Internet and Web Architecture

A review

Lecture 2

Srinivas Narayana

http://www.cs.rutgers.edu/~sn624/553-S23

RUTGERS
UNIVERSITY | NEW BRUNSWICK

Outline

- Name resolution
- The HTTP protocol
- Socket abstraction
- Underlying transport concerns: reliability, basic congestion control
- Internet routing: IP address organization, BGP, and concerns
- CDN reading : Tom Leighton, Akamai full detailed study
- Relevant points from Google, FB, Microsoft edge and peering papers
- HTTP/TCP interaction?

Software/hardware organization at hosts

Application: useful user-level functions

Transport: provide guarantees to apps

Network: best-effort global pkt delivery

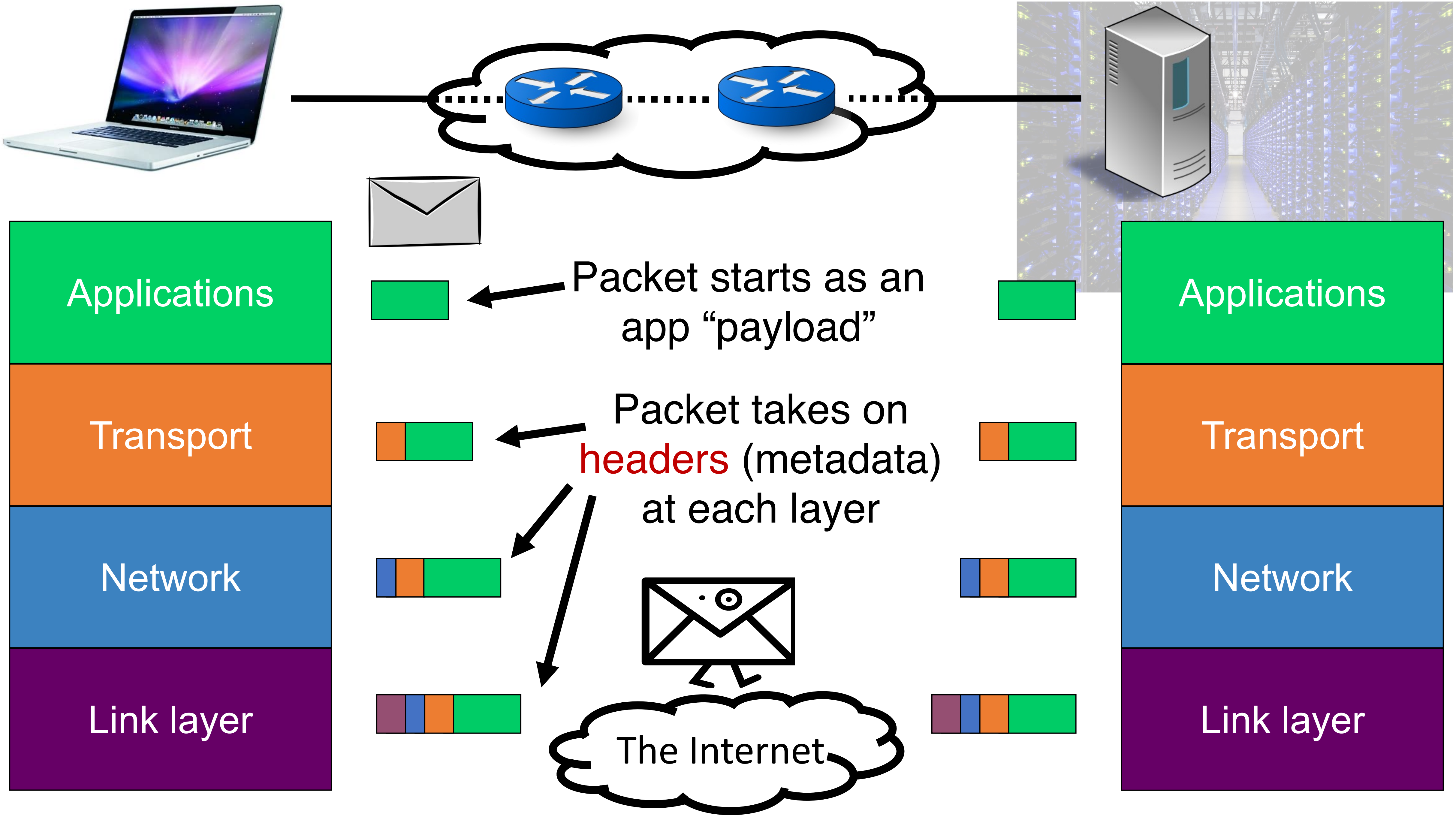
Link: best-effort local pkt delivery

Communication functions broken up and “stacked”

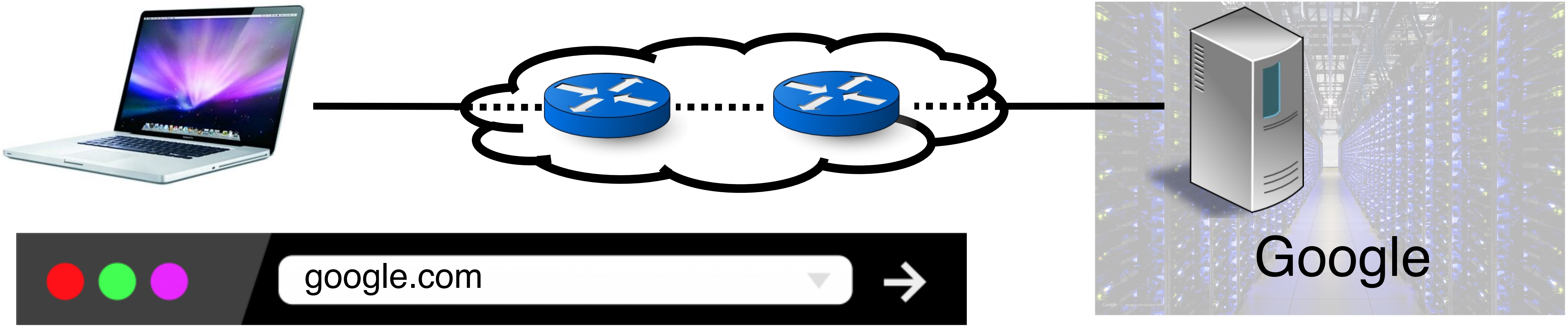
Each layer depends on the one below it.

Each layer supports the one above it.

The interfaces between layers are well-defined and standardized.


Applications
Transport
Network
Link layer
Packet starts as an app "payload"
Packet takes on headers (metadata) at each layer
The Internet
Applications
Transport
Network
Link layer

Name Resolution



Machines communicate using IP addresses and ports
IP addresses: ~12 digits (IPv4) or more
Ports: fixed based on application (e.g., 80: web)

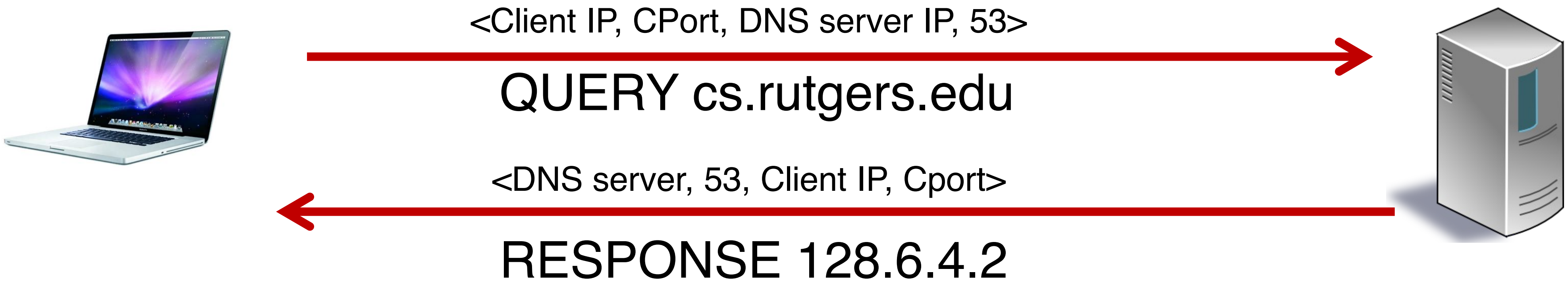
Need a way to turn human-readable addresses into Internet addresses.

Ask someone	Ask everyone	Tell everyone
Directory service	Query broadcast	Information flooding

Asking "someone" could involve asking many machines…

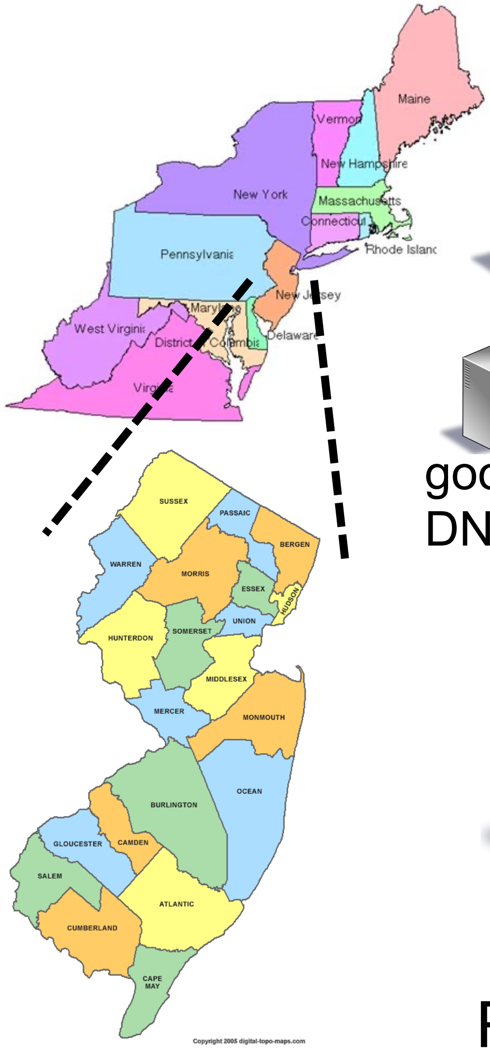
Domain Name Service

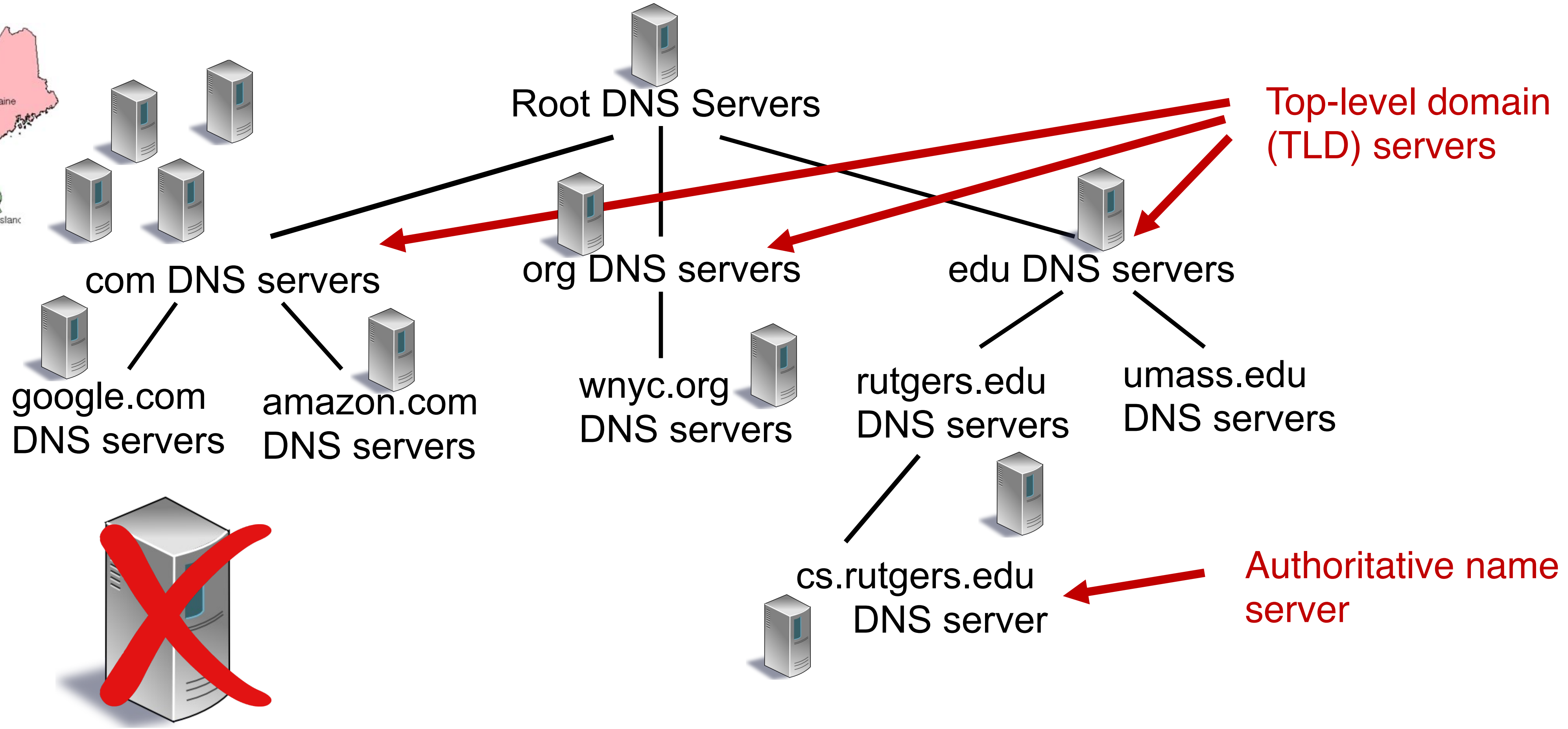
DOMAIN NAME	IP ADDRESS
spotify.com	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86



- Key idea: Implement a server that looks up a table.
- Will this scale?
  - Every new (changed) host needs to be (re)entered in this table
  - Performance: can the server serve billions of Internet users?
  - Failure: what if the server or the database crashes?
  - Security: What if someone “takes over” this server?

Distributed and hierarchical database





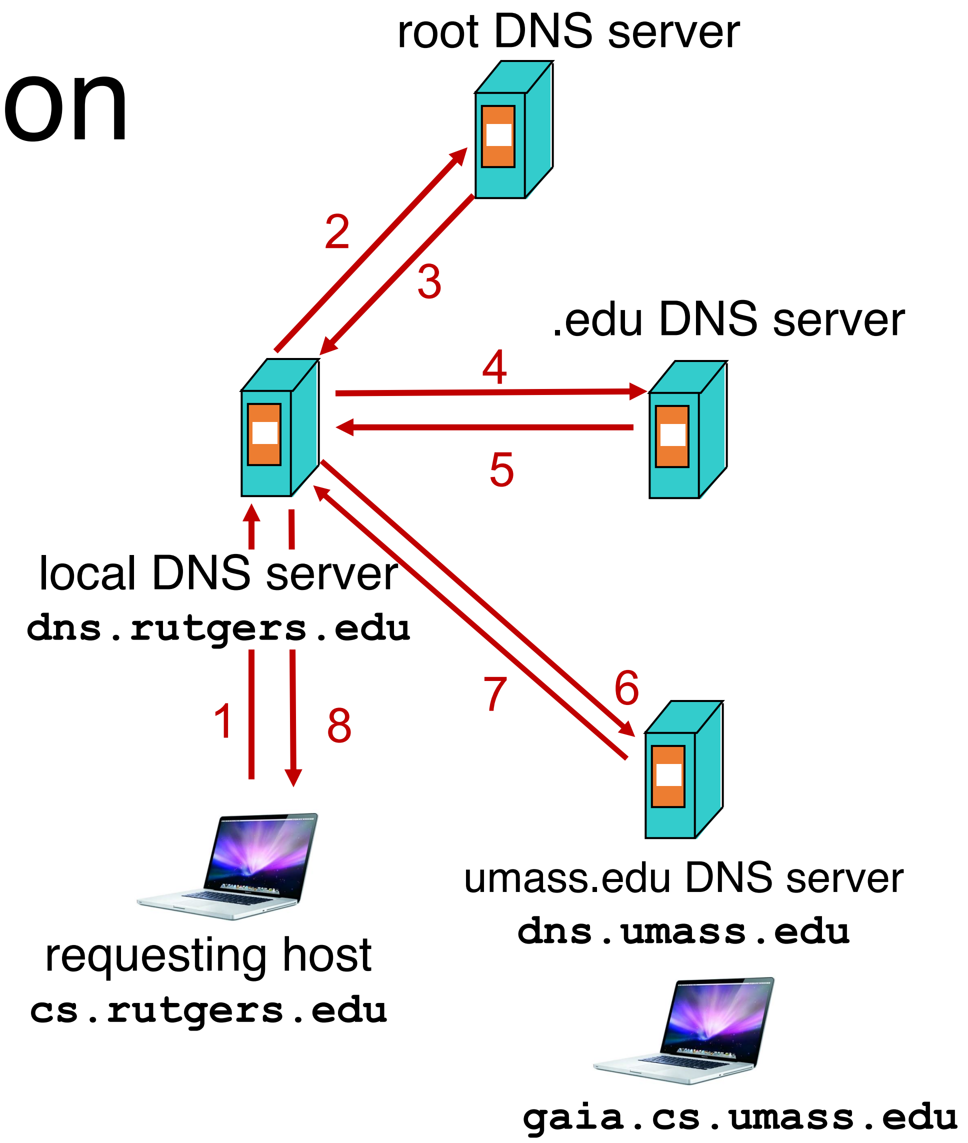
RFC 1034

Hierarchy Replication Indirection

DNS name resolution

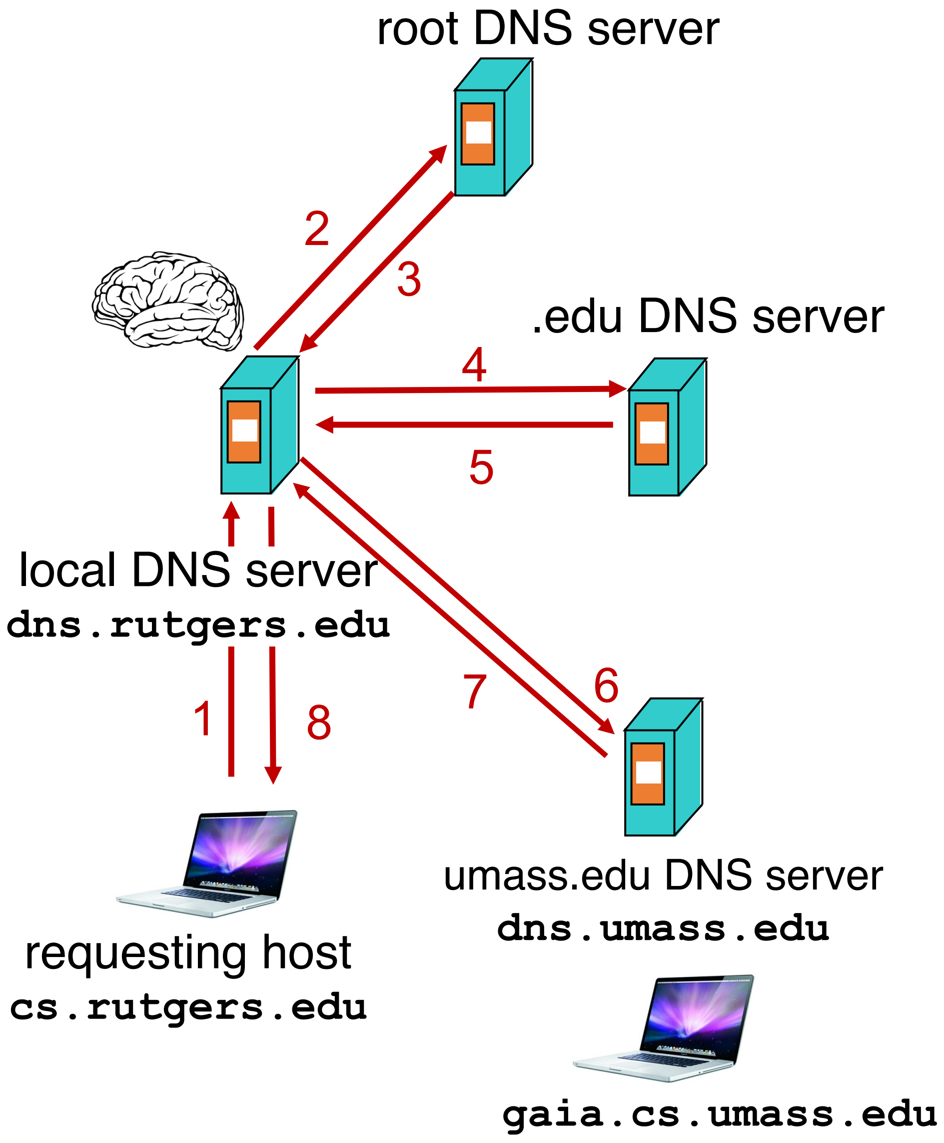
- Host at cs.rutgers.edu wants IP address for gaia.cs.umass.edu

- Local DNS server
- Root DNS server
- TLD DNS server
- Authoritative DNS server



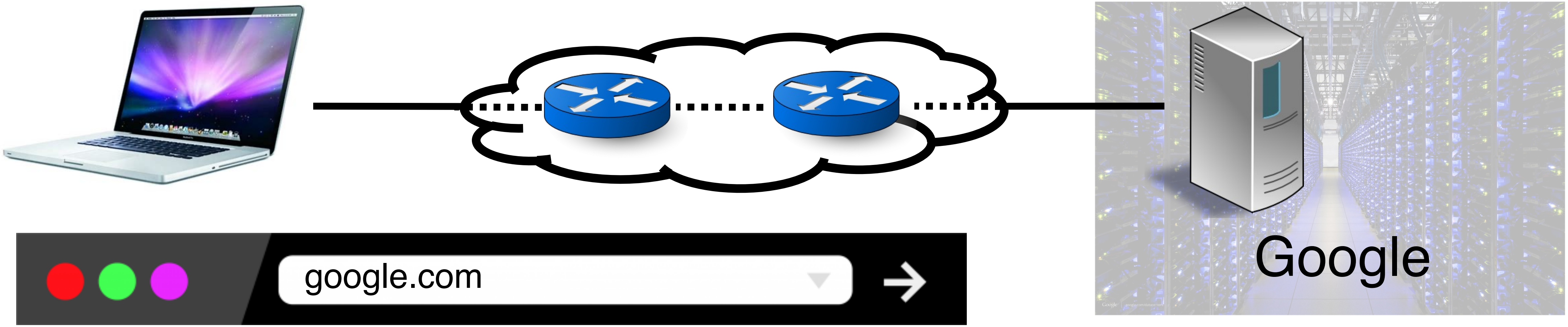
DNS caching

- Once (any) name server learns a name to IP address mapping, it *caches* the mapping
- Cache entries timeout (disappear) after some time
- TLD servers typically cached in local name servers
- In practice, root name servers aren't visited often!
- Caching is pervasive in DNS



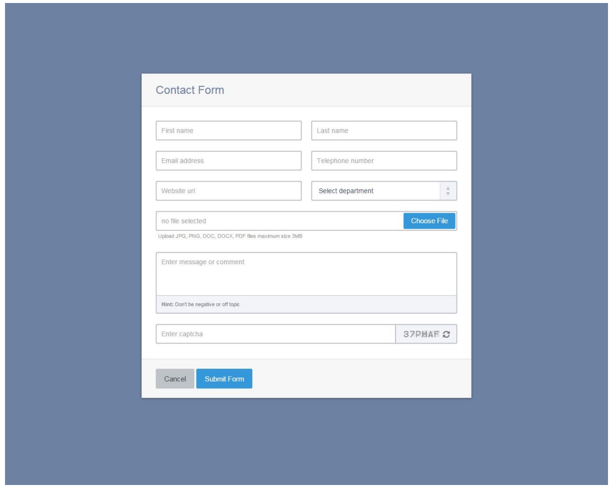
Example DNS interactions

- `dig <domain-name>`
- `dig +trace <domain-name>`
- `dig @<dns-server> <domain-name>`



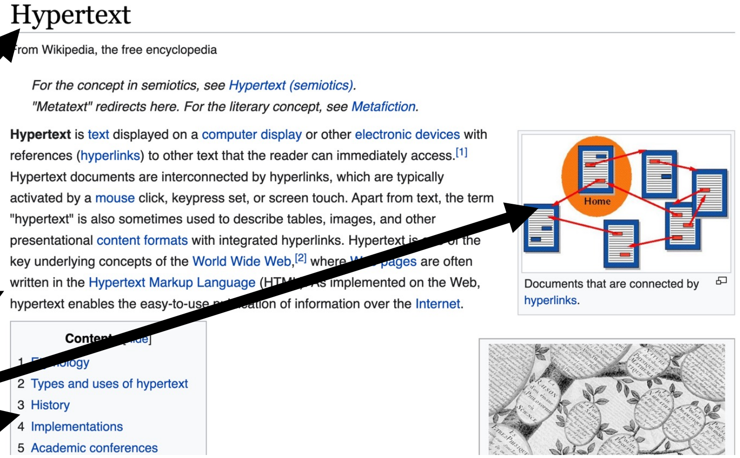
The web is a *specific* application protocol running over a network: HyperText Transfer Protocol (HTTP)

Each object addressable by a name (URL)



Named objects can be static (image, video)

… or the result of a dynamic app process


Hypertext

rom Wikipedia, the free encyclopedia

For the concept in semiotics, see Hypertext (semiotics).
"Metatext" redirects here. For the literary concept, see Metafiction.

Hypertext is text displayed on a computer display or other electronic devices with references (hyperlinks) to other text that the reader can immediately access.[1] Hypertext documents are interconnected by hyperlinks, which are typically activated by a mouse click, keypress set, or screen touch. Apart from text, the term "hypertext" is also sometimes used to describe tables, images, and other presentational content formats with integrated hyperlinks. Hypertext is … of the key underlying concepts of the World Wide Web,[2] where … pages are often written in the Hypertext Markup Language (HTM… implemented on the Web, hypertext enables the easy-to-use … ation of information over the Internet.

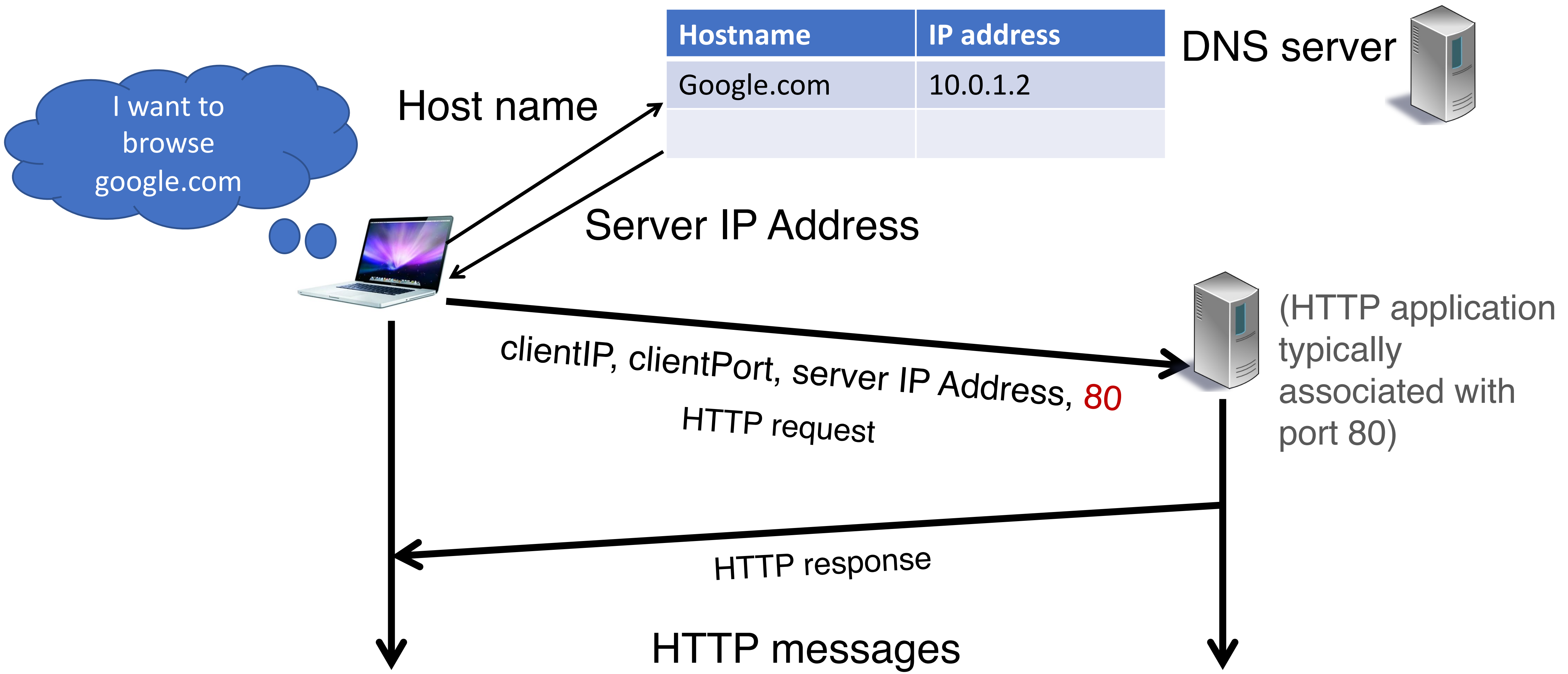
Documents that are connected by hyperlinks.

Content…
1 …ology
2 Types and uses of hypertext
3 History
4 Implementations
5 Academic conferences

Objects

Web interactions

Hostname	IP address
Google.com	10.0.1.2



Example HTTP interactions

- `wget google.com (or) curl google.com`

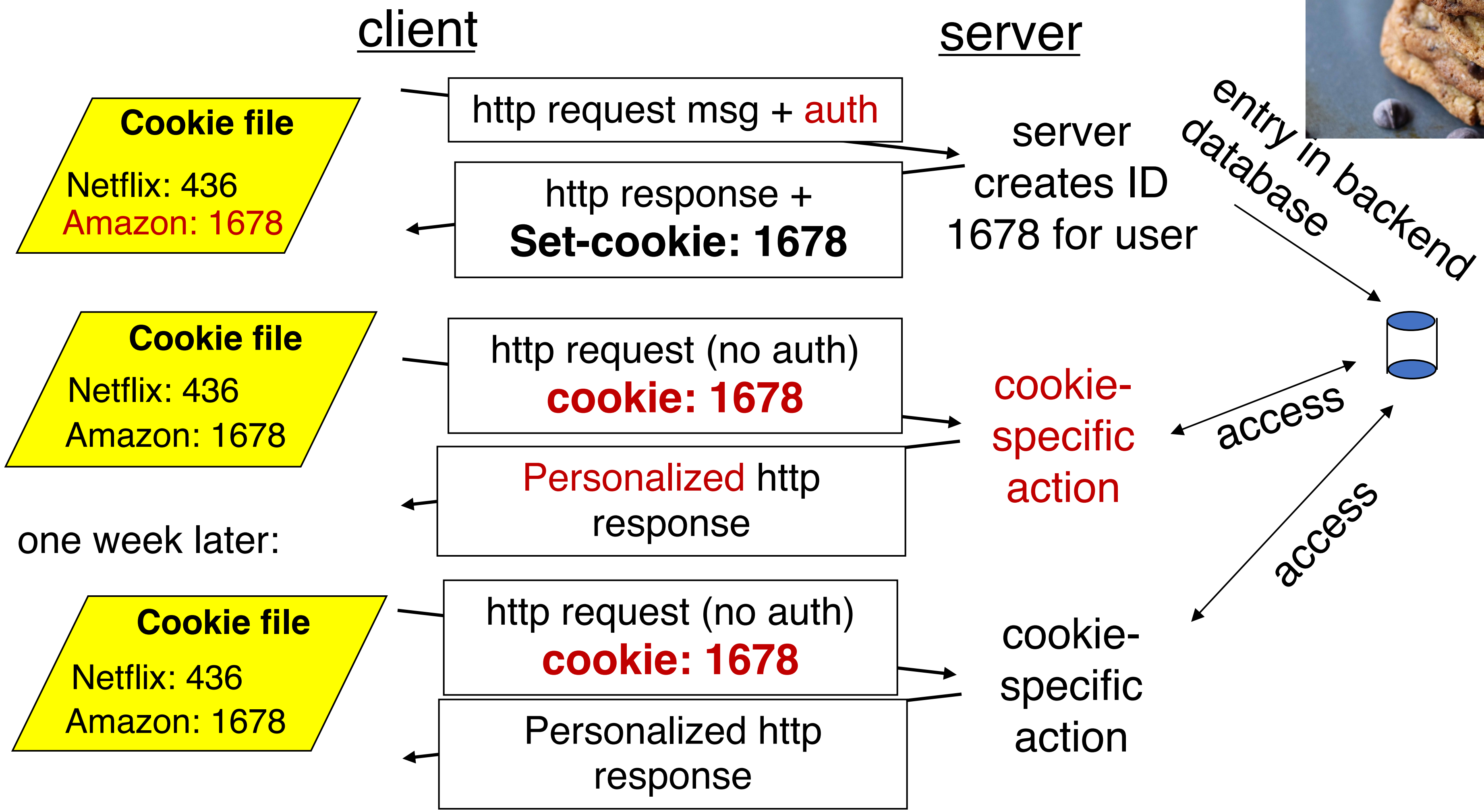
- `telnet example.com 80`
  - `GET / HTTP/1.1`
  - `Host: example.com`

(followed by two enter's)

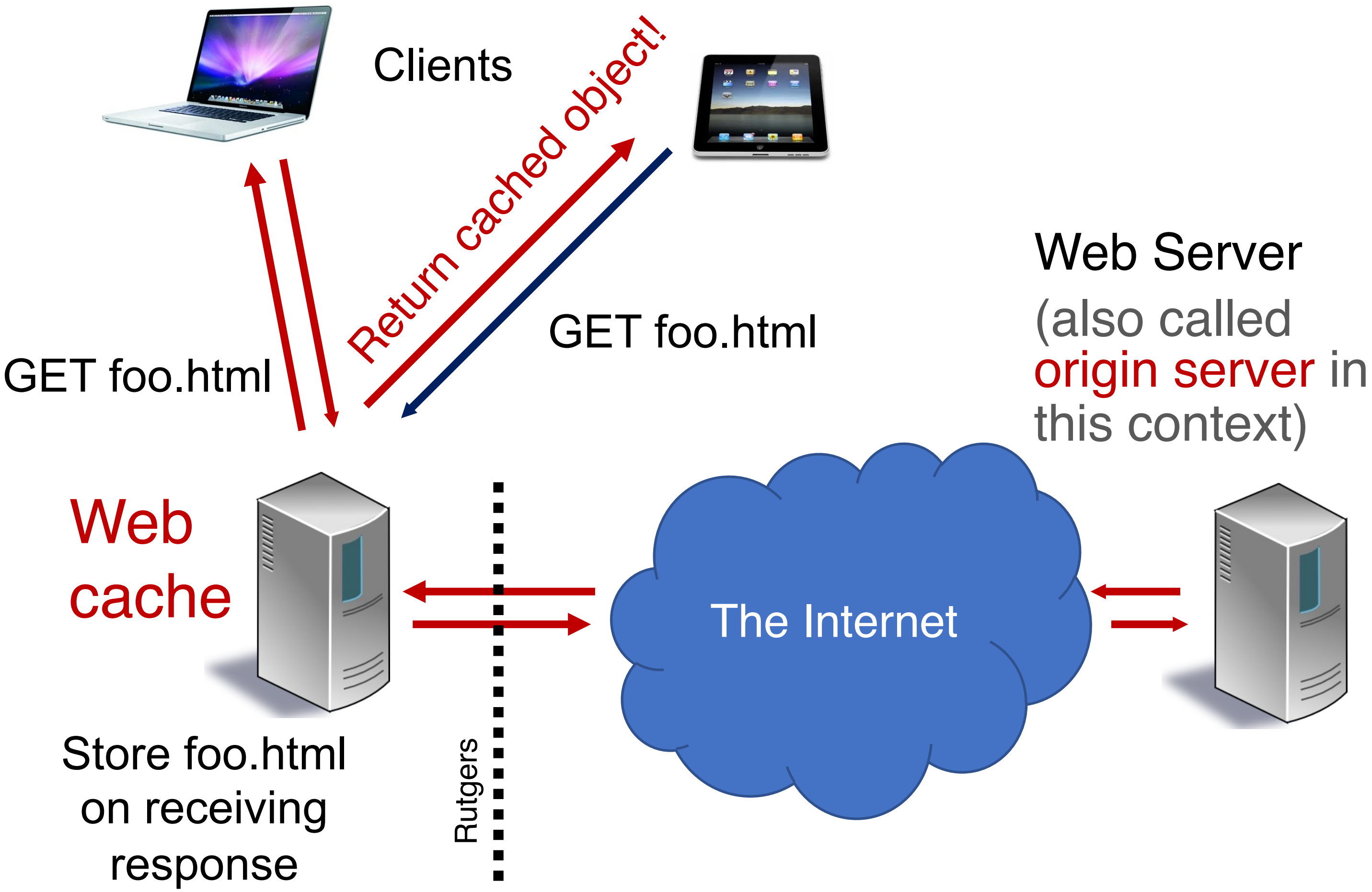
- Exercise: try
  - `telnet google.com 80`
  - `telnet web.mit.edu 80`

Remembering users: cookies

Cookie is typically opaque to client.



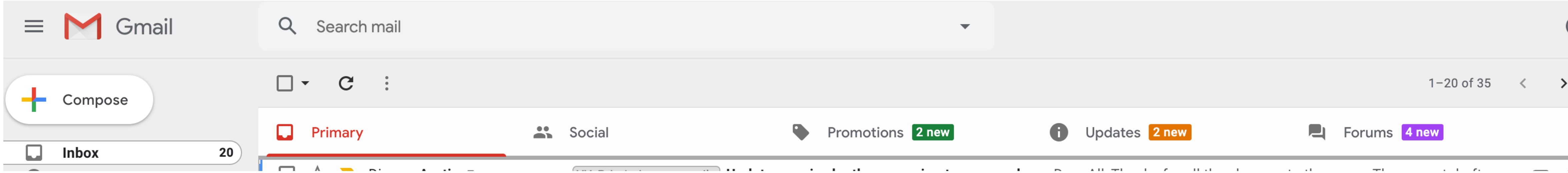
Improving performance: Web caching



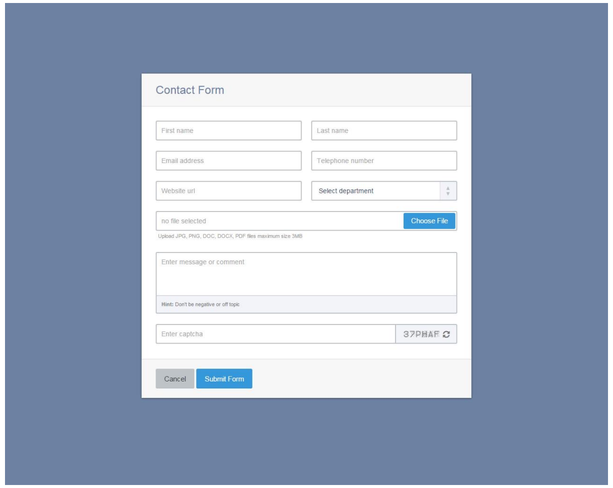
- Network administrators (e.g., Rutgers) may run web caches to remember popular web objects
- Hit: cache returns object
- Miss: obtain object from originating web server (origin server) and return to client
  - Also cache the object locally
- Reduce response time
- Reduce traffic requirements (and $$) on an organization's network connections

Not all content is effectively cacheable

- Personalized content

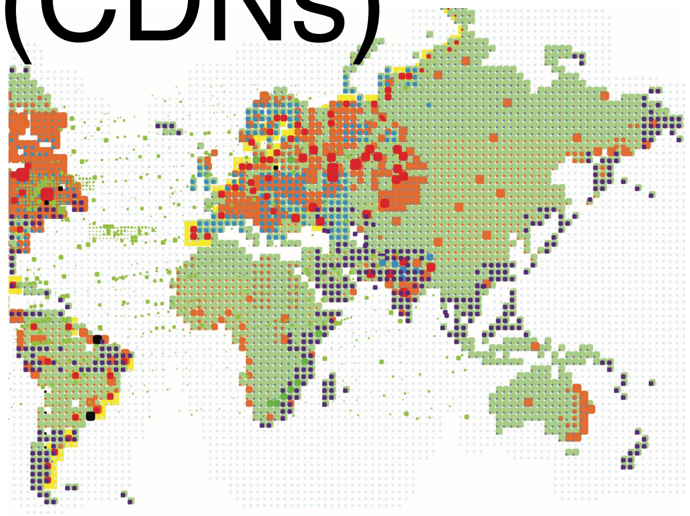


- Interactive processing
  - e.g., forms, shopping carts, ajax, etc.



- Long tail of (obscure) content

Content Distribution Networks (CDNs)



A global network of web caches

- Provisioned by ISPs and network operators
- Or content providers, like Netflix, Google, etc.

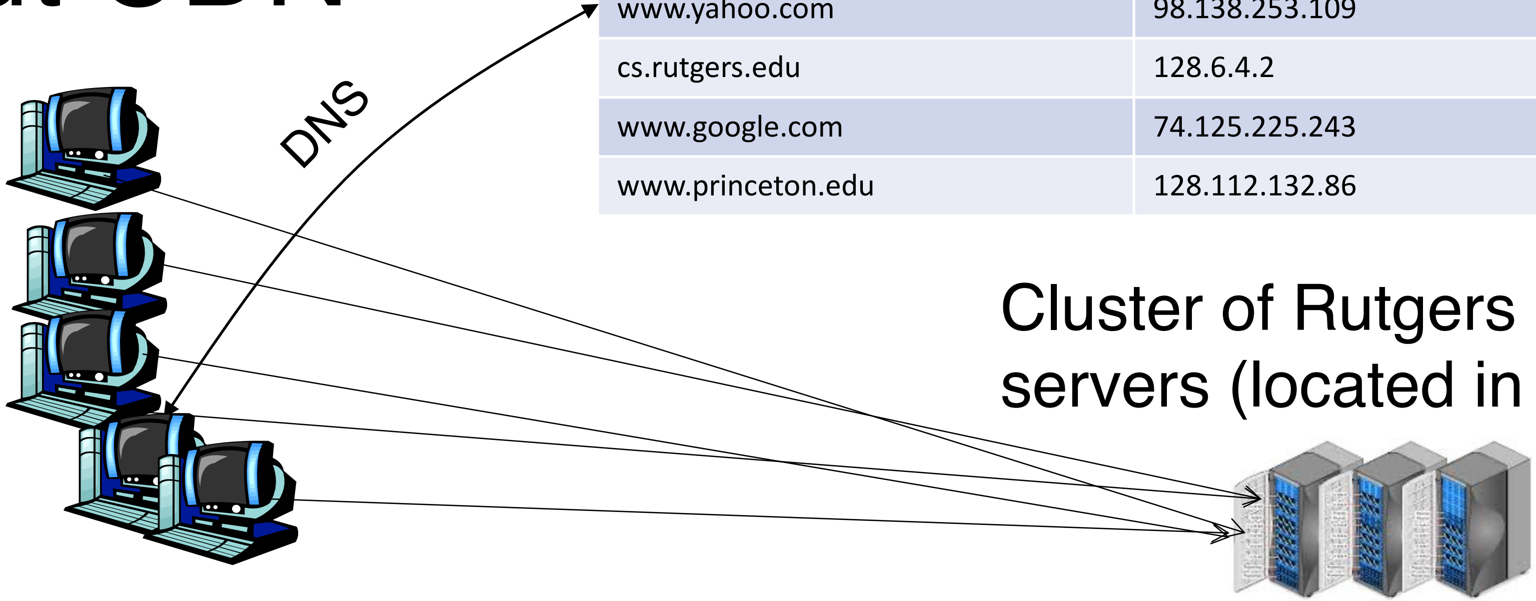
Uses

- Reduce traffic on a network's Internet connection, e.g., Rutgers
- Improve response time for users: CDN nodes are closer to users than origin servers (servers holding original content)
- Reduce bandwidth requirements on content provider
- Reduce $$ to maintain origin servers

Without CDN

DOMAIN NAME	IP ADDRESS
www.yahoo.com	98.138.253.109
cs.rutgers.edu	128.6.4.2
www.google.com	74.125.225.243
www.princeton.edu	128.112.132.86

Clients distributed all over the world



Cluster of Rutgers CS origin servers (located in NJ, USA)

128.6.4.2

- Problems:
- Huge bandwidth requirements for Rutgers
- Large propagation delays to reach users

Where the CDN comes in

- Distribute content of the origin server over geographically distributed CDN servers

- But how will users get to these CDN servers?

- Use DNS!
  - DNS provides an additional layer of indirection
  - Instead of returning IP address, return another DNS server (NS record)
  - The second DNS server (run by the CDN) returns IP address to client

- The CDN runs its own DNS servers (CDN name servers)
  - Custom logic to send users to the "closest" CDN web server

With CDN

DOMAIN NAME	IP ADDRESS
www.yahoo.com	98.138.253.109
cs.rutgers.edu	124.8.9.8 (NS record pointing to CDN name server)
www.google.com	74.125.225.243

DNS reply

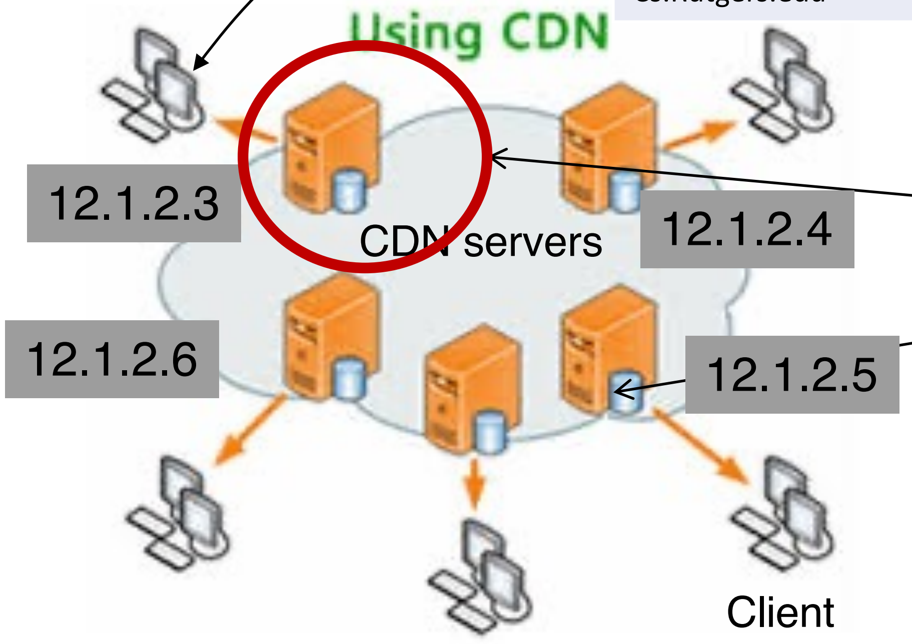
NS record delegates the choice of IP address to the CDN name server.

CDN Name Server (124.8.9.8)

DOMAIN NAME	IP ADDRESS
Cs.Rutgers.edu	12.1.2.3
Cs.Rutgers.edu	12.1.2.4
Cs.Rutgers.edu	12.1.2.5
Cs.Rutgers.edu	12.1.2.6

Custom logic to map ONE domain name to one of many IP addresses!

Popular CDNs:
CloudFlare
Akamai
Level3
…



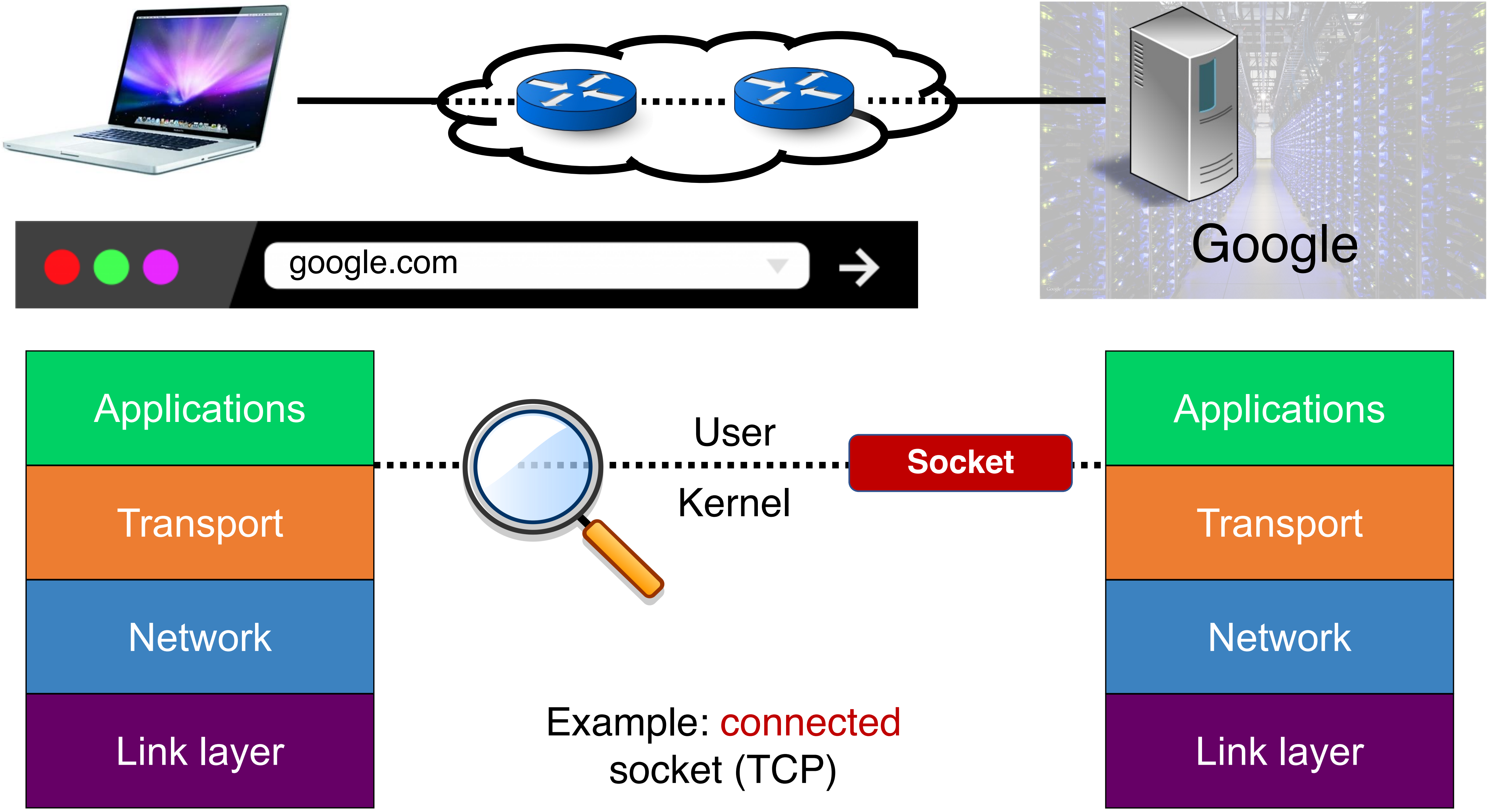
128.6.4.2
Origin server

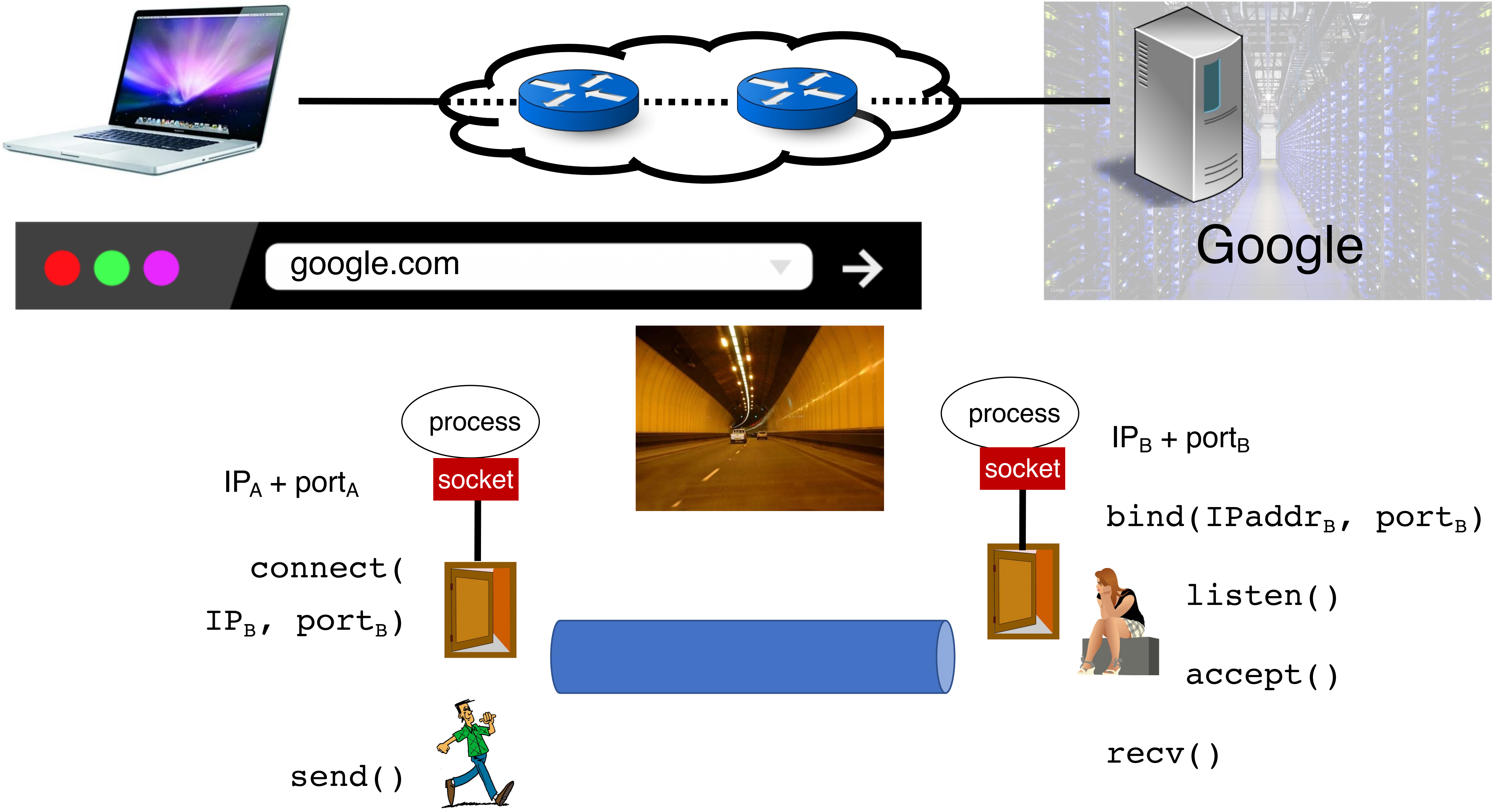
Most requests go to CDN servers (caches).
CDN servers may request object from origin
Few client requests go directly to origin server

Seeing a CDN in action

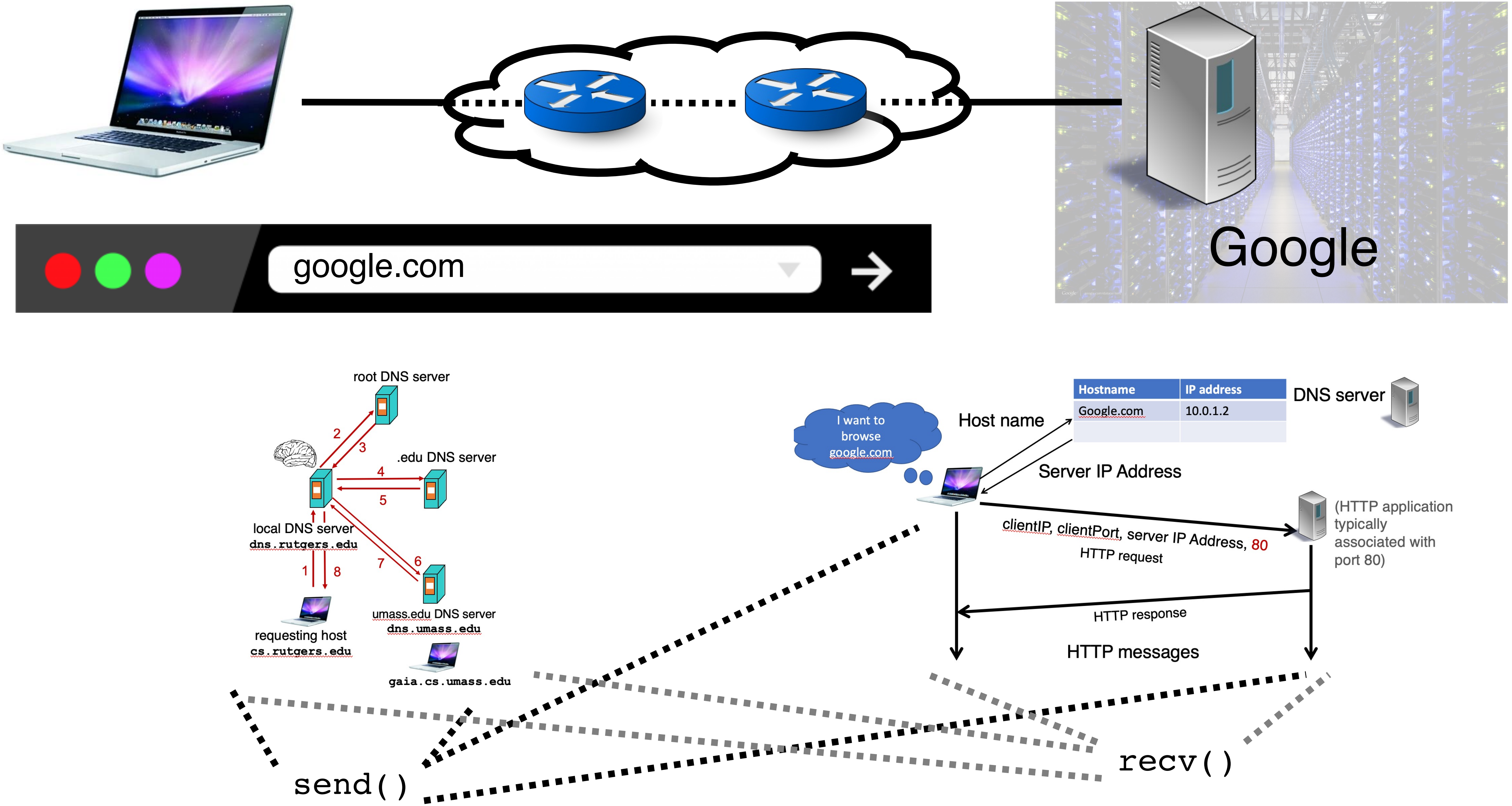
- `dig web.mit.edu` (or) `dig +trace web.mit.edu`
- `telnet web.mit.edu 80`

Application-OS interface

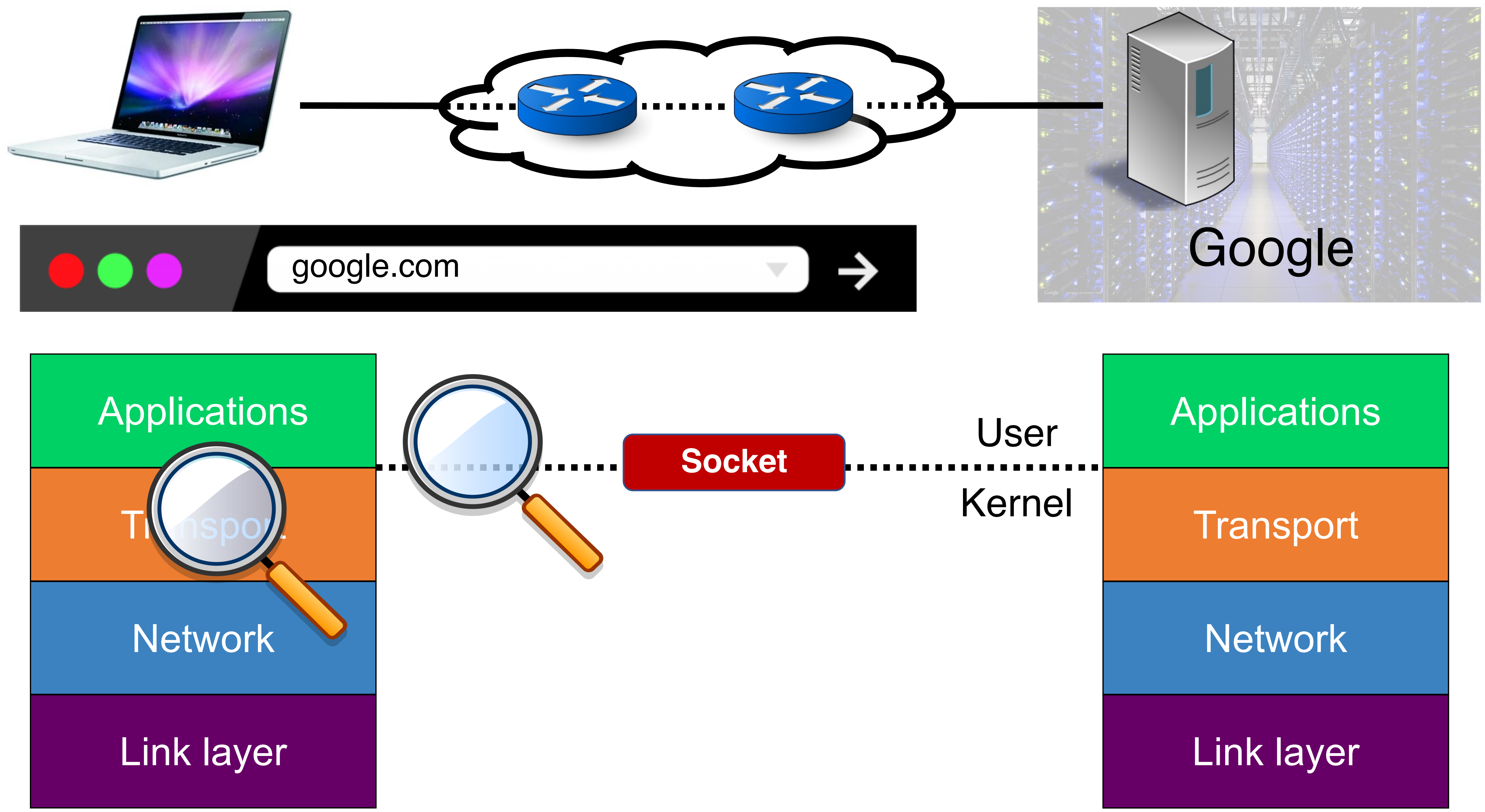

google.com
Google
Applications
Transport
Network
Link layer
User
Socket
Kernel
Applications
Transport
Network
Link layer
Example: connected
socket (TCP)



google.com
Google
process
socket
IP_A + port_A
connect(
IP_B, port_B)
send()
process
socket
IP_B + port_B
bind(IPaddr_B, port_B)
listen()
accept()
recv()



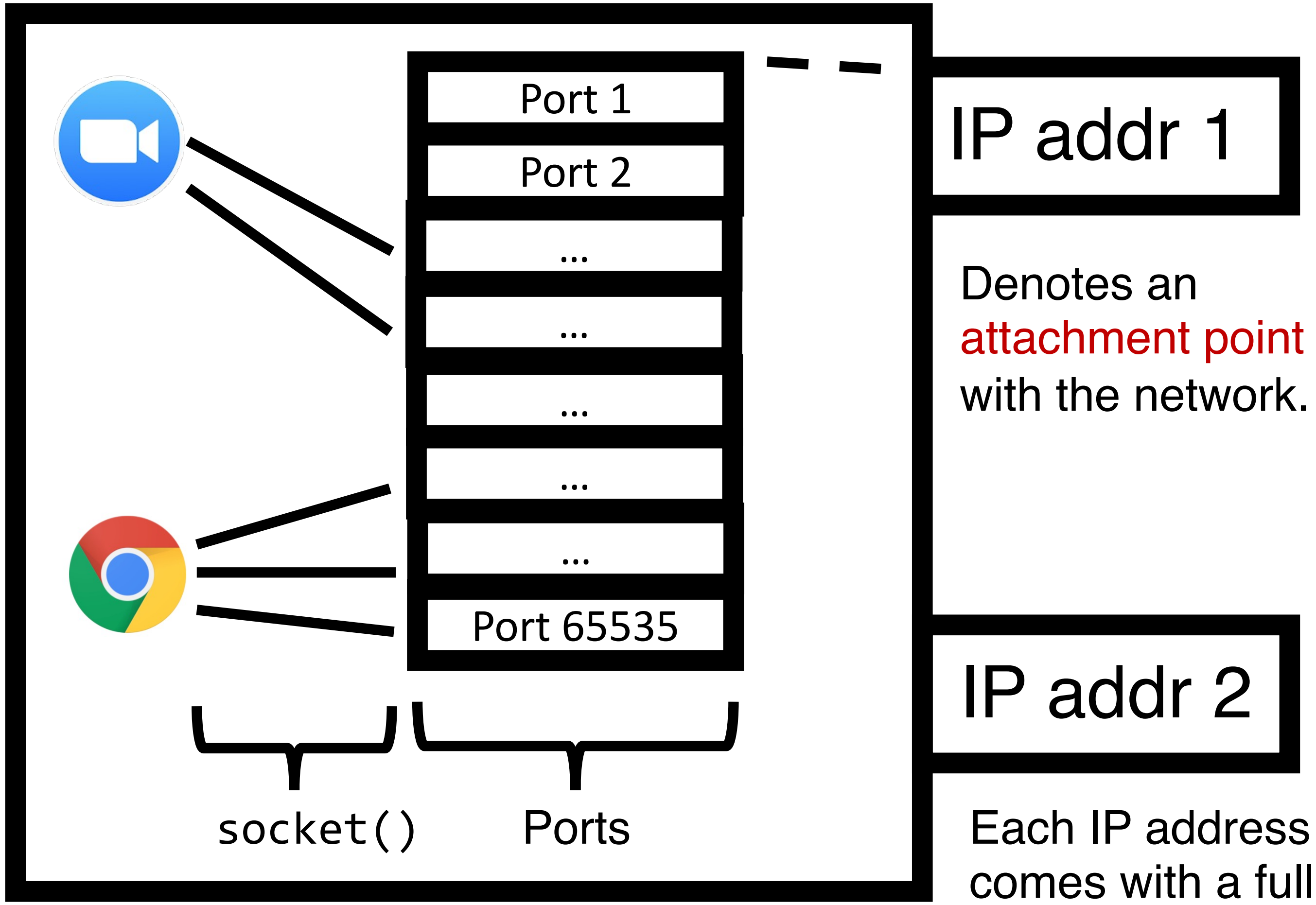
google.com
Google
root DNS server
.edu DNS server
local DNS server
dns.rutgers.edu
requesting host
cs.rutgers.edu
umass.edu DNS server
dns.umass.edu
gaia.cs.umass.edu
1
2
3
4
5
6
7
8
I want to browse google.com
Host name
Server IP Address
Hostname
IP address
Google.com
10.0.1.2
DNS server
clientIP, clientPort, server IP Address, 80
HTTP request
HTTP response
HTTP messages
(HTTP application typically associated with port 80)
send()
recv()



google.com
Google
Applications
Transport
Network
Link layer
Socket
User
Kernel
Applications
Transport
Network
Link layer

Transport

(1) (De)multiplexing

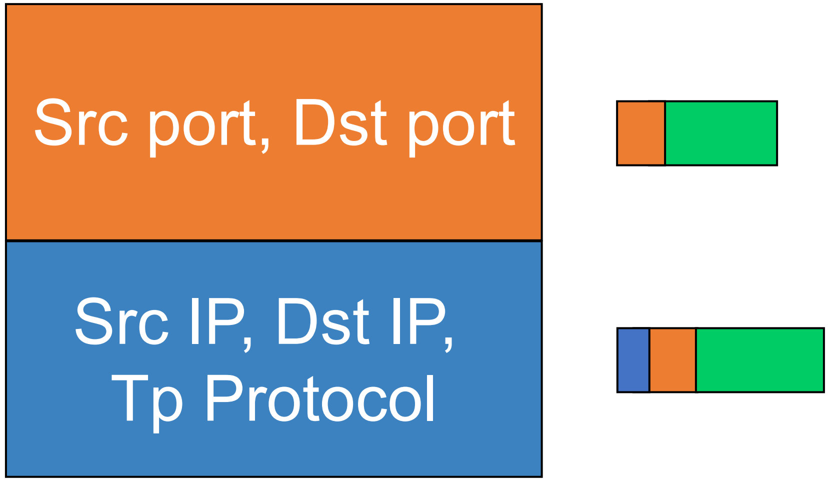


Machine

Denotes an attachment point with the network.

Each IP address comes with a full copy of its own ports.

Connection lookup: The operating system does a lookup using these data to determine the right socket and app.



UDP or TCP listening:
(dst IP, dst port, TCP)

TCP established:
(dst IP, dst port, src IP, src port, TCP)

TCP sockets of different types

Listening (bound but unconnected)

```
# On server side
ls = socket(AF_INET, SOCK_STREAM)
ls.bind(serv_ip, serv_port)
ls.listen() # no accept() yet
```

(dst IP, dst port)

➔

Socket (ss)

Enables new connections to be demultiplexed correctly

Connected (Established)

```
# On server side
cs, addr = ls.accept()

# On client side
connect(serv_ip, serv_port)
```

`accept()` creates a new socket with the 4-tuple (established) mapping

(src IP, dst IP, src port, dst port)

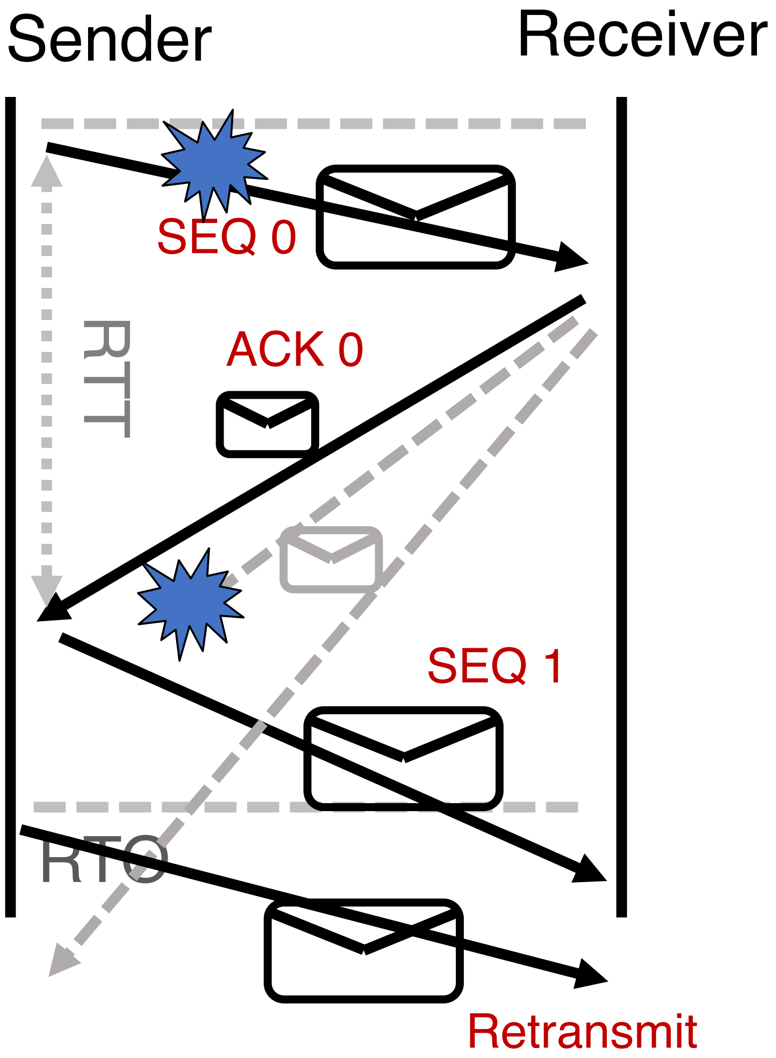
➔

Socket (cs NOT ls)

Enables established connections to be demultiplexed correctly

(2) Reliability: Stop and Wait. 3 Ideas

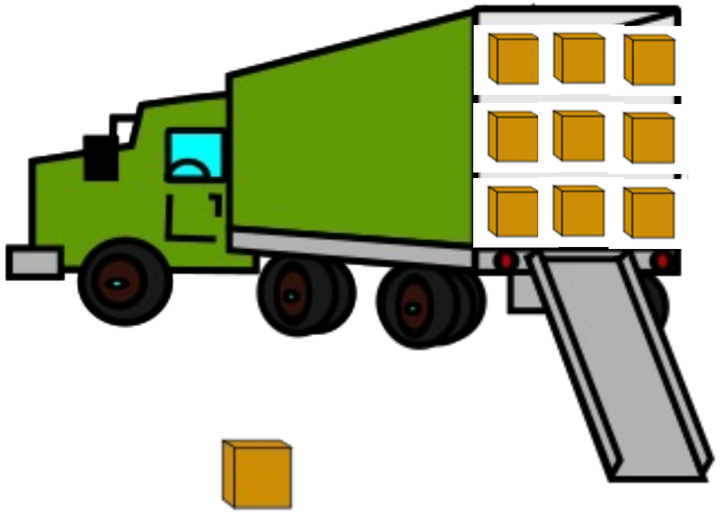
- ACKs: Sender sends a single packet, then waits for an ACK to know the packet was successfully received. Then the sender transmits the next packet.
- RTO: If ACK is not received until a timeout, sender retransmits the packet
- Seq: Disambiguate duplicate vs. fresh packets using sequence numbers that change on “adjacent” packets



Sending one packet per RTT makes the data transfer rate limited by the time between the endpoints, rather than the bandwidth.

Ensure you got the (one) box safely; make N trips

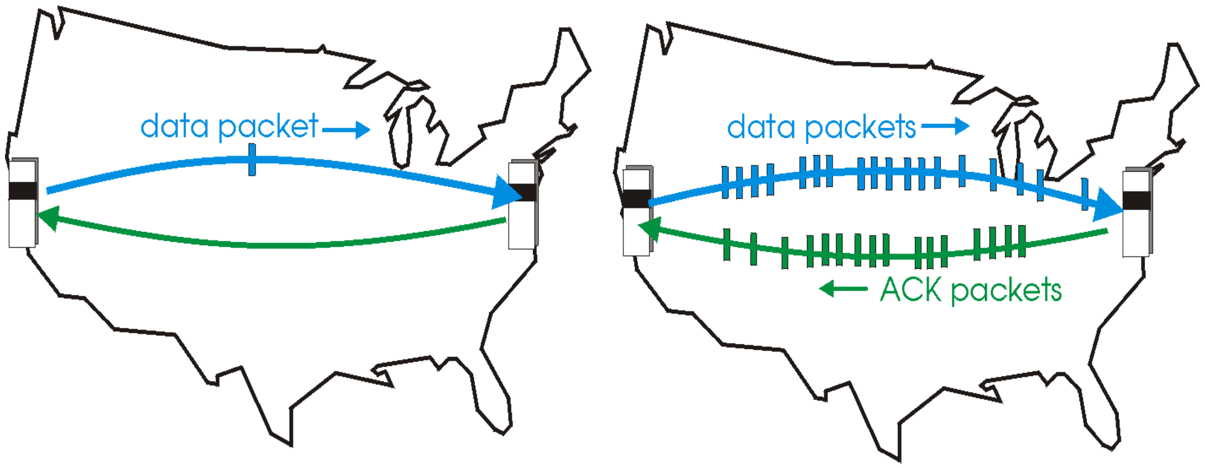
Ensure you get N boxes safely; make just 1 trip!



Keep many packets in flight

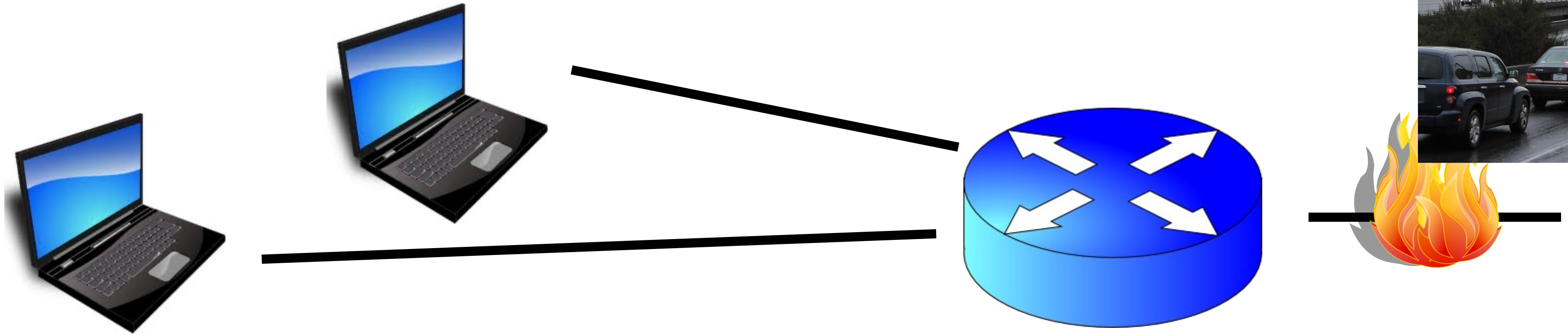
Pipelined reliability

- Data in flight: data that has been sent, but sender hasn't yet received ACKs from the receiver
  - Note: can refer to packets in flight or bytes in flight
- New packets sent at the same time as older ones still in flight
- New packets sent at the same time as ACKs are returning
- More data moving in same time!
- Improves throughput
  - Rate of data transfer

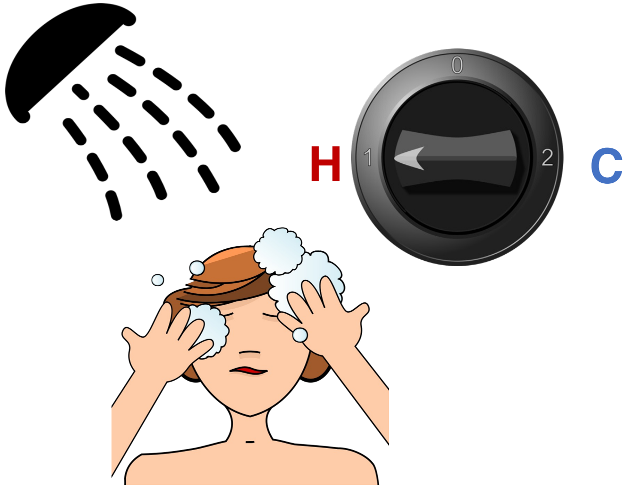


(a) a stop-and-wait protocol in operation (b) a pipelined protocol in operation

(3) How much data to keep in flight?



- Avoid overwhelming network resources: Congestion control
- Internet: every endpoint makes its own decisions!
  - Distributed algorithm: no central authority
  - Goal 1: efficiency (use available capacity)
  - Goal 2: fairness (distribute capacity equitably)



Feedback Control

Finding the right congestion window

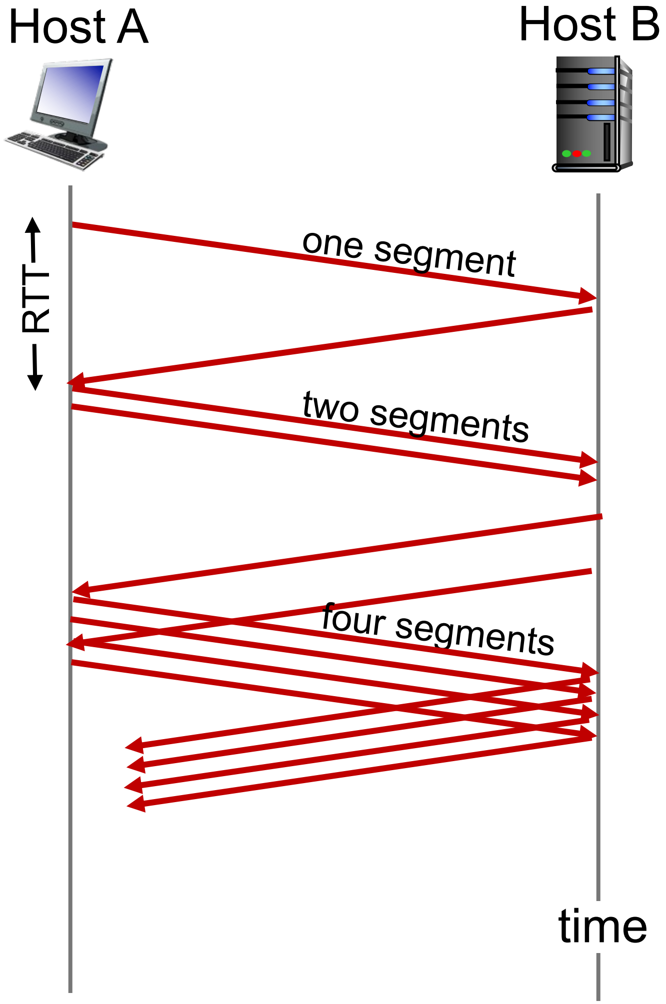
- There is an unknown bottleneck link rate that the sender must match
- If sender sends more than the bottleneck link rate:
  - packet loss, delays, etc.
- If sender sends less than the bottleneck link rate:
  - all packets get through; successful ACKs
- Congestion window (`cwnd`): amount of data in flight

Quickly finding a rate: TCP slow start

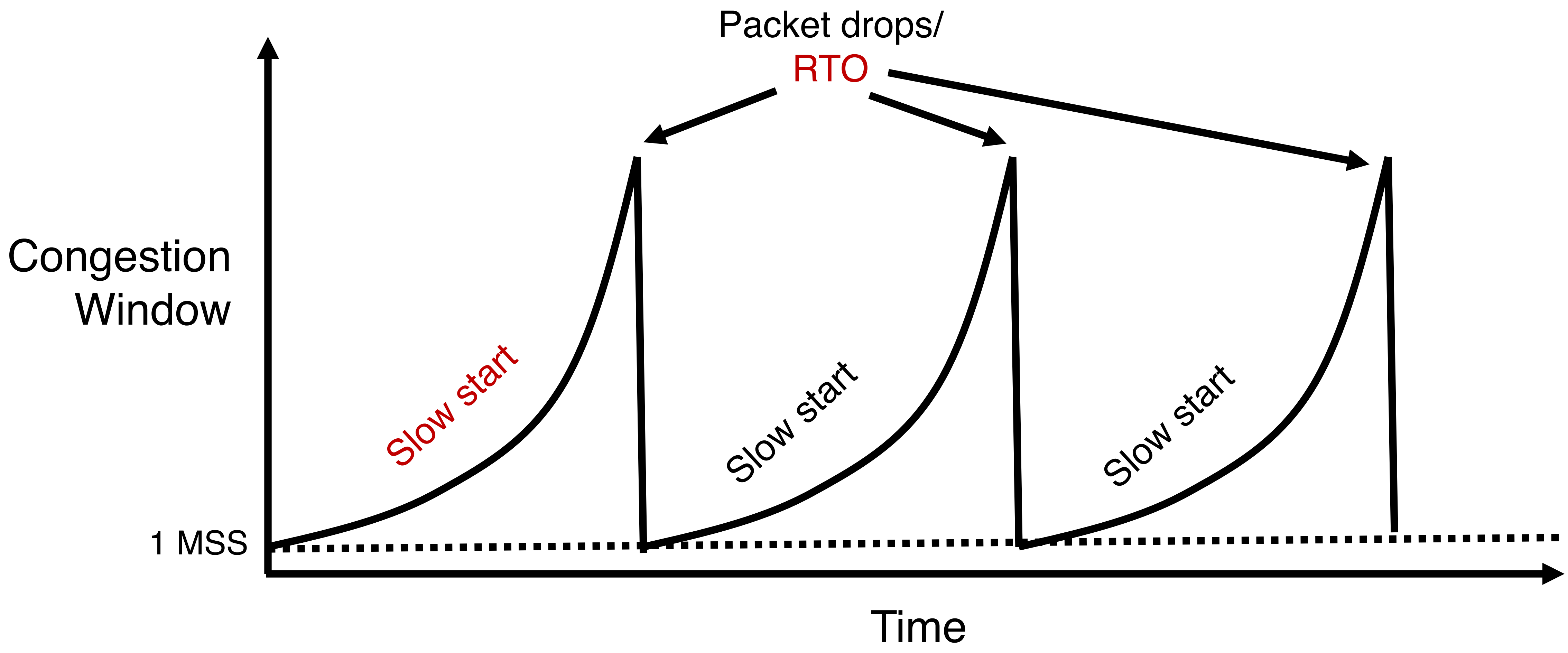
- Initially `cwnd` = 1 MSS
  - MSS is “maximum segment size”
- Upon receiving an ACK of each MSS, increase the `cwnd` by 1 MSS
- Effectively, double `cwnd` every RTT

▪ Initial rate is slow but ramps up exponentially fast

▪ On loss (RTO), restart from `cwnd := 1 MSS`



Behavior of slow start

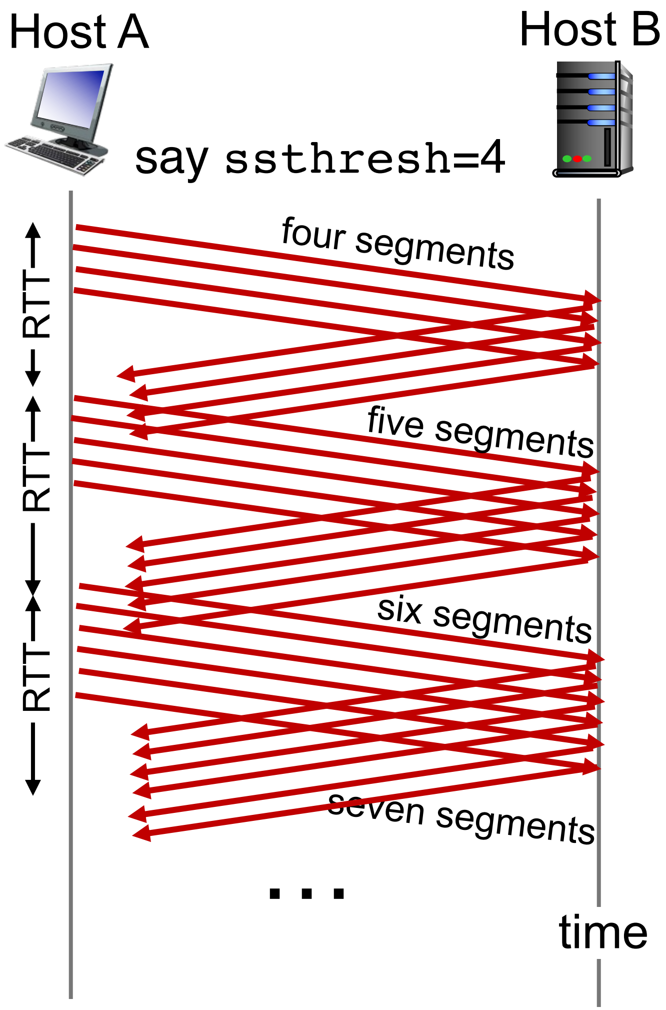


Slow start has problems

- Congestion window increases too rapidly
  - Example: suppose the “right” window size `cwnd` is 17
  - `cwnd` would go from 16 to 32 and then dropping down to 1
  - Result: massive packet drops
- Congestion window decreases too rapidly
  - Suppose the right `cwnd` is 31, and there is a loss when `cwnd` is 32
  - Slow start will resume all the way back from `cwnd` 1
  - Result: unnecessarily low speed of sending data
- Instead, perform finer adjustments of `cwnd`: congestion avoidance

TCP New Reno: Additive Increase

- Remember the recent past to find a good estimate of link rate
- The last good `cwnd` without packet drop is a good indicator
  - TCP New Reno calls this the slow start threshold (`ssthresh`)
- Increase `cwnd` by 1 MSS every RTT after `cwnd` hits `ssthresh`
  - Effect: increase window additively per RTT



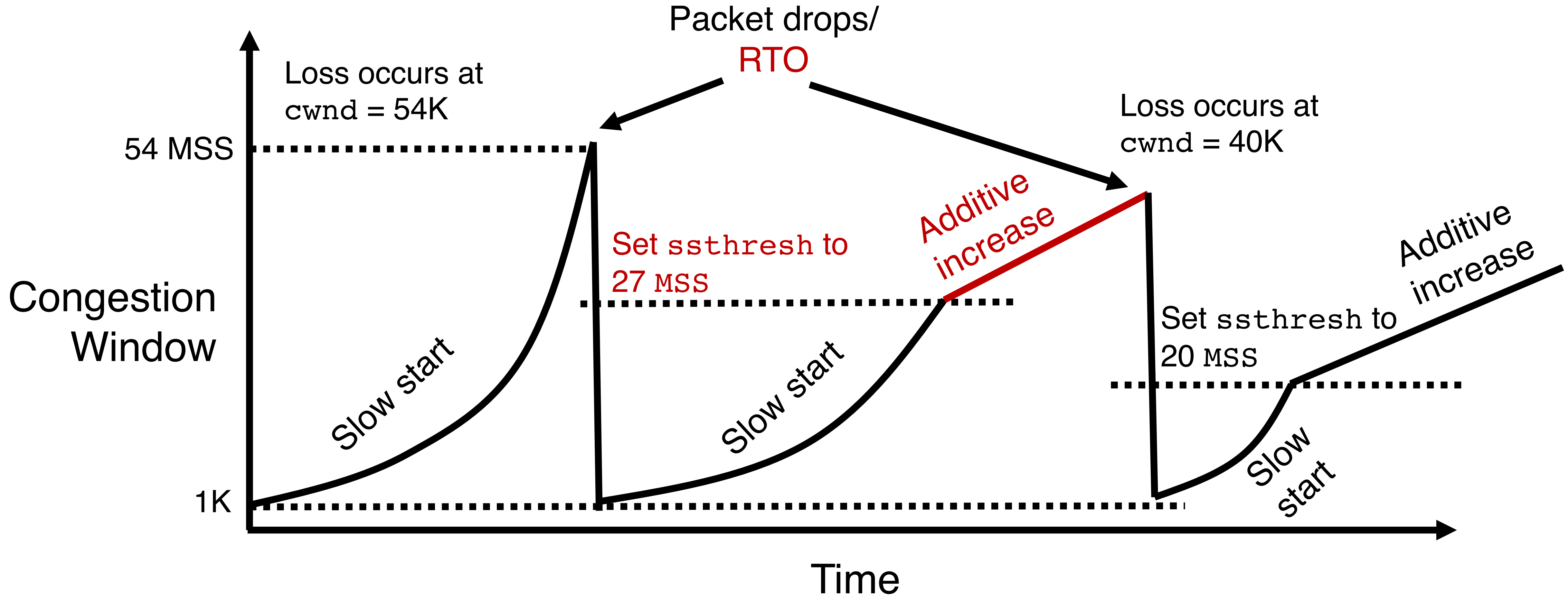
TCP New Reno: Additive increase

- Start with `ssthresh = 64K bytes` (TCP default)
- Do slow start until `ssthresh`
- Once the threshold is passed, do additive increase
  - Add one MSS to `cwnd` for each `cwnd` worth data ACK'ed
  - For each MSS ACK'ed, `cwnd = cwnd + (MSS * MSS) / cwnd`
- Upon a TCP timeout (RTO),
  - Set `cwnd = 1 MSS`
  - Set `ssthresh = max(2 * MSS, 0.5 * cwnd)`
  - i.e., the next linear increase will start at half the current `cwnd`

Behavior of Additive Increase

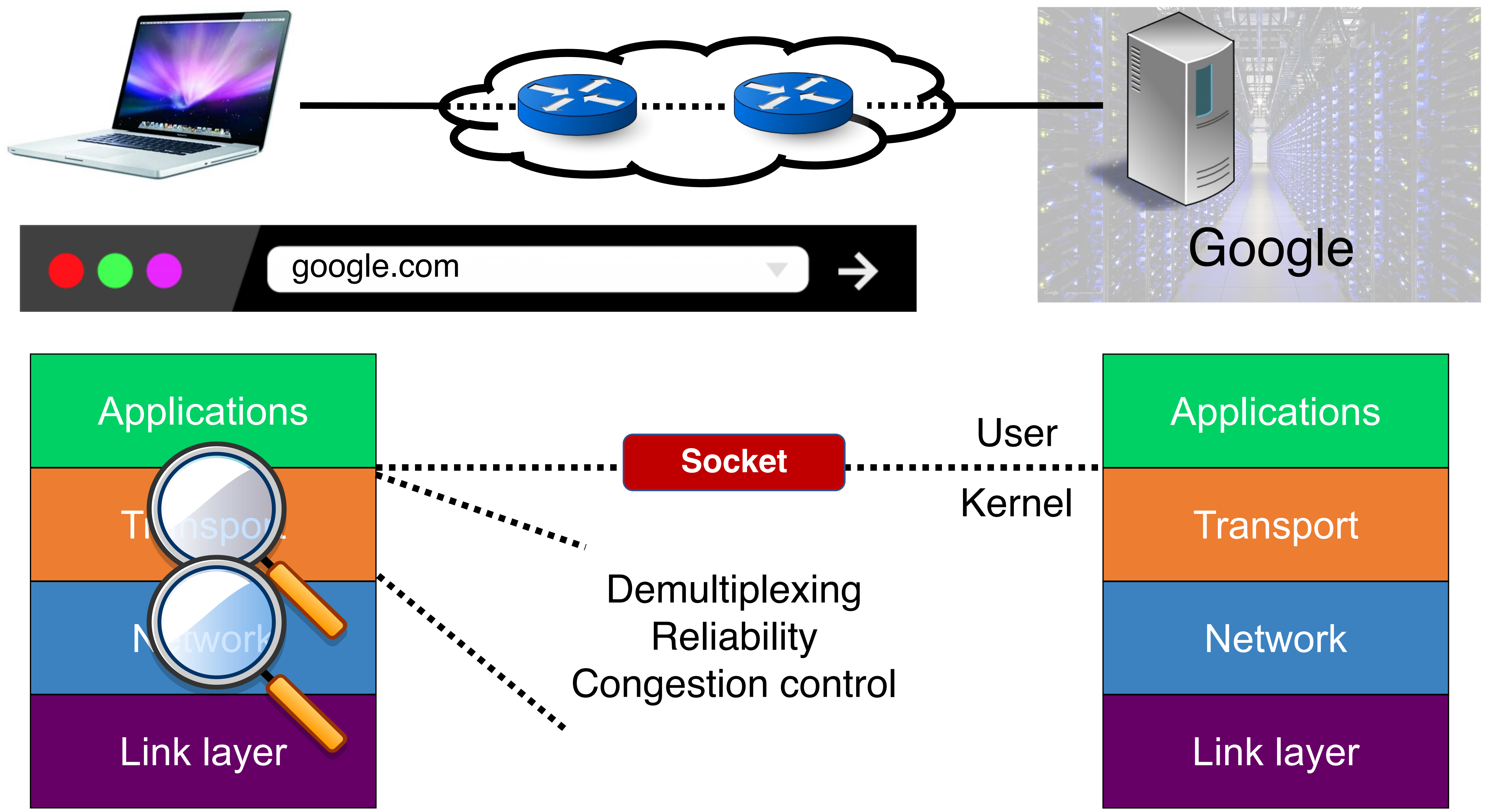
Say `MSS` = 1 KByte

Default `ssthresh` = 64KB = 64 `MSS`



Routing


Norwegian Sea
Iceland
Sweden
Finland
Norway
Russia
Denmark
Ireland
Poland
Belarus
Ukraine
Austria
Romania
Kazakhstan
Spain
Portugal
Turkey
Turkmenistan
Sea of Japan
Japan
China
Sea of Okhotsk
Tunisia
Syria
Iraq
Iran
Afghanistan
Pakistan
Morocco
Algeria
Libya
Egypt
Saudi Arabia
Nepal
India
Western Sahara
Mauritania
Mali
Niger
Chad
Sudan
Oman
Yemen
Myanmar (Burma)
Burkina Faso
Guinea
Nigeria
Ghana
South Sudan
Ethiopia
Gulf of Aden
Arabian Sea
Bay of Bengal
Philippines
Gulf of Guinea
Somalia
Laccadive Sea
Gabon
Kenya
DR Congo
Tanzania
Indonesia
Banda Sea
Arafura Sea
Papua New Guinea



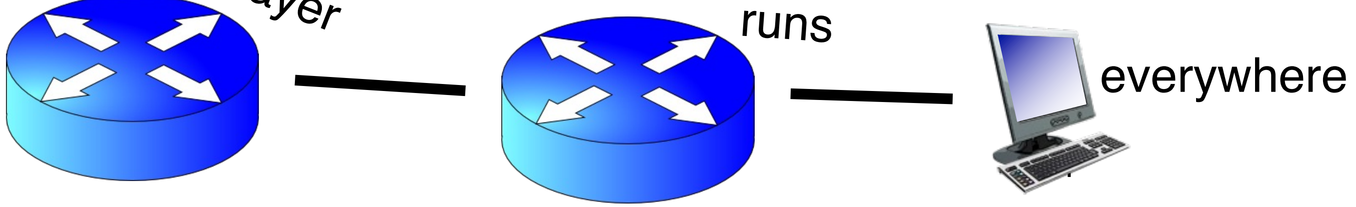
google.com
Google
Applications
Transport
Network
Link layer
Socket
User
Kernel
Demultiplexing
Reliability
Congestion control
Applications
Transport
Network
Link layer

Two key network-layer functions

- Forwarding: move packets from router's input to appropriate router output

- Routing: determine route taken by packets from source to destination
  - routing algorithms

- The network layer solves the routing problem.



Analogy: taking a road trip

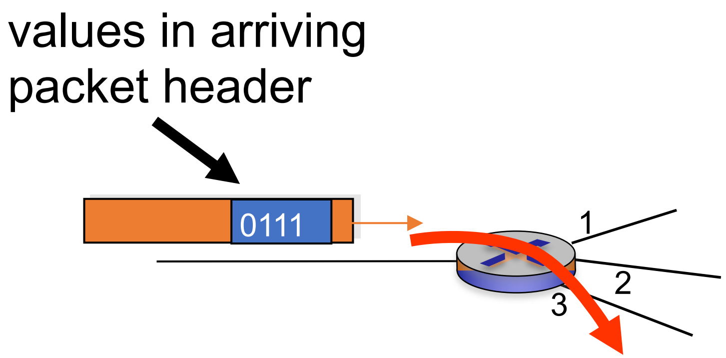
- Forwarding: process of getting through single exit

- Routing: process of planning trip from source to destination

Control/Data Planes

Data plane = Forwarding

- local, per-router function
- determines how datagram arriving on router input port is forwarded to router output port



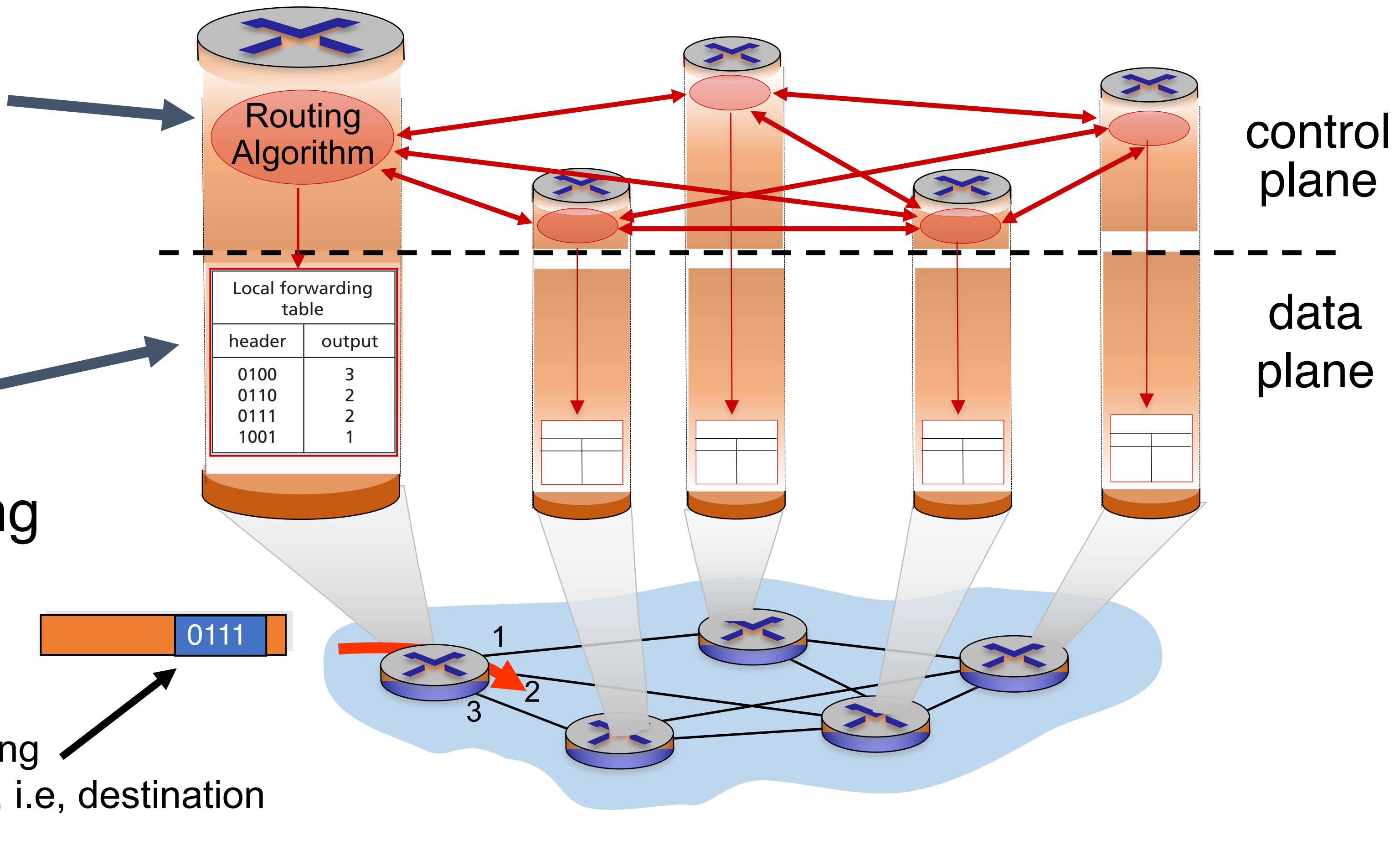
Control plane = Routing

- network-wide logic
- determines how datagram is routed along end-to-end path from source to destination endpoint
- two control-plane approaches:
  - Distributed routing algorithm running on each router
  - Centralized routing algorithm running on a (logically) centralized machine

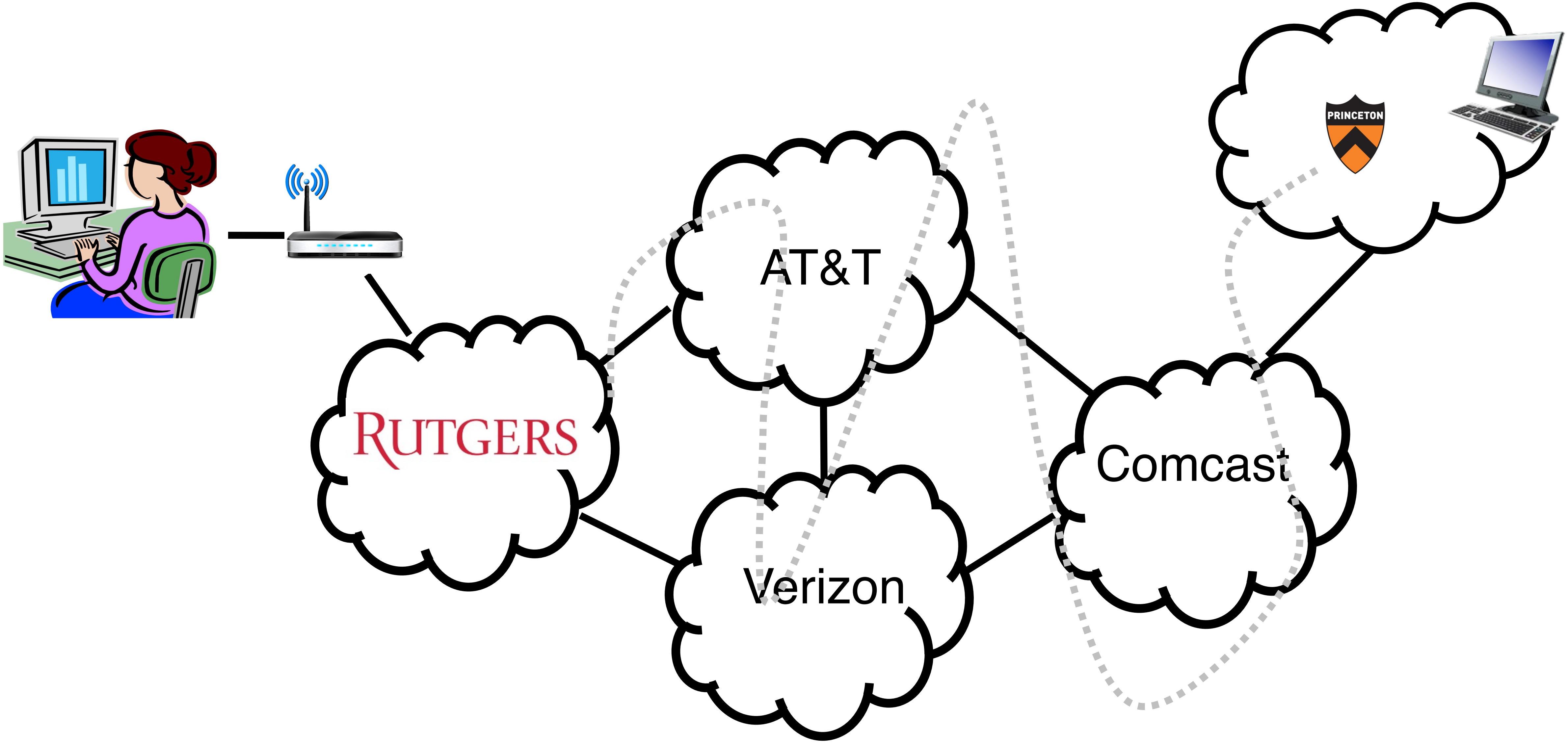
Distributed routing

Control plane
Traditional routing protocols: per route-change processing (~ a few tens of seconds)

Data plane
per-packet processing (~ tens of nanoseconds)



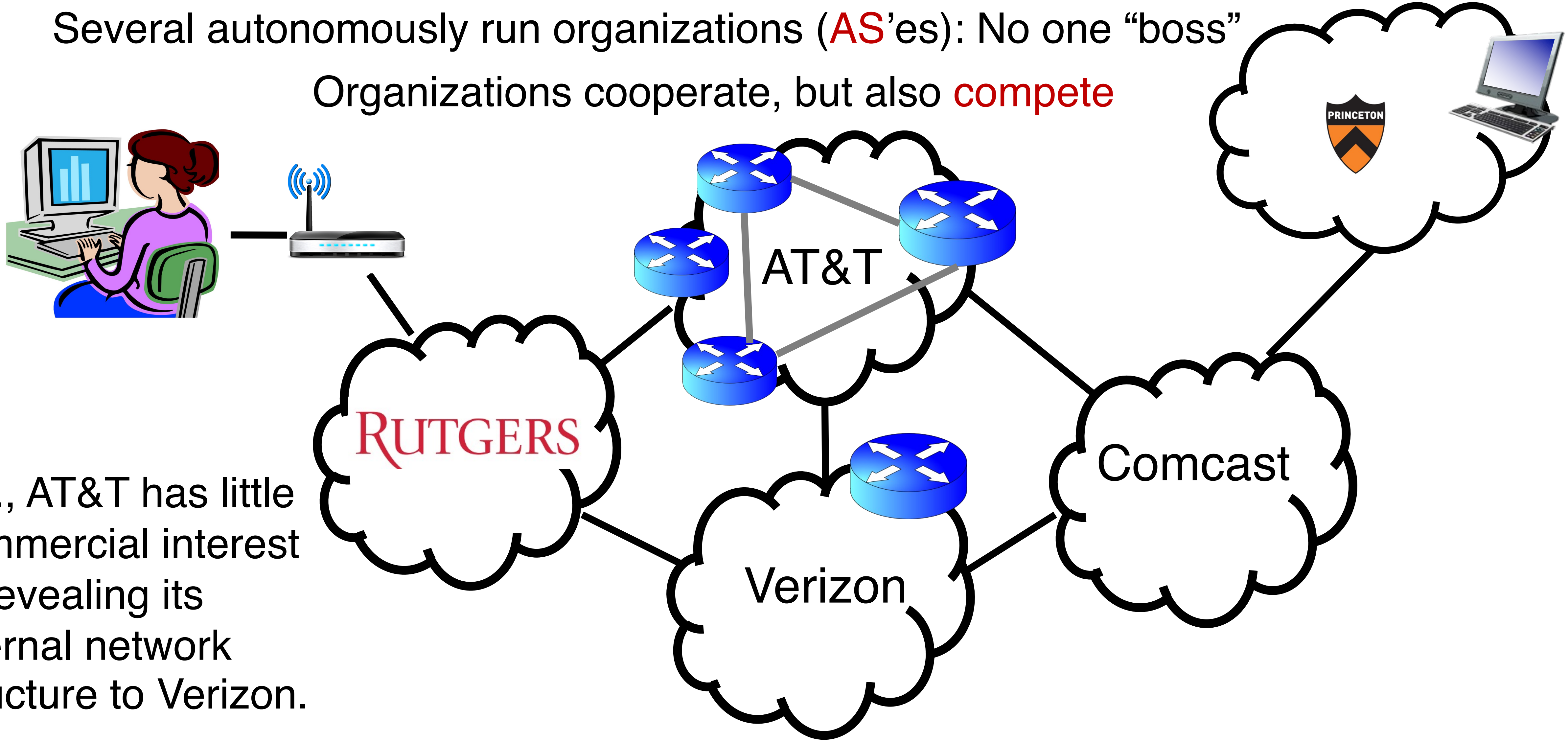
The Internet is a large federated network



The Internet is a large federated network

Several autonomously run organizations (AS'es): No one "boss"

Organizations cooperate, but also compete

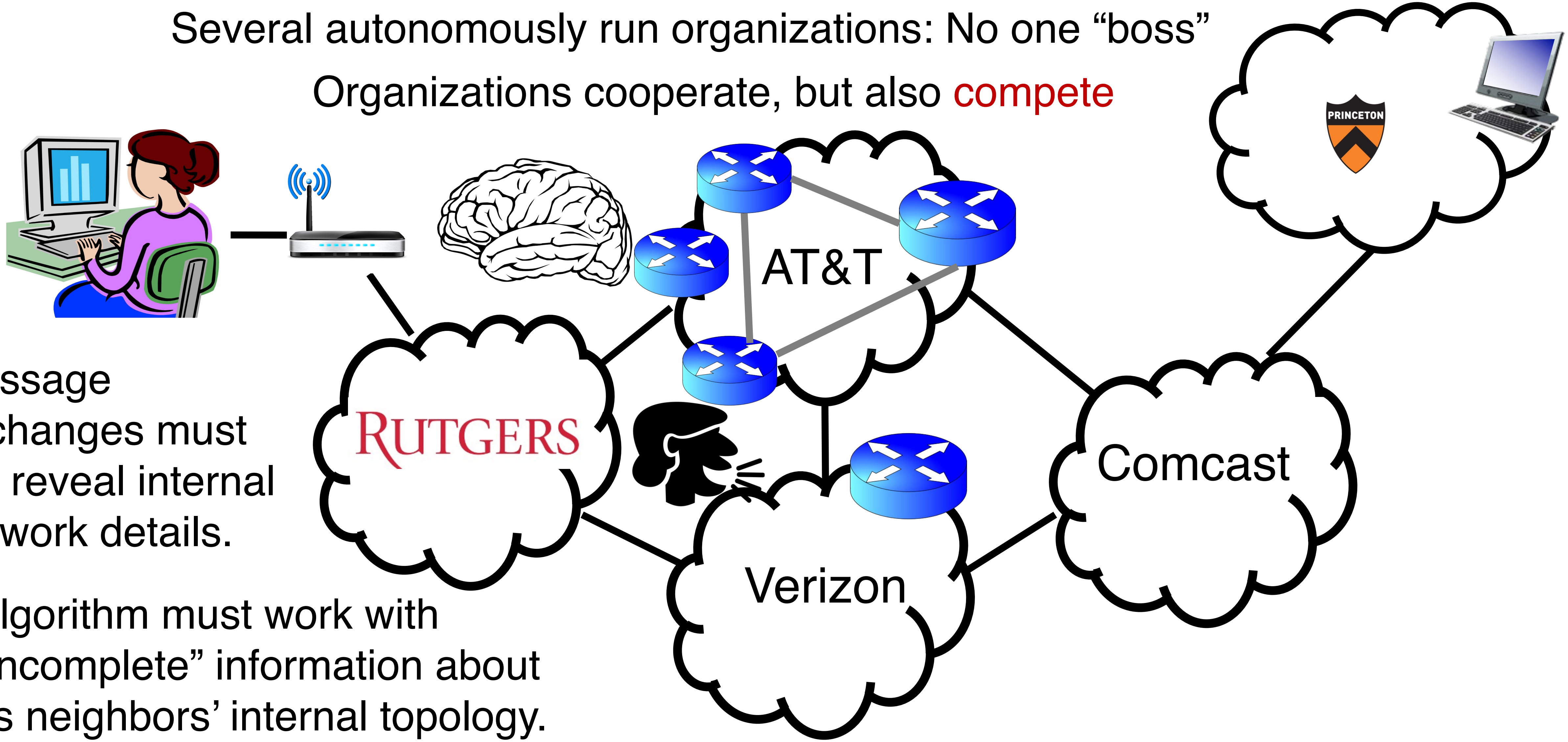


e.g., AT&T has little commercial interest in revealing its internal network structure to Verizon.

The Internet is a large federated network

Several autonomously run organizations: No one "boss"

Organizations cooperate, but also compete



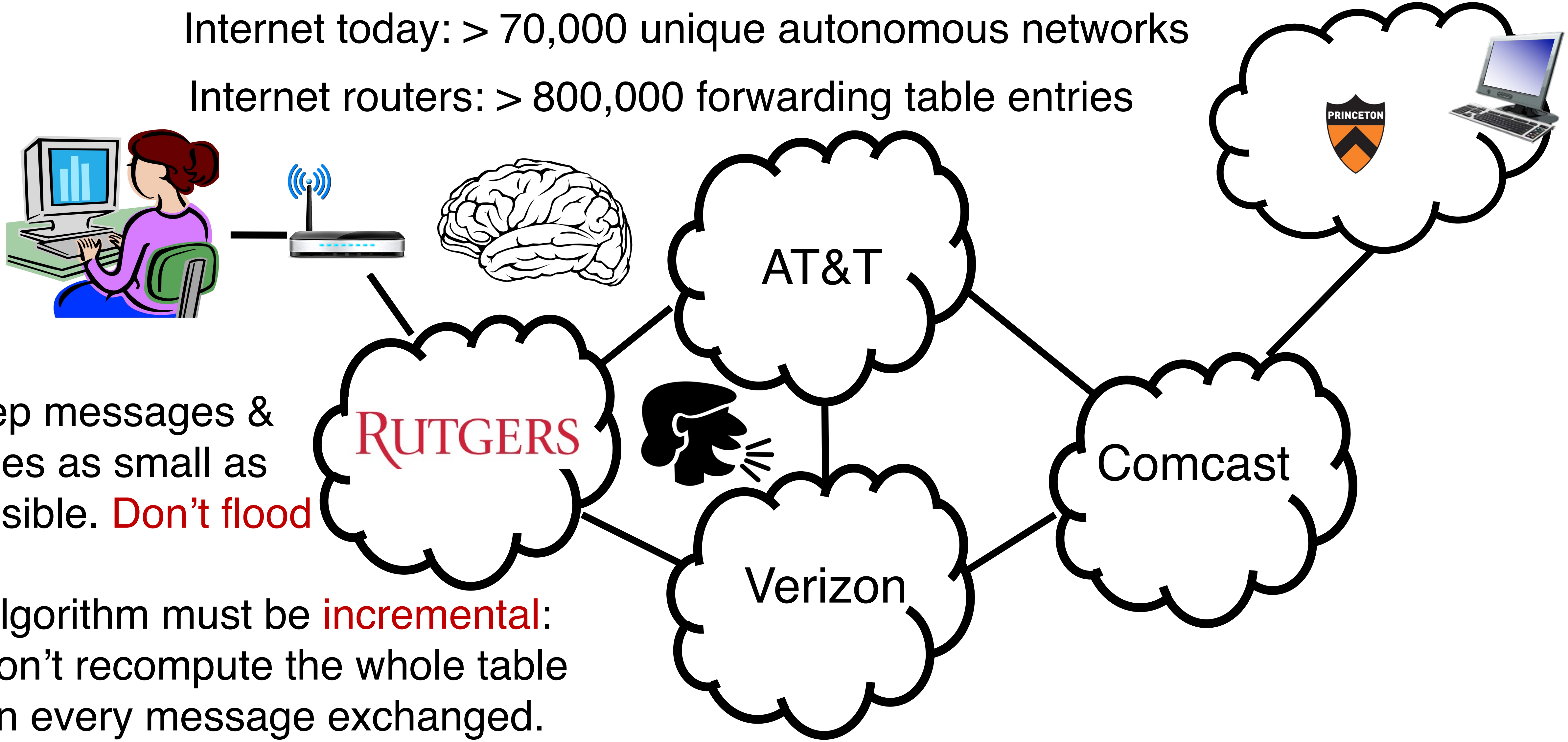
Message exchanges must not reveal internal network details.

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

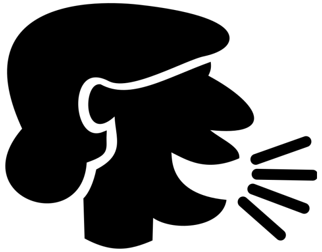


Keep messages & tables as small as possible. Don’t flood

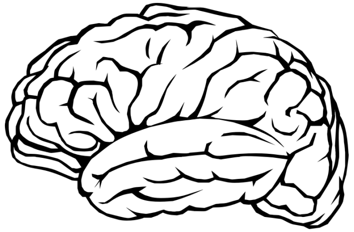
Algorithm must be incremental: don’t recompute the whole table on every message exchanged.

Inter-domain Routing

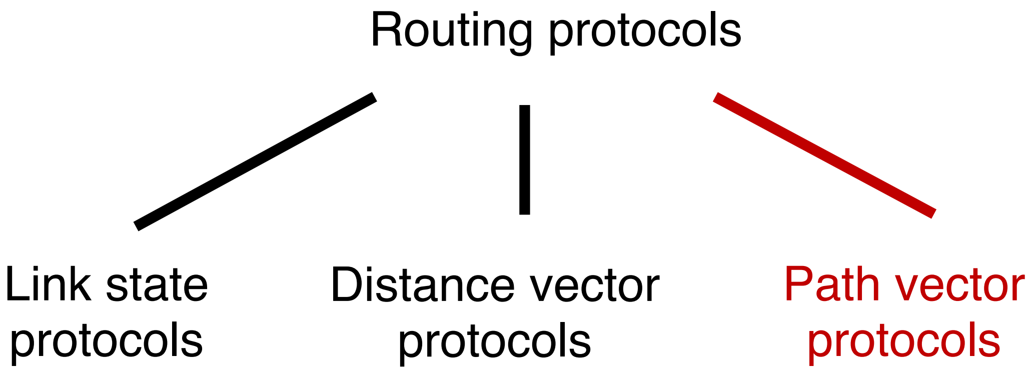
- The Internet uses Border Gateway Protocol (BGP)
- All AS'es speak BGP. It is the glue that holds the Internet together
- BGP is a path vector protocol



Messages?

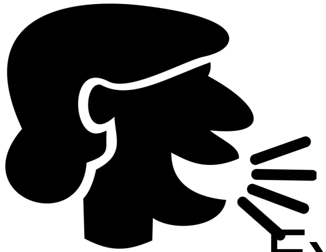


Algorithm?



Applicable within a single AS

(1) BGP Messages

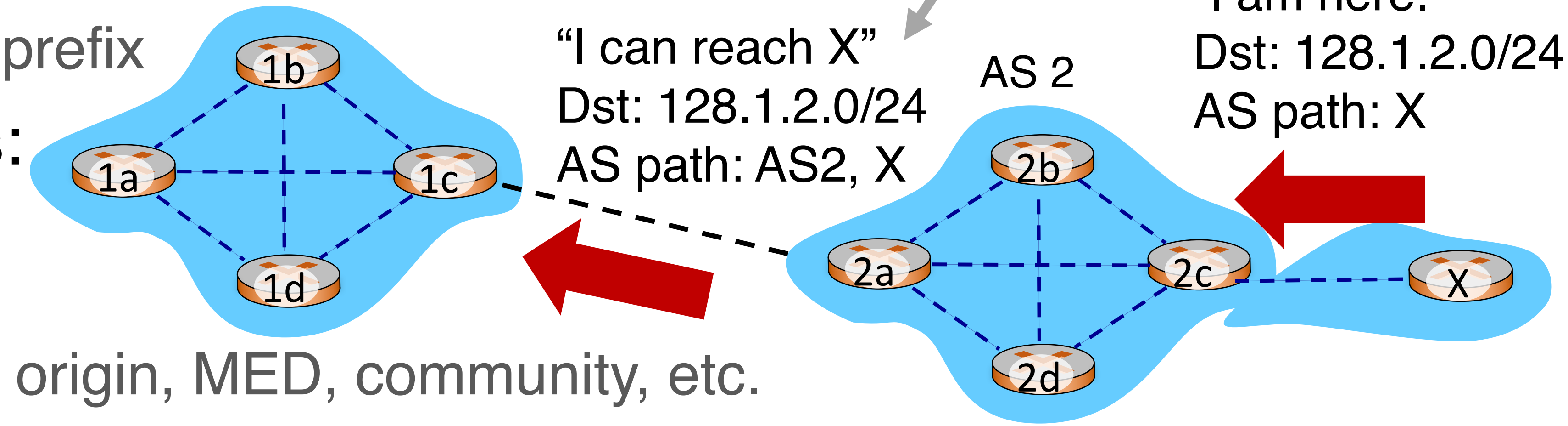


Loop detection is easy
(no "count to infinity")

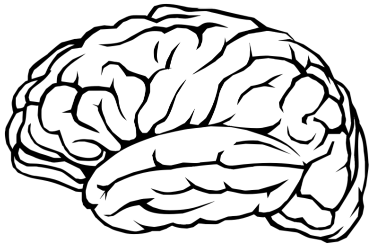
Exchange paths: path vector

No link metrics, distances!

- Routing Announcements or Advertisements
  - "I am here" or "I can reach here"
  - Occur over a TCP connection (BGP session) between routers
- Route announcement = destination + attributes
  - Destination: IP prefix
- Route Attributes:
  - AS-level path
  - Next hop
  - Several others: origin, MED, community, etc.
- An AS promises to use advertised path to reach destination
- Only route changes are advertised after BGP session established



(2) BGP algorithm



- A BGP router does *not* consider every routing advertisement it receives by default to make routing decisions!
  - An import policy determines whether a route is even considered a candidate
- Once imported, the router performs route selection
- A BGP router does *not* propagate its chosen path to a destination to all other AS'es by default!
  - An export policy determines whether a (chosen) path can be advertised to other AS'es and routers

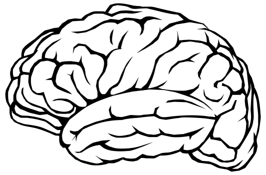
Programmed by network operator

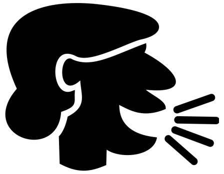
Business policy considerations drive BGP.
NOT efficiency considerations.

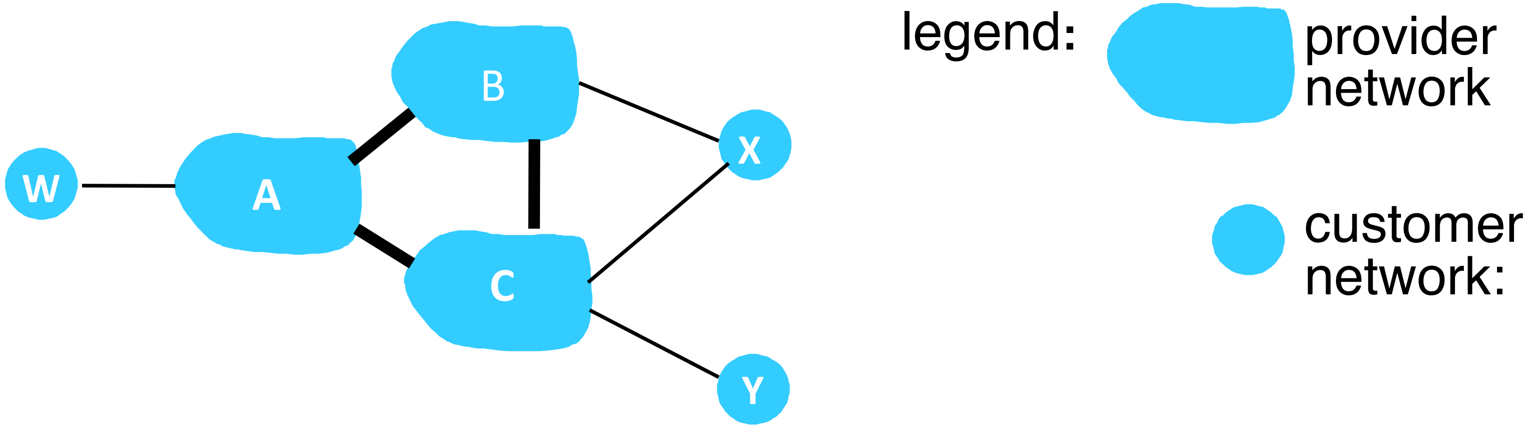
Policy arises from business relationships

- Customer-provider relationships:
  - E.g., Rutgers is a customer of AT&T
- Peer-peer relationships:
  - E.g., Verizon is a peer of AT&T
- Business relationships depend on where connectivity occurs
  - "Where", also called a "point of presence" (PoP)
  - e.g., customers at one PoP but peers at another
  - Internet-eXchange Points (IXPs) are large PoPs where ISPs come together to connect with each other (often for free)

BGP Export Policy



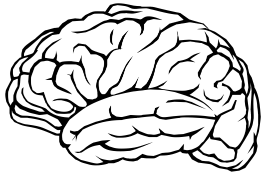


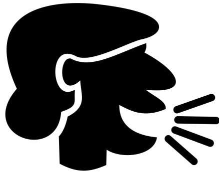


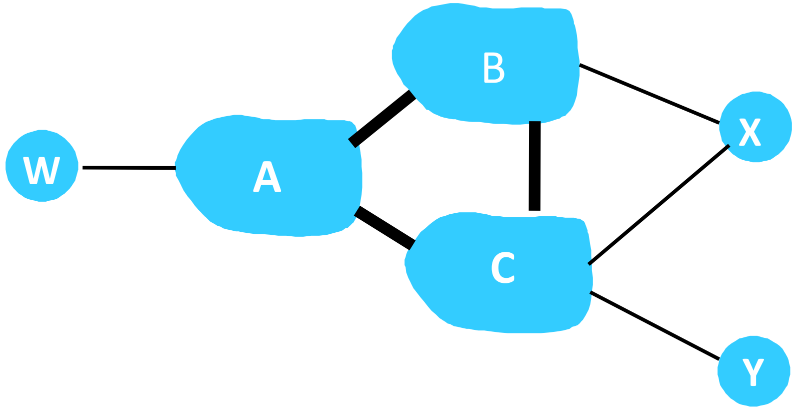
Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

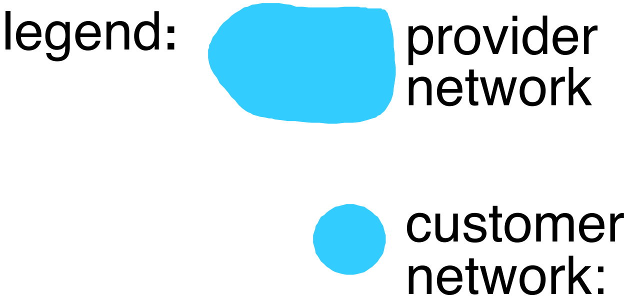
- A,B,C are provider networks
- X,W,Y are customers (of provider networks)
- X is dual-homed: attached to two networks
- policy to enforce: X does not want to route from B to C via X
  - So, X will not announce to B a route to C

BGP Export Policy





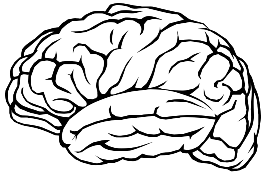


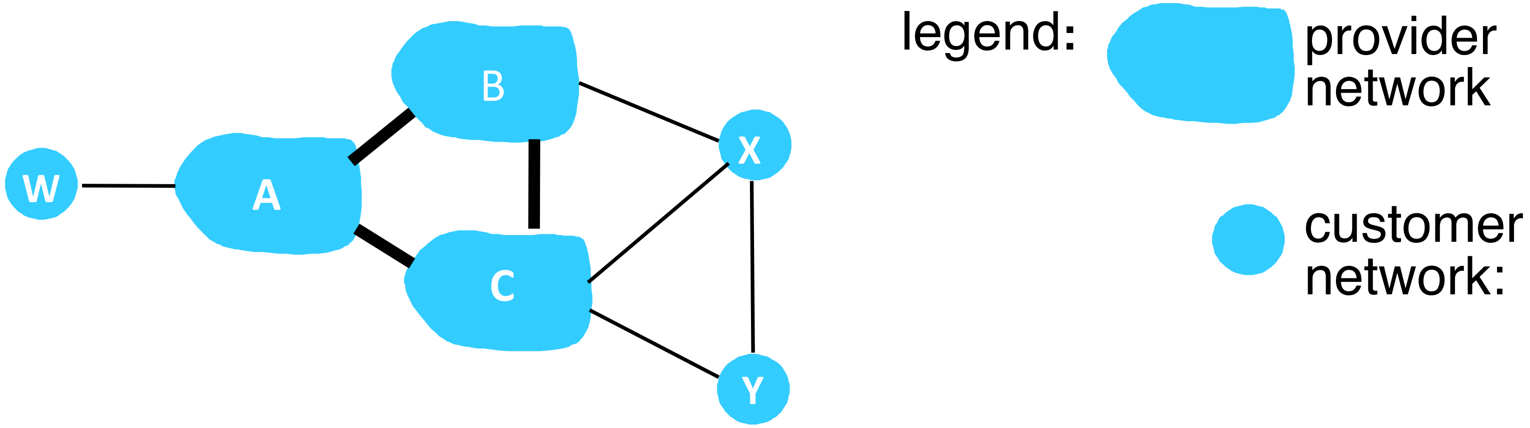


Suppose an ISP only wants to route traffic to/from its customer networks (does not want to carry transit traffic between other ISPs)

- A announces path Aw to B and to C
- B will not announce BAw to C:
  - B gets no "revenue" for routing CBAw, since none of C, A, w are B's customers
- C will route CAw (not using B) to get to w

BGP Import Policy

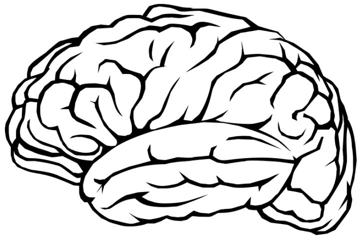




Suppose an ISP wants to minimize costs by avoiding routing through its providers when possible.

- Suppose C announces path Cy to x
- Further, y announces a direct path (“y”) to x
- Then x may choose not to import the path Cy to y since it has a peer path (“y”) towards y

Q2. BGP Route Selection



- When a router imports more than one route to a destination IP prefix, it selects route based on:
  1. local preference value attribute (import policy decision -- set by network admin)
  2. shortest AS-PATH
  3. closest NEXT-HOP router
  4. Several additional criteria: You can read up on the full, complex, list of criteria, e.g., at https://www.cisco.com/c/en/us/support/docs/ip/border-gateway-protocol-bgp/13753-25.html

Problems with BGP

- Not designed for efficiency

  1. local preference value attribute (import policy decision -- set by network admin)
  2. shortest AS-PATH
  3. closest NEXT-HOP router

  Nothing to do with path length, delay, or available capacity.

- Only a single path per destination
- Slow to converge after a change
- Vulnerable to bugs & malice