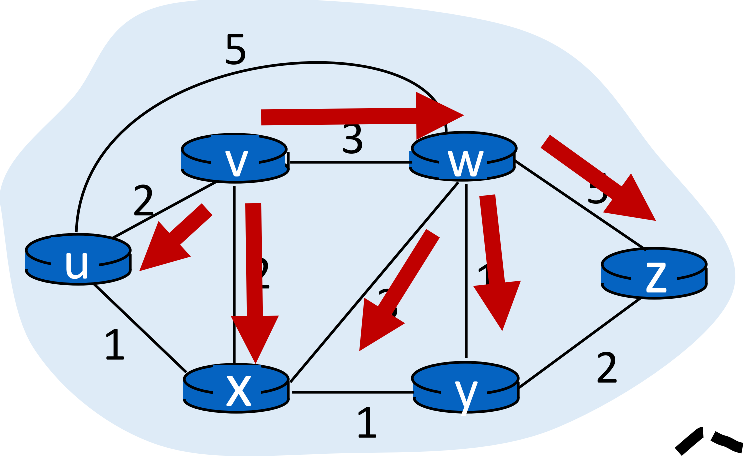
Distance vector protocols

Q1. What info exchanged?



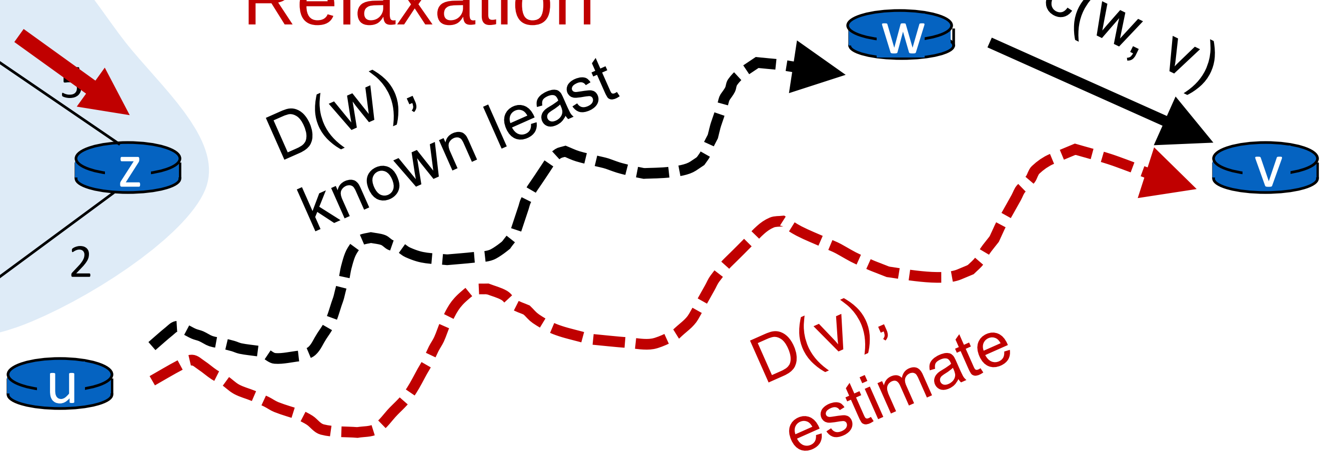
Q2. What computation?

Flooding



Dijkstra’s algorithm

Relaxation



Distance Vector Protocols

Distance Vector Protocol

- Each router only exchanges a distance vector with its neighbors
  - Distance: how far the destination is
  - Vector: a value for each destination
- DVs are only exchanged between neighbors; not flooded

- Use incomplete view of graph derived from neighbors’ distance vectors to compute the shortest paths

Q1. What info exchanged?

Q2. What computation?

Routing protocols

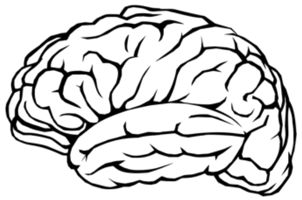
Link state protocols

Distance vector protocols

Q1: Distance Vectors

- $D_x(y)$ = estimate of least cost from x to y
- Distance vector: $\mathbf{D_x} = [D_x(y)\text{: } y \in N]$
- Node x knows cost of edge to each neighbor v: c(x,v)
- Node x maintains $\mathbf{D_x}$
- Node x also maintains its neighbors' distance vectors
  - For each neighbor v, x maintains $\mathbf{D_v} = [D_v(y)\text{: } y \in N]$
- Nodes exchange distance vector periodically and whenever the local distance vector changes (e.g., link failure, cost changes)

Q2: Algorithm



Bellman-Ford algorithm

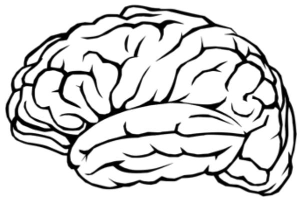
- Each node initializes its own distance vector (DV) to edge costs
- Each node sends its DVs to its neighbors
- When a node `x` receives new DV from a neighbor `v`, it updates its own DV using the Bellman-Ford equation:
- Given $d_x(y)$:= estimated cost of the least-cost path from x to y
- Update $d_x(y) = \min_v \{c(x,v) + d_v(y)\}$

cost of path from neighbor v to destination y

minimum taken over
all neighbors v of x

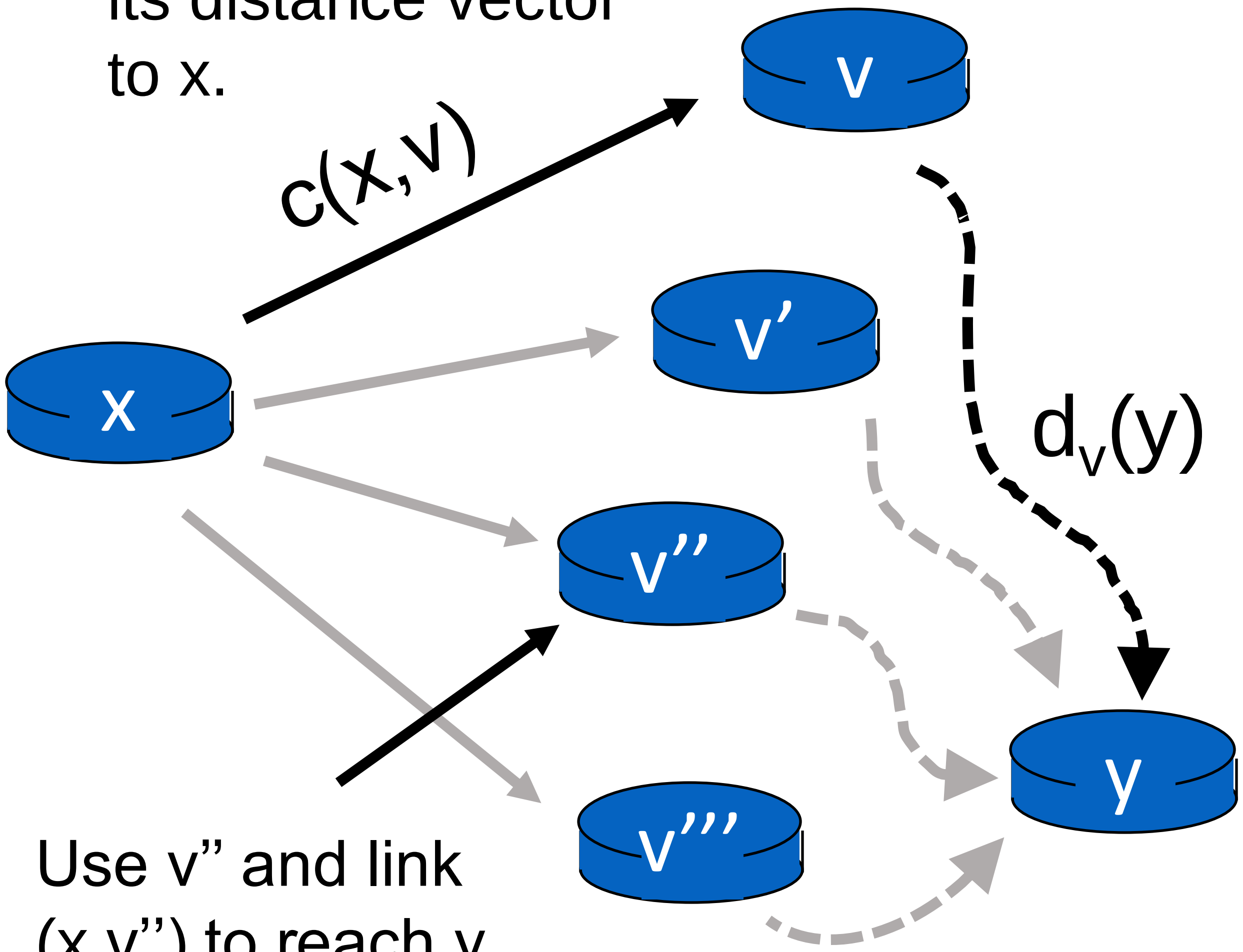
cost to reach neighbor v directly from x

Visualization



Neighbor v sends its distance vector to x.

- Which neighbor v offers the current best path from x to y?
- Path through neighbor v has cost c(x,v) + $d_v(y)$
- Choose min-cost neighbor
- Remember min-cost neighbor as the one used to reach node y
- This neighbor determines the output port!



Use v’’ and link (x,v’’) to reach y.

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

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

node x table

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

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

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

node y table

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

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

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

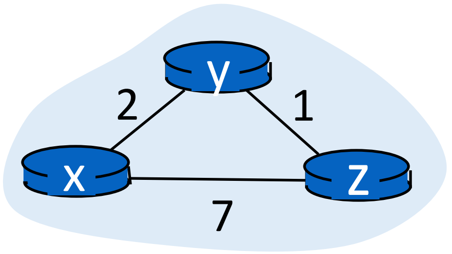
node z table

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

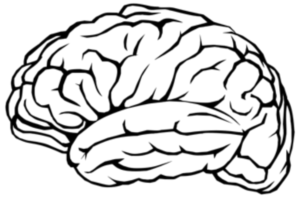
from \ cost to	x	y	z
x	0	2	7
y	2	0	1
z	3	1	0

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

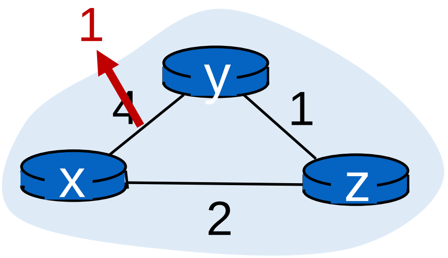
time



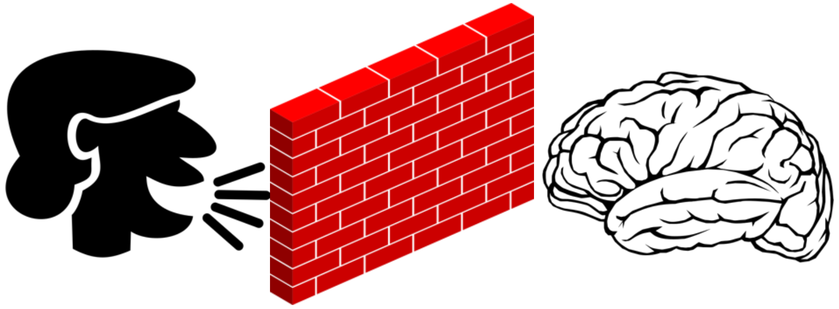
Good news travels fast



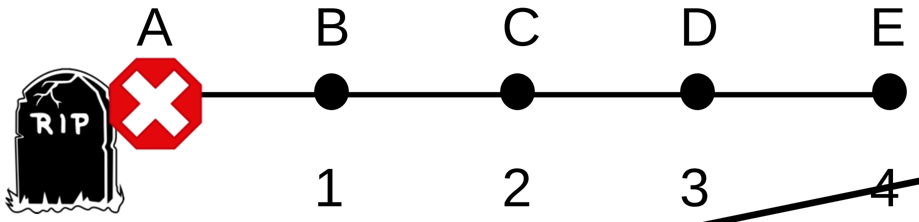
- Suppose the link cost reduces or a new better path becomes available in a network.
- The immediate neighbors of the change detect the better path immediately
- Since their DV changed, these nodes notify their neighbors immediately.
    - And those neighbors notify still more neighbors
    - … until the entire network knows to use the better path
- Good news travels fast through the network
- This is despite messages only being exchanged among neighbors



Bad news travels slowly



- If router goes down, could be a while before network realizes it.



1	2	3	4	Initially
3	2	3	4	After 1 exchange
3	4	3	4	After 2 exchanges
5	4	5	4	After 3 exchanges
5	6	5	6	After 4 exchanges
7	6	7	6	After 5 exchanges

B still thinks it can reach A through C… bad!

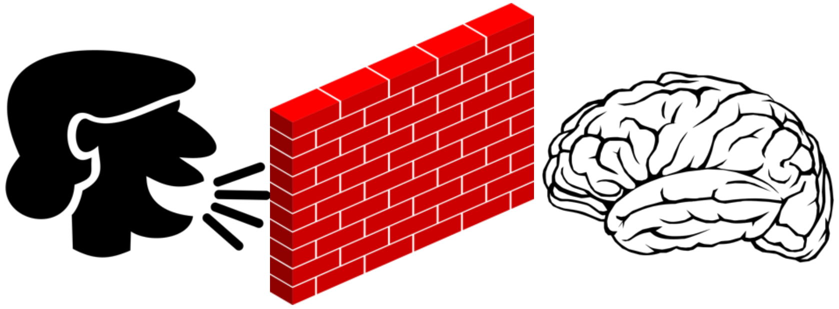
C thinks it can reach A through B… worse!

B, D think they can reach A through C… ugly!

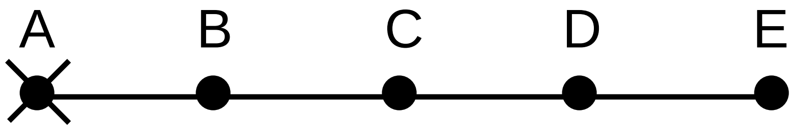
Count to infinity problem

etc… to infinity

Bad news travels slowly



- Reacting appropriately to bad news requires information that only other routers have. DV does not exchange sufficient info.



- B needs to know that C has no other path to A other than via B.
- DV does not exchange paths; just distances!
- Poisoned reverse: if X gets its route to Y via Z, then X will announce $d_X(Y) = \infty$ in its message to Z
  - Effect: Z won't use X to route to Y
  - However, this won't solve the problem in general (think why.)

Summary: Comparison of LS and DV

Link State Algorithms

- Nodes have full visibility into the network's graph
- Copious message exchange: each LSA is flooded over the whole network
- Robust to network changes and failures

OSPF
Open Shortest Path First
(v2 RFC 2328)

Distance Vector Algorithms

- Only distances and neighbors are visible
- Sparse message exchange: DVs are exchanged among neighbors only
- Brittle to router failures. Incorrect info may propagate all over net

EIGRP
Enhanced Interior Gateway Routing Protocol
(RFC 7868)

Routing protocols

Link state protocols

Distance vector protocols

Every router is aware of the existence of every other router.

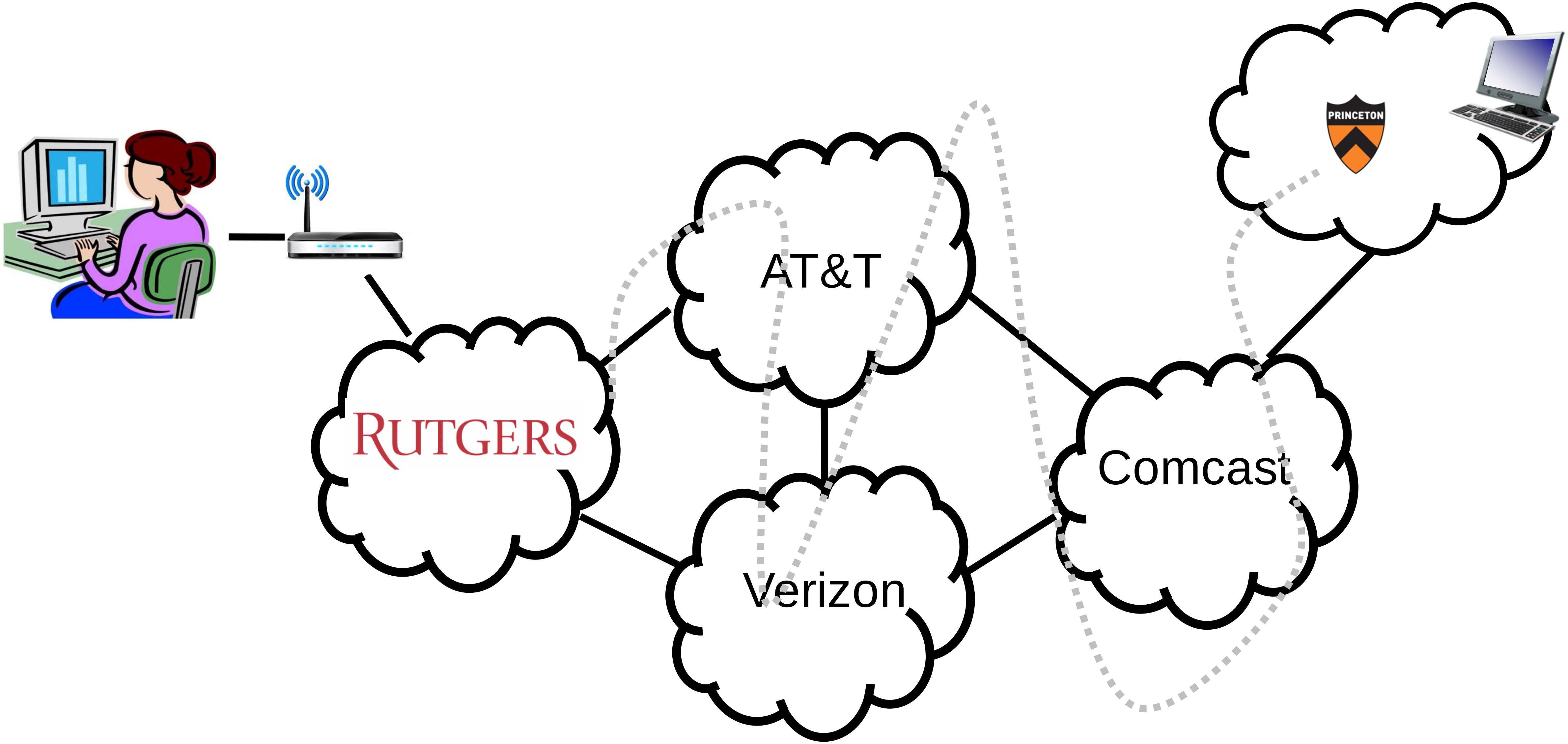
Messages reveal information on the full network (graph) structure.

Message exchange and forwarding tables scale with network size.

These assumptions/settings cannot work on the Internet.

Internet Routing

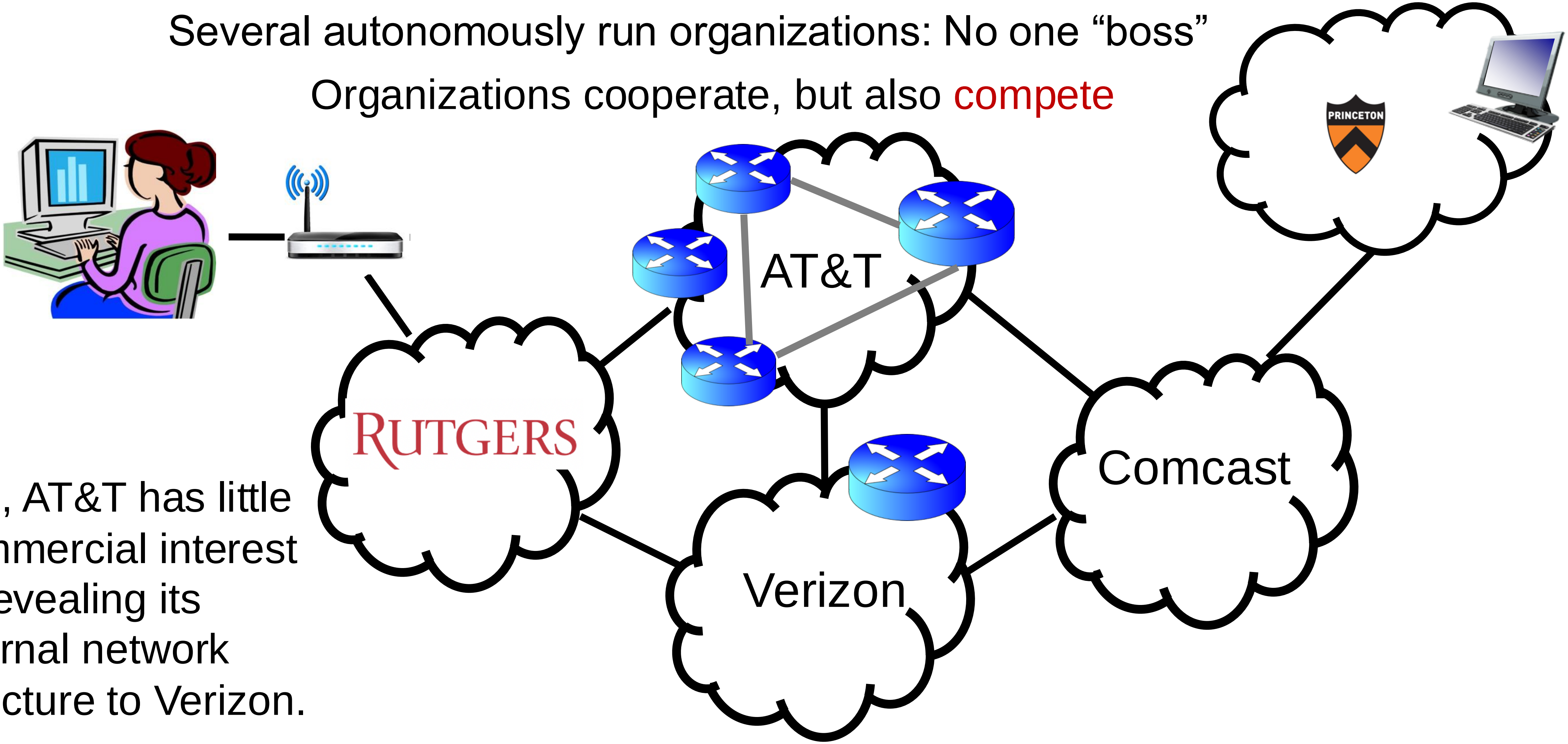
The Internet is a large federated network



The Internet is a large federated network

Several autonomously run organizations: No one “boss”

Organizations cooperate, but also compete

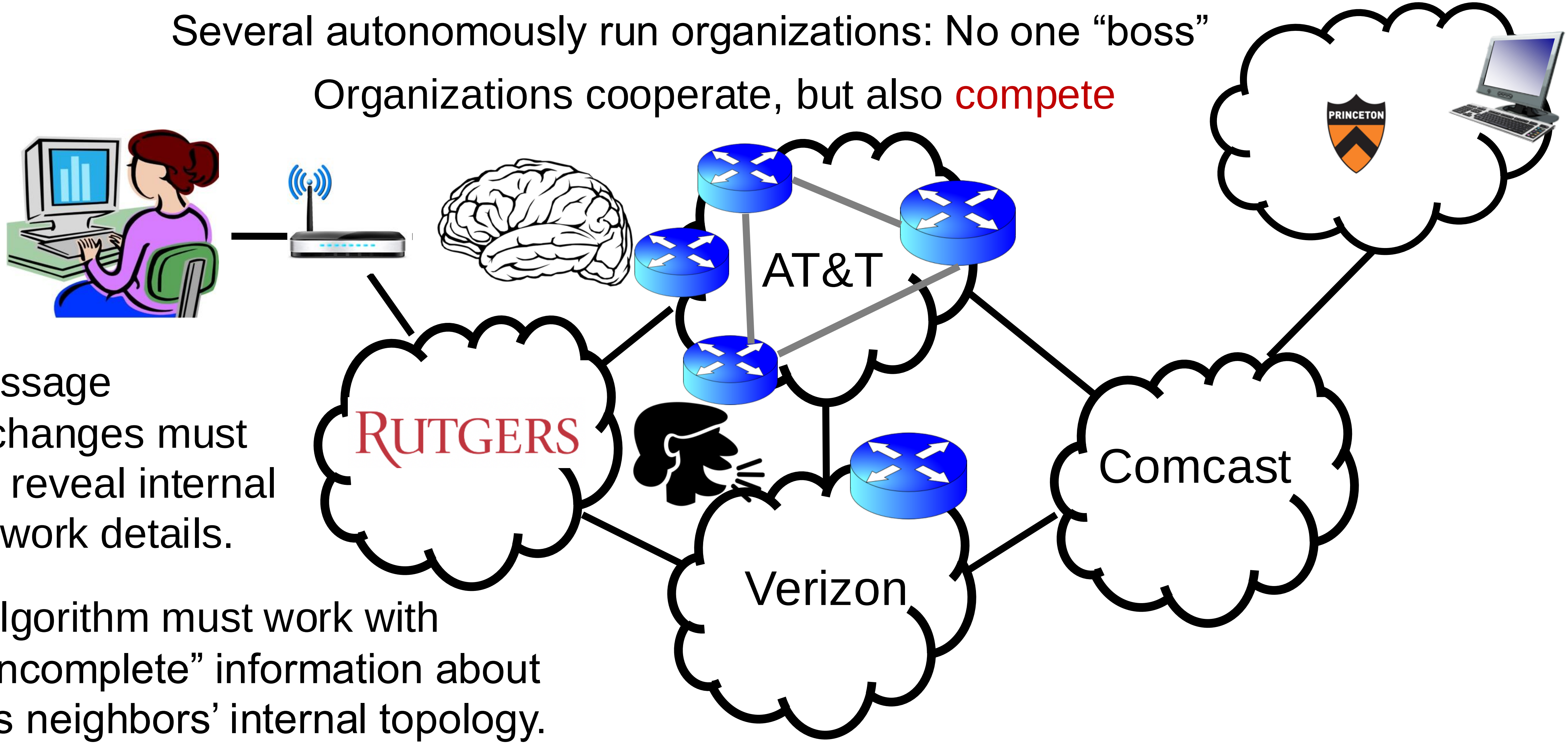


e.g., AT&T has little commercial interest in revealing its internal network structure to Verizon.

The Internet is a large federated network

Several autonomously run organizations: No one “boss”

Organizations cooperate, but also compete



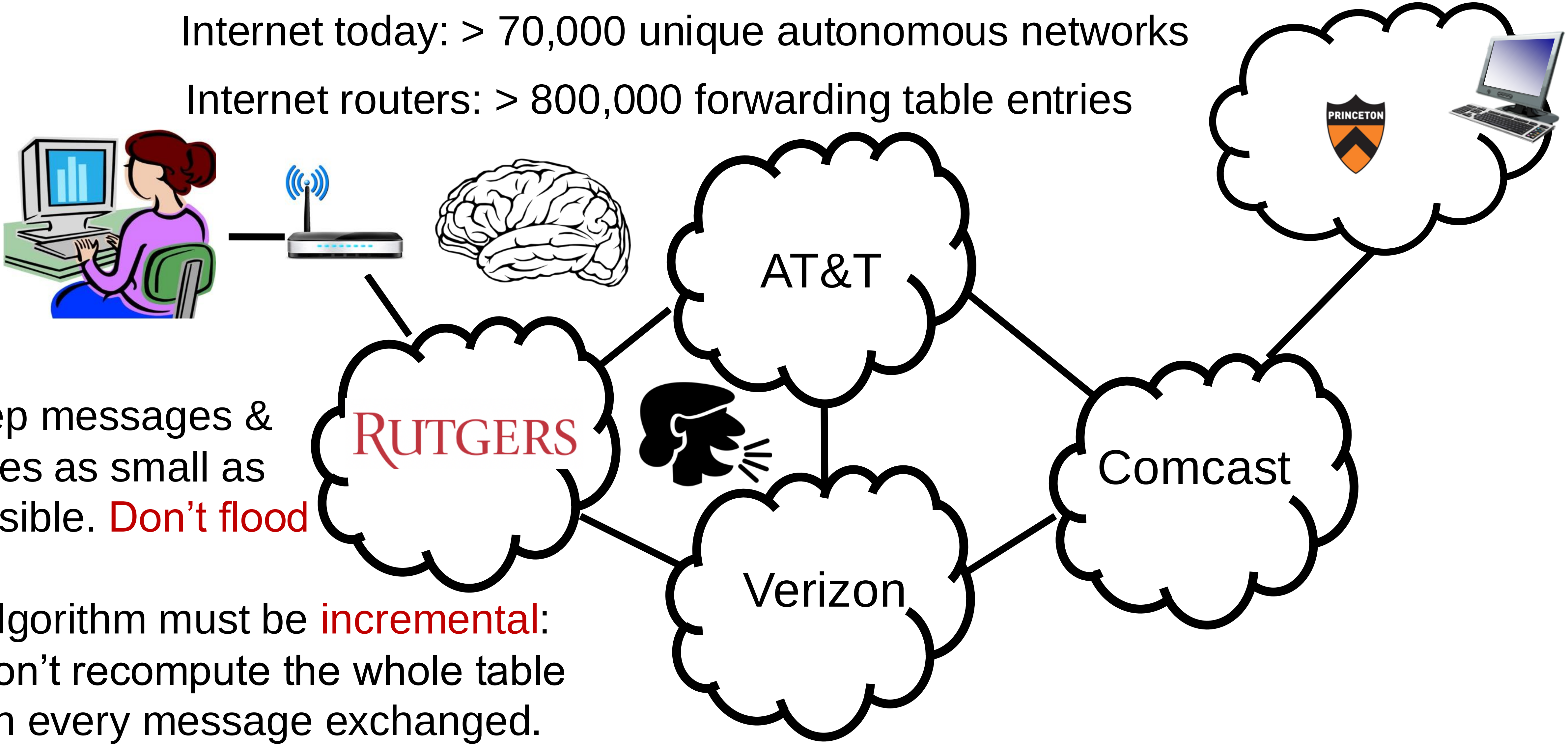
Message exchanges must not reveal internal network details.

Algorithm must work with “incomplete” information about its neighbors’ internal topology.

The Internet is a large federated network

Internet today: > 70,000 unique autonomous networks

Internet routers: > 800,000 forwarding table entries



Keep messages & tables as small as possible. Don’t flood

Algorithm must be incremental: don’t recompute the whole table on every message exchanged.