Design principles/protocol functions

Goals:

- Identify, study common architectural components, protocol mechanisms, approaches do we find in network architectures?
- *Synthesis:* big picture

Principles / protocol functions:

- Signaling
  - Protocols
  - Separation of data, control
  - Hard state versus soft state
- Randomization
- Indirection
- Network virtualization
- Multiplexing
- Design for scale

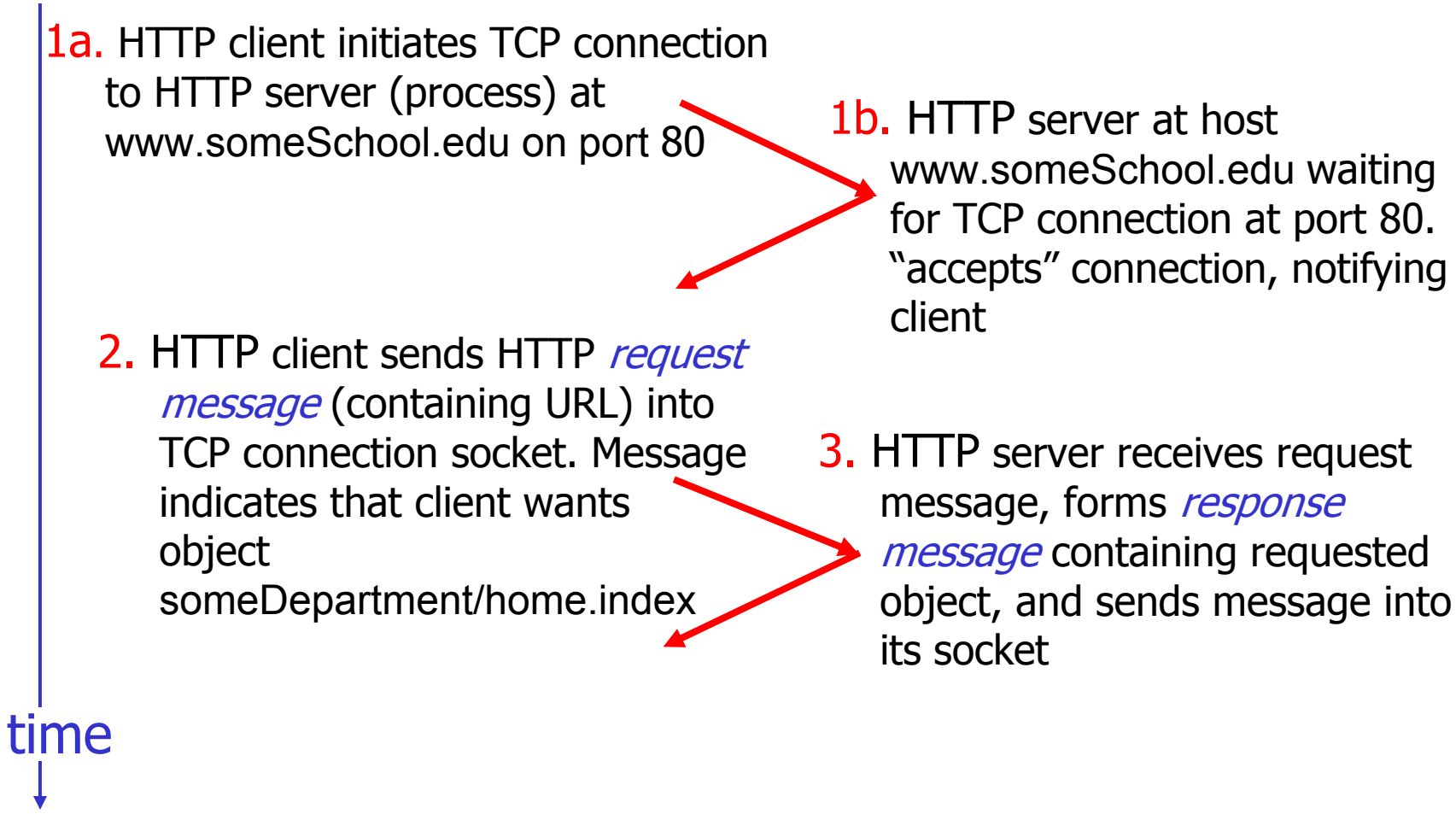
1: Separation of control and data

- Internet:
  - http: in-band signaling; ftp: out-of-band signaling
  - RSVP (signaling) separate from routing, forwarding.
- PSTN (public switched telephone network):
  - SS7 (packets-switched control network) separate from (circuit-switched) call trunk lines
  - Earlier tone-based (in-band signaling)

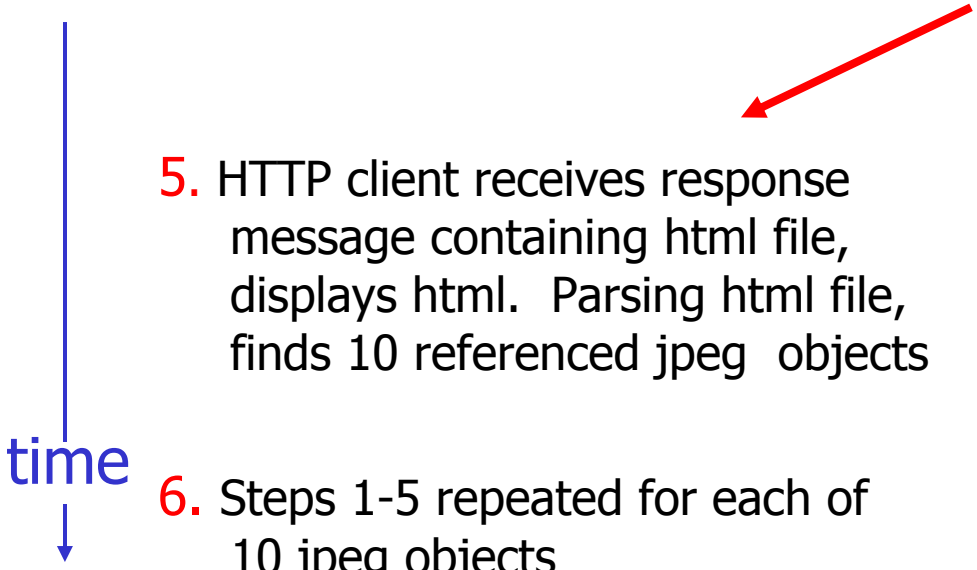
Internet: HTTP – inband signaling

Suppose user enters URL

`www.someSchool.edu/someDepartment/home.index`



Nonpersistent HTTP (cont.)



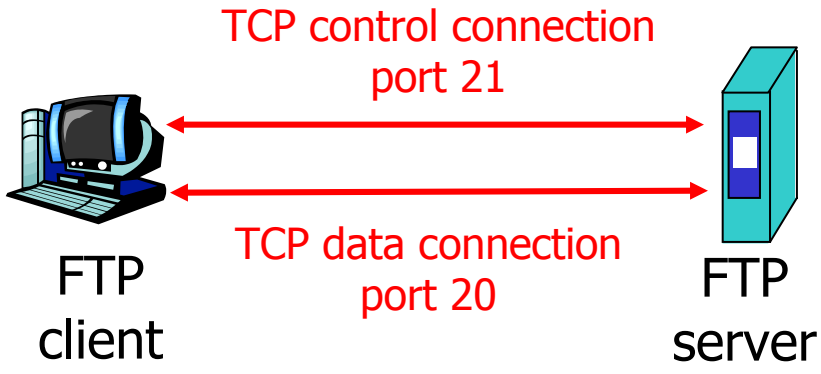
4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

FTP: Separate control, data connections

- FTP client contacts FTP server at port 21
- Client obtains authorization over control connection
- Client browses remote directory via commands sent over control connection.
- When server receives file transfer commend server opens new TCP data connection to client
- After transferring one file, server closes connection.



- Server opens 2nd TCP data connection to transfer another file.
- Control connection: "out of band" signaling
- FTP server maintains "state": current directory, earlier authentication

Separate control, data: Why (or why not)?

Why?

- Allows concurrent control + data
- Allows perform authentication at control level
- Simplifies processing of data/control streams- higher throughput
- Provide QoS appropriate for control/data streams

Why not?

- Separate channels complicate management, increases resource requirements
- Can increase latency, e.g., http – two tcp connections vs. one.

2: Maintaining network state

State: Information *stored* in network nodes by network protocols

- Updated when network "conditions" change
- Stored in multiple nodes
- Often associated with end-system generated call or session
- Examples:
  - TCP sequence numbers, timer values, RTT estimates

State: Senders, receivers

- Sender: Network node that (re)generates signaling (control) msgs to install, keep-alive, remove state from other nodes
- Receiver: Node that creates, maintains, removes state based on signaling msgs received from sender

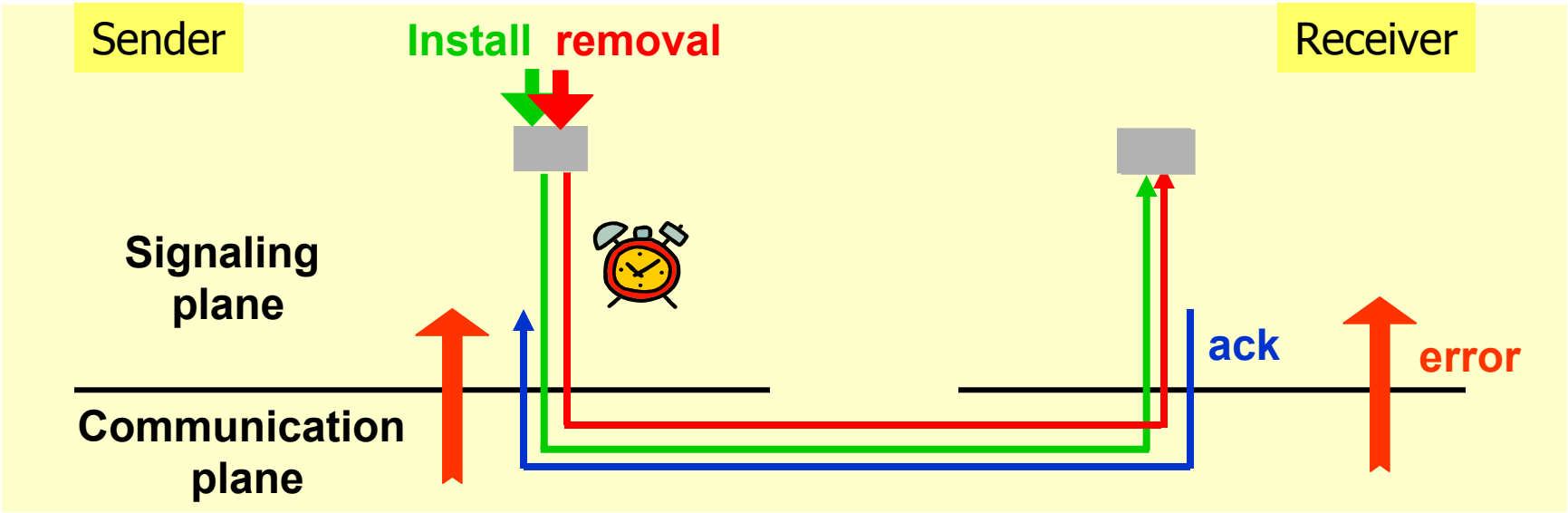
Hard-state

- State *installed* by receiver on receipt of *setup msg* from sender
- State *removed* by receiver on receipt of *teardown msg* from sender
- *Default assumption:* state valid unless told otherwise
  - In practice: failsafe-mechanisms (to remove orphaned state) in case of sender: e.g., receiver-to-sender "heartbeat": Is this state still valid ?
- Examples:
  - TCP
  - ST-II (Internet hard-state signaling)

Soft-state

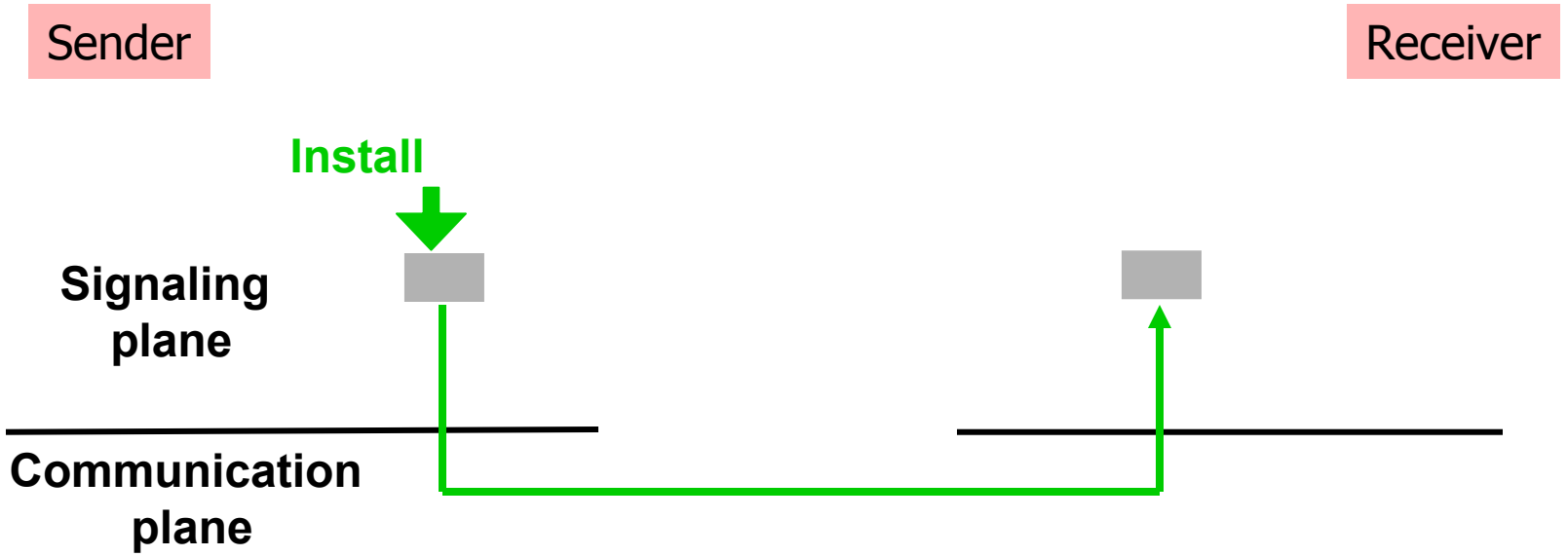
- State *installed* by receiver on receipt of setup (trigger) msg from sender (typically, an endpoint)
  - sender also sends periodic *refresh msg*: indicating receiver should continue to maintain state
- State *removed* by receiver via timeout, in absence of refresh msg from sender
- Default assumption: state becomes invalid unless refreshed
  - in practice: explicit state removal (*teardown*) msgs also used
- Examples:
  - RTP
  - RSVP

Hard-state signaling



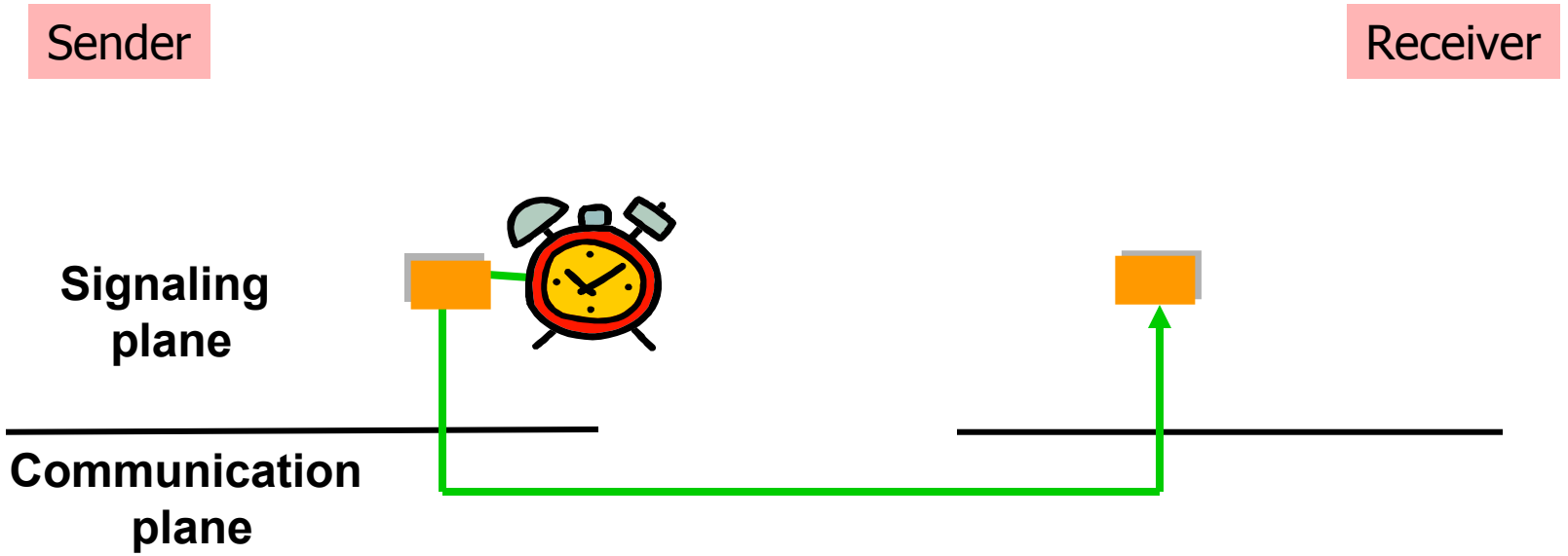
- Reliable signaling
- State removal by request
- Requires additional error handling
  - E.g., sender failure

Soft-state signaling



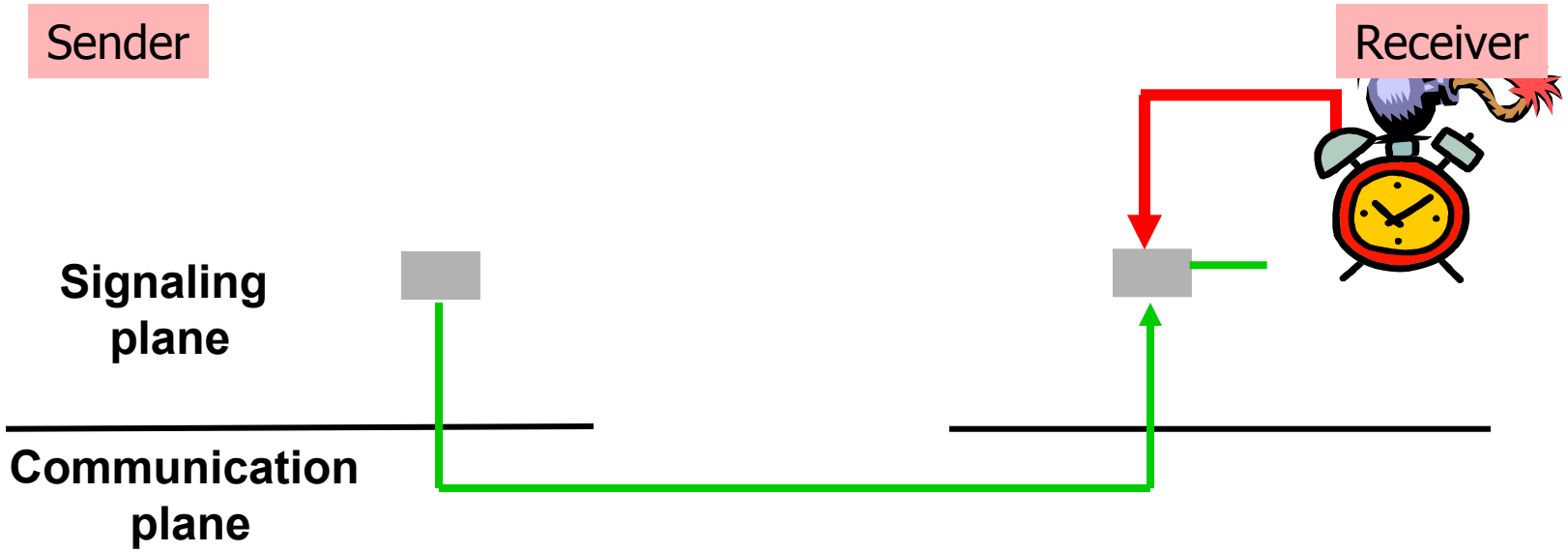
- Best effort signaling

Soft-state signaling



- Best effort signaling
- Refresh timer, periodic refresh

Soft-state signaling



- Best effort signaling
- Refresh timer, periodic refresh
- State time-out timer, state removal only by time-out

Soft-state: Claims

- "Systems built on soft-state are robust" [Raman 99]
- "Soft-state protocols provide … greater robustness to changes in the underlying network conditions …" [Sharma 97]
- "Obviates the need for complex error handling software" [Balakrishnan 99]

Hard-state versus soft-state: Discussion

Q: Which is preferable and why?

Hard state:

- better if message overhead really high
- potentially greater consistency
- system wide coupling → difficult to analyze

Soft state:

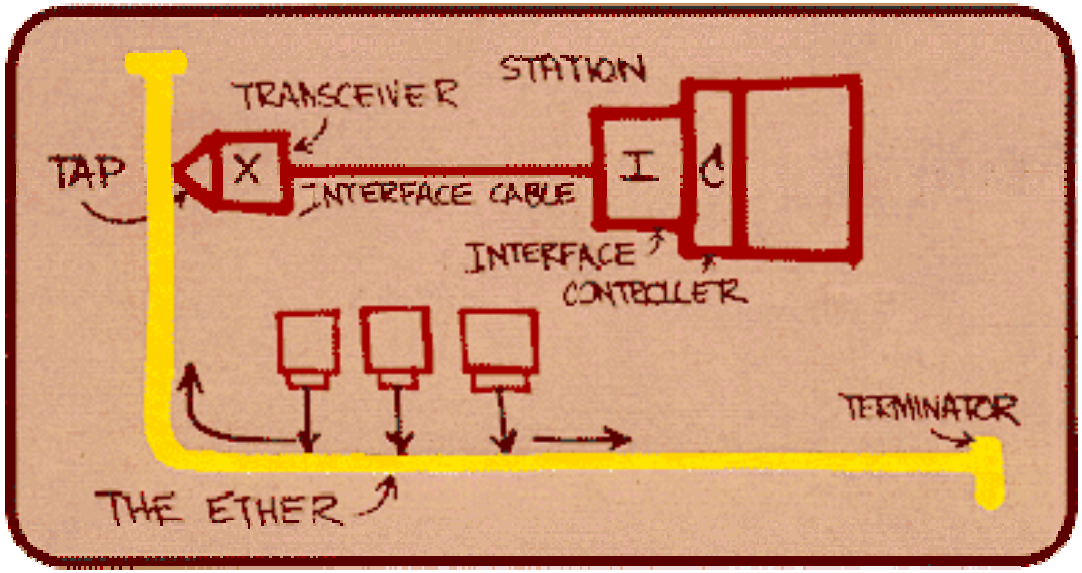
- robustness, shorter convergence times
- easily decomposed → simpler analysis

3. Randomization

- Randomization used in many protocols
- Examples:
  - Ethernet multiple access protocol
  - Randomization in Router Queue Management RED
  - Router (de)synchronization
  - Switch scheduling

Ethernet

- Single shared broadcast channel
- 2+ simultaneous transmissions by nodes: interference
  - only one node can send successfully at a time
- *Multiple access protocol:* distributed algorithm that determines how nodes share channel, i.e., determine when node can transmit



Metcalfe's Ethernet sketch

Ethernet's CSMA/CD

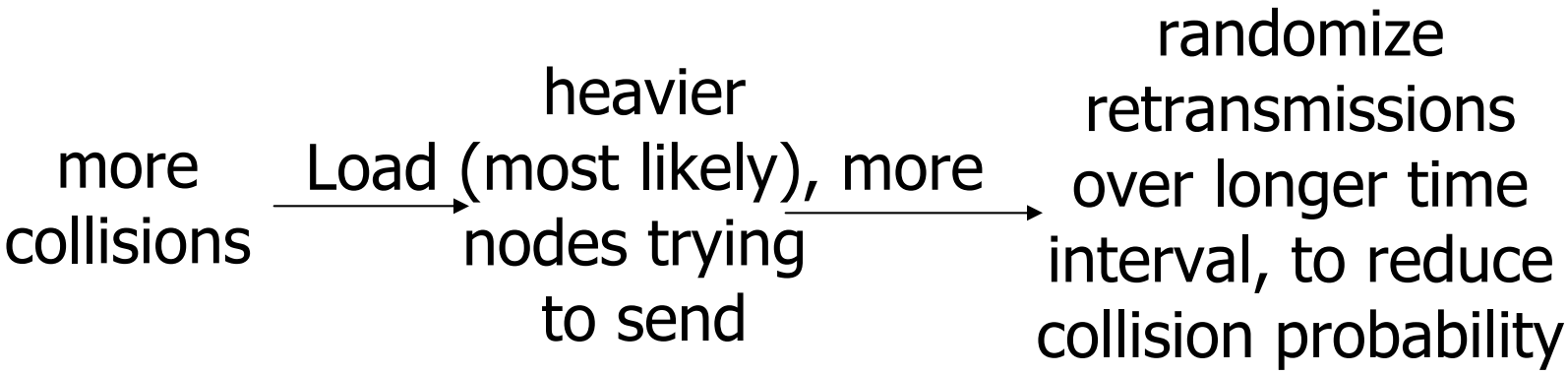
Jam Signal: make sure all other transmitters are aware of collision; 48 bits;

Exponential Backoff:

- First collision for given packet: choose K randomly from {0,1}; delay is K × 512 bit transmission times
- After second collision: choose K randomly from {0,1,2,3}…
- After ten or more collisions, choose K randomly from {0,1,2,3,4,…,1023}

Ethernet's use of randomization

- *Resulting behavior:* probability of retransmission attempt (equivalently length of randomization interval) adapted to current load
  - Simple, load-adaptive, multiple access



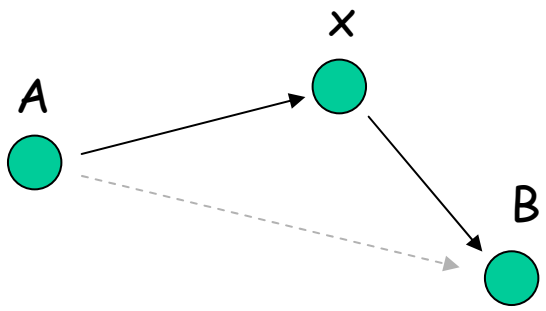
The bottom line

- Why does Ethernet use randomization:
  to desynchronize:

  A distributed adaptive algorithm to spread out load over time when there is contention for multiple access channel

4: Indirection

Indirection: Rather than reference an entity directly, reference it (“indirectly) via another entity, which in turn can or will access the original entity



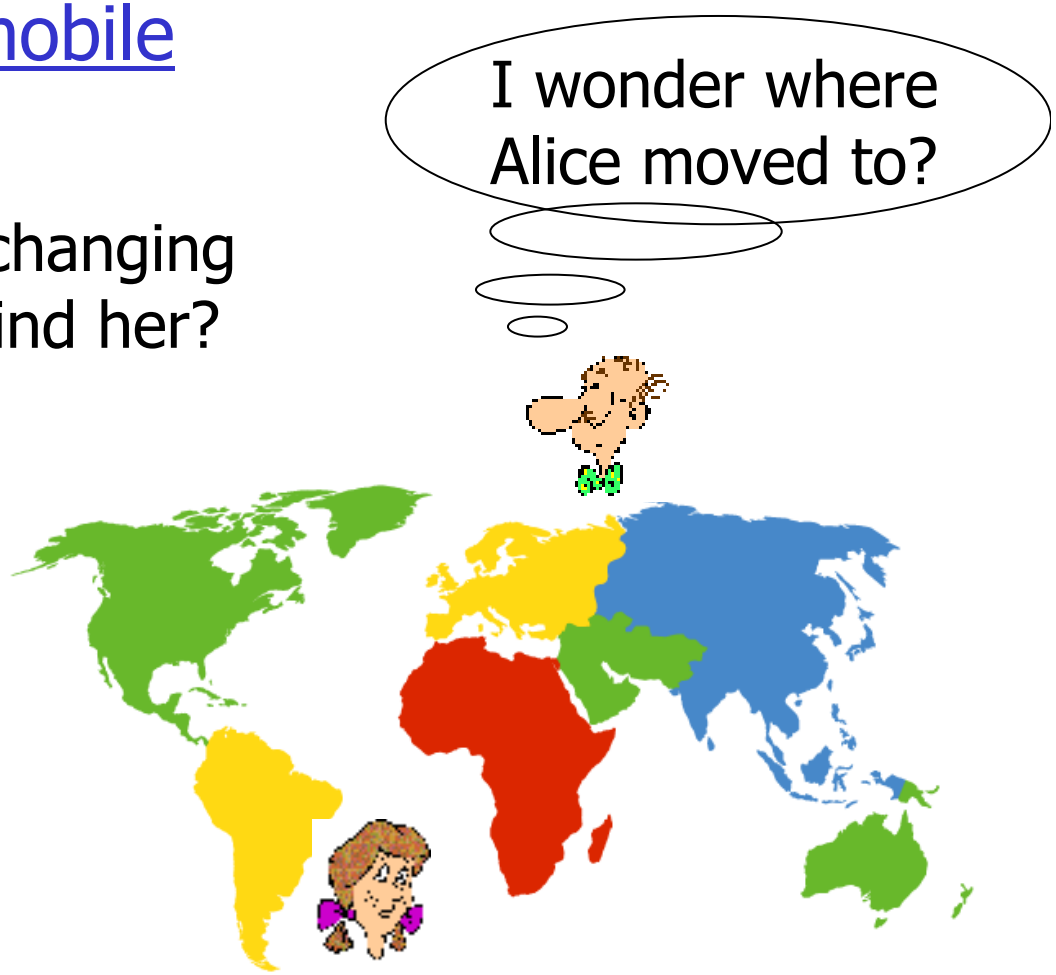
"Every problem in computer science can be solved by adding another level of indirection"
— Butler Lampson

Mobility and indirection

How do *you* contact a mobile friend?

Consider friend frequently changing addresses, how do you find her?

- Search all phone books?
- Call her parents?
- Expect her to let you know where he/she is?



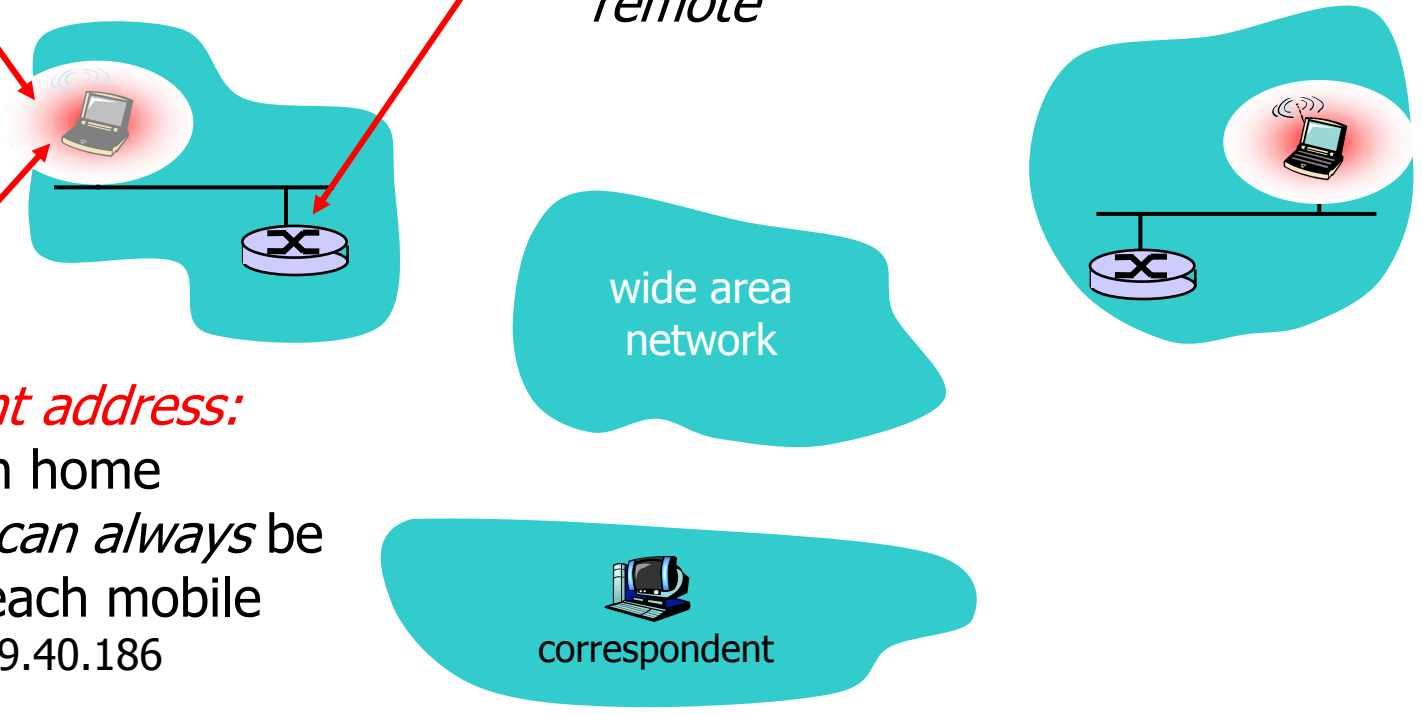
Mobility and indirection:

- Mobile node moves from network to network
- Correspondents want to send packets to mobile node
- Two approaches:
  - *Indirect routing:* communication from correspondent to mobile goes through home agent, then forwarded to remote
  - *Direct routing:* correspondent gets foreign address of mobile, sends directly to mobile

Mobility: Vocabulary

Home network: Permanent "home" of mobile (e.g., 128.119.40/24)

Home agent: *Entity that will perform mobility functions on behalf of mobile, when mobile is remote*



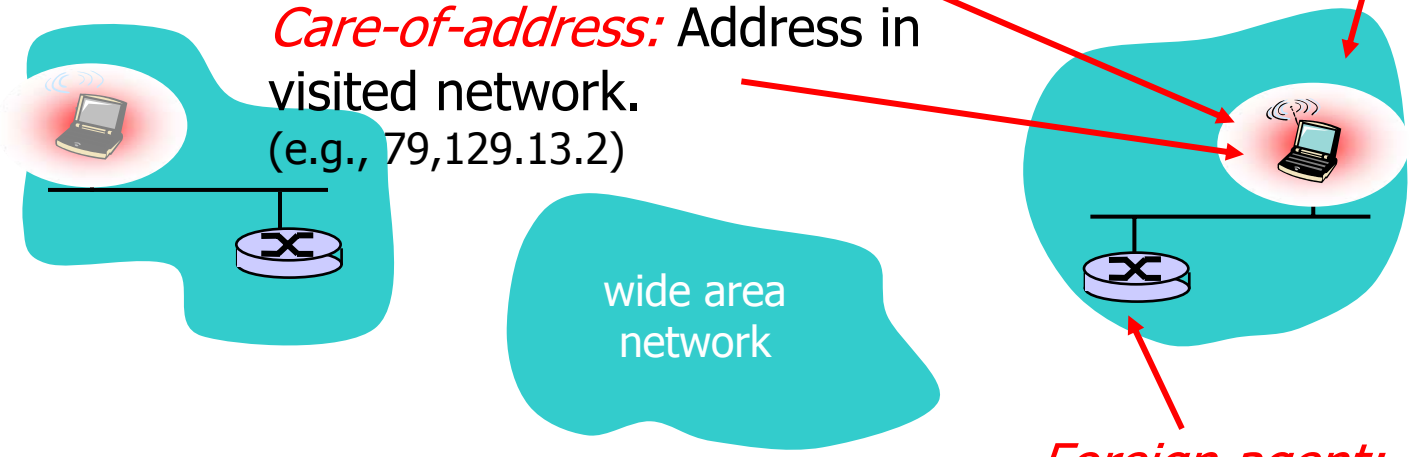
Permanent address: Address in home network, *can always* be used to reach mobile e.g., 128.119.40.186

Mobility: More vocabulary

Permanent address: Remains constant (e.g., 128.119.40.186)

Visited network: Network in which mobile currently resides (e.g., 79.129.13/24)

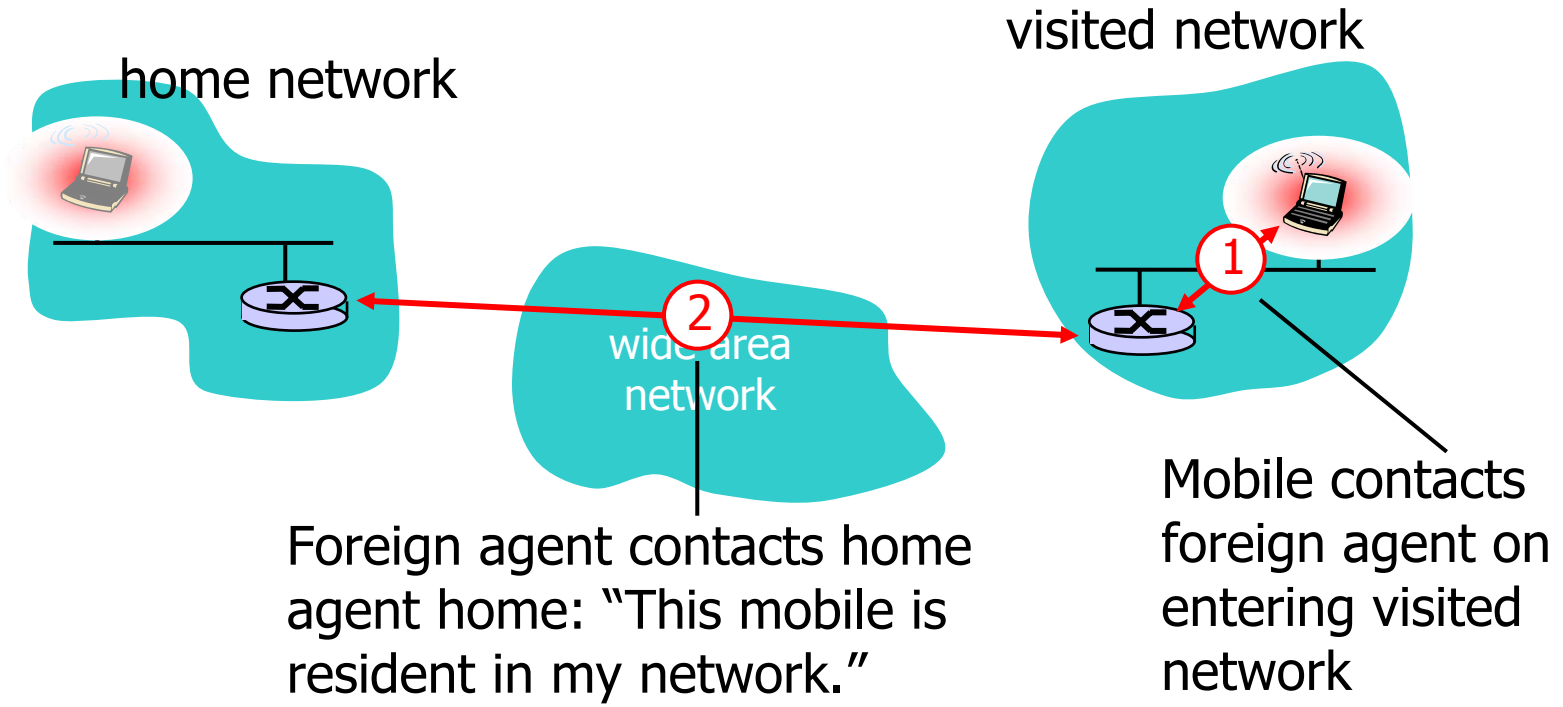
Care-of-address: Address in visited network. (e.g., 79,129.13.2)



Foreign agent: *Entity in visited network that performs mobility functions on behalf of mobile.*

Correspondent: *Wants to communicate with mobile*

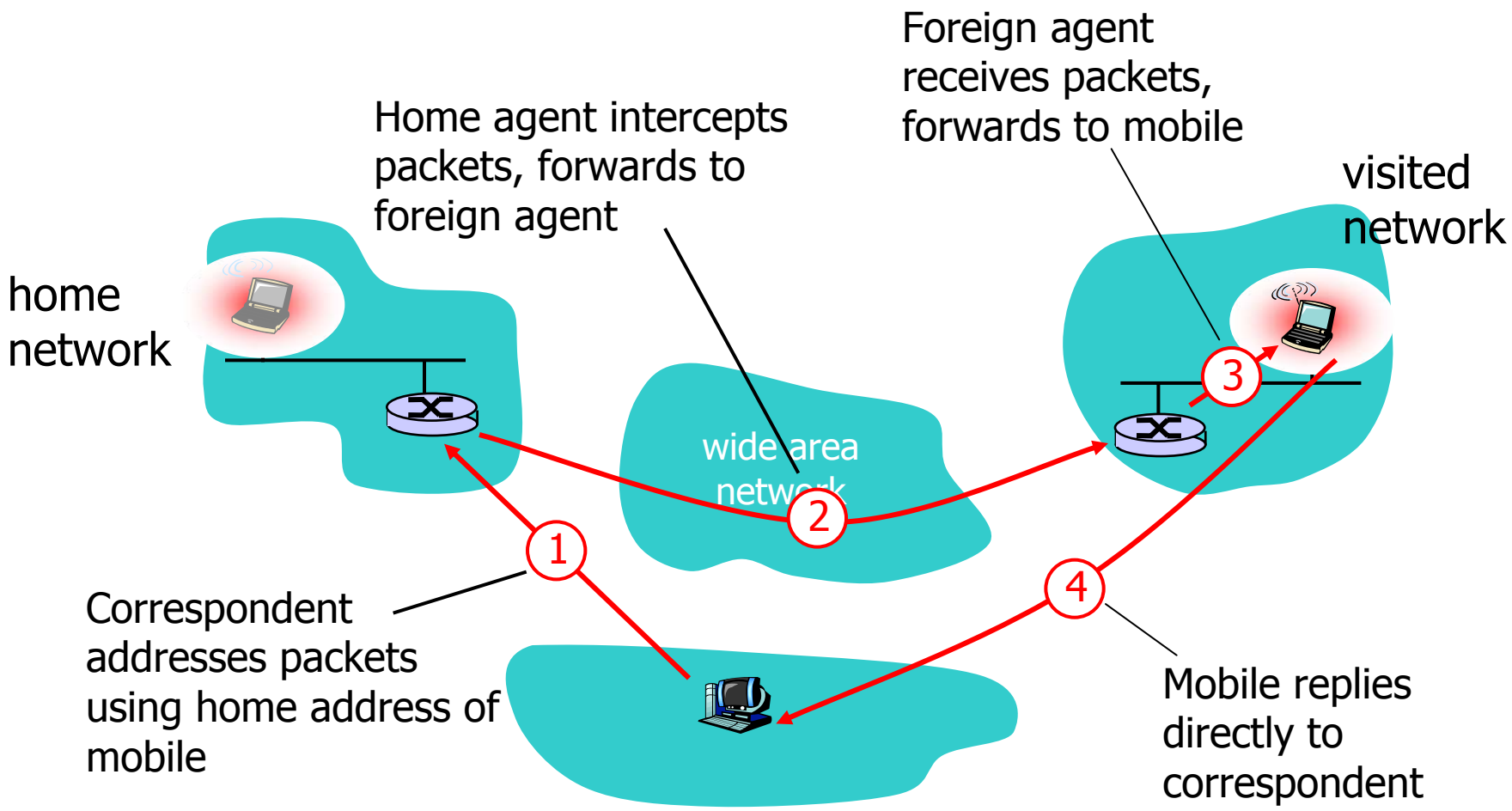
Mobility: Registration



End result:

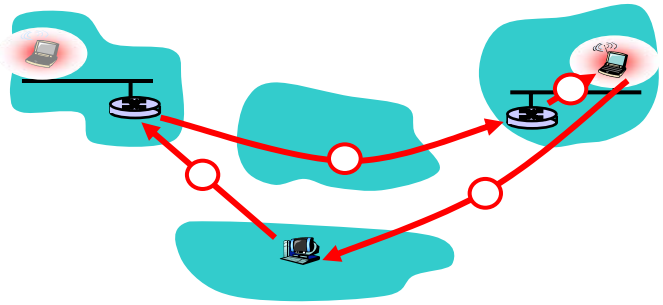
- Foreign agent knows about mobile
- Home agent knows location of mobile

Mobility via indirect routing



Indirect Routing: Comments

- Mobile uses two addresses:
  - Permanent address: Used by correspondent (hence mobile location is *transparent* to correspondent)
  - Care-of-address: Used by home agent to forward datagrams to mobile
- Foreign agent functions may be done by mobile itself
- Triangle routing: Correspondent-home-network-mobile
  - Inefficient when correspondent, mobile are in same network



Mobility via indirection: Why indirection?

- Transparency to correspondent
- "Mostly" transparent to mobile (except that mobile must register with foreign agent)
  - Transparent to routers, rest of infrastructure
  - Potential concerns if egress filtering is in place in origin networks (since source IP address of mobile is its home address)

Indirection: Summary

We've seen indirection used in many ways:

- Mobility
- Multicast

The uses of indirection:

- Sender does not need to know receiver id - do not *want* sender to know intermediary identities
- Beauty, grace, elegance
- Transparency of indirection is important
- Performance: is it more efficient?

5. Multiplexing

Multiplexing: *Sharing* resource(s) among users of the resource.

Multiplexed human resources:

- Roadways, traffic intersections
- Bathrooms (except for the rich)
- Reserve reading
- …

6. Virtualization of networks

Virtualization of resources:

powerful abstraction in systems engineering

- Computing examples: virtual memory, virtual devices
  - Virtual machines: e.g., Java
  - IBM VM OS from 1960's/70's
- Layering of abstractions: don't sweat the details of the lower layer, only deal with lower layers abstractly

The Internet: Virtualizing local networks

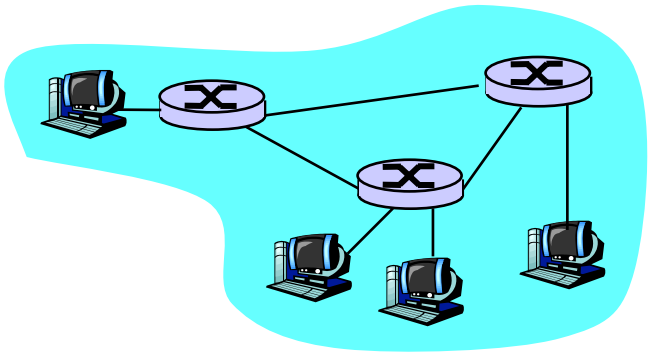
1974: multiple unconnected networks

- ARPAnet
- Data-over-cable networks
- Packet satellite network (Aloha)
- Packet radio network

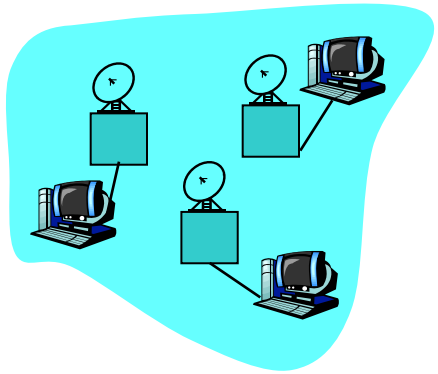
... differing in:

- Addressing conventions
- Packet formats
- Error recovery
- Routing

Cerf & Kahn: Interconnecting two networks



ARPAnet



satellite net

- "… interconnection must preserve intact the internal operation of each network."
- "... the interface between networks must play a central role in the development of any network interconnection strategy. We give a special name to this interface that performs these functions and call it a GATEWAY."
- "... prefer that the interface be as simple and reliable as possible, and deal primarily with passing data between networks that use different packet-switching strategies
- "… address formats is a problem between networks because the local network addresses of TCP's may vary substantially in format and size. A uniform internetwork TCP address space, understood by each GATEWAY and TCP, is essential to routing and delivery of internetwork packets."

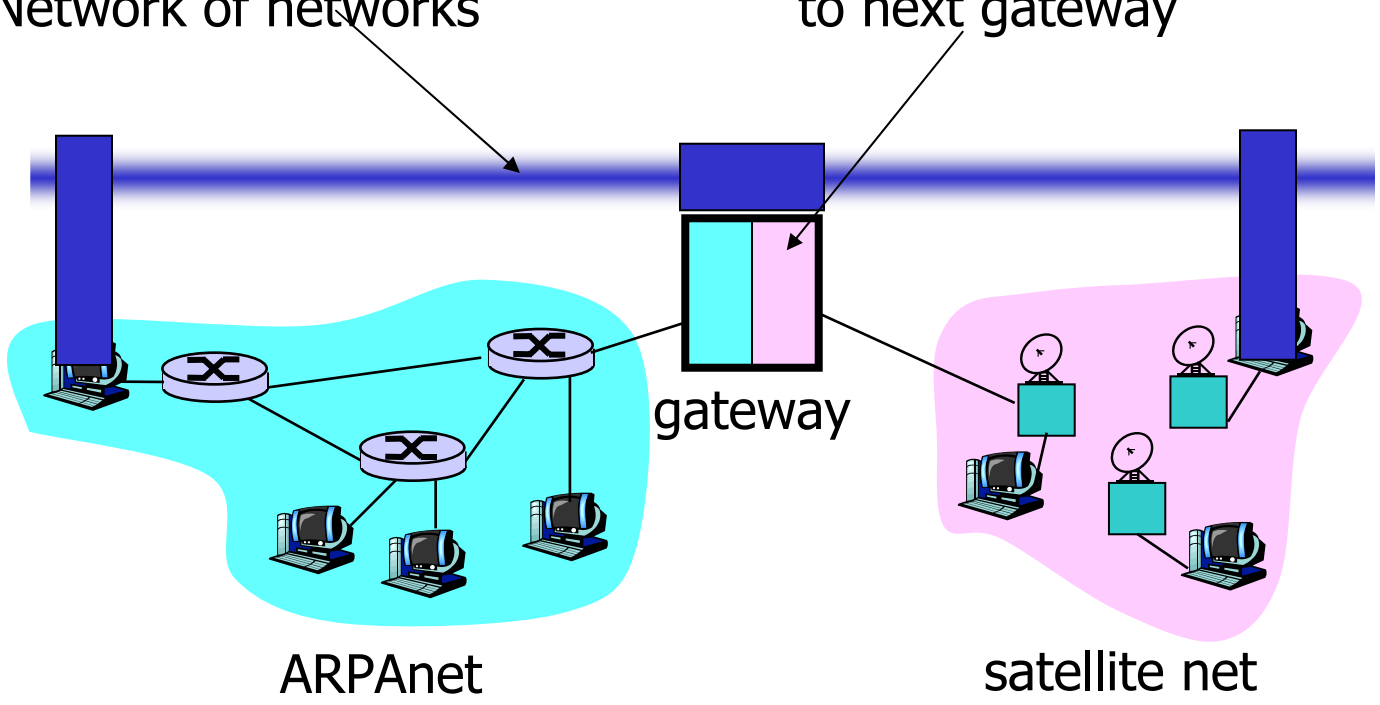
Cerf & Kahn: Interconnecting two networks

Internetwork layer:

- Addressing: internetwork appears as a single, uniform entity, despite underlying local network heterogeneity
- Network of networks

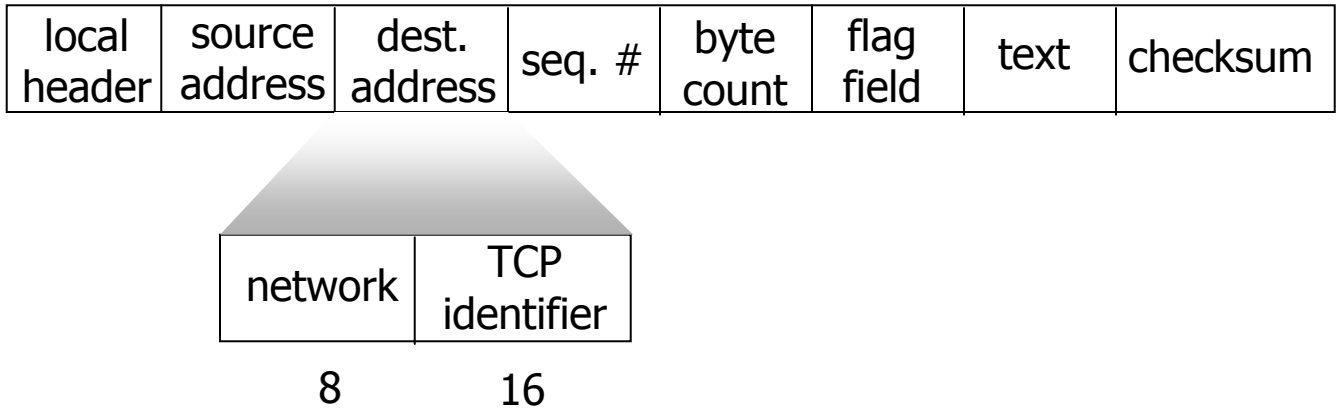
Gateway:

- "Embed internetwork packets in local packet format or extract them"
- Route (at internetwork level) to next gateway



Historical aside

Proposed Internetwork packet in 1974:



Cerf & Kahn's Internetwork Architecture

What is virtualized?

- Two layers of addressing:
  Internetwork and local network
- New layer makes everything homogeneous
- Underlying local network technology (cable, satellite, 56K modem) is "invisible" at internetwork layer

7. Designs for scale

How to deal with large numbers (millions) of entities in a system?

- IP devices in the internet (0.5 billion)
- Users in P2P network (millions)

More generally:

- Are there advantages to large scale?
- "For every type of animal there is a most convenient size, and a large change in size *inevitably* carries with it a change of form."
  True for networks?

Dealing with scale: Hierarchical routing

Scale: with 500 million destinations:

- Can't store all dest's in routing tables!
- Routing table exchange would swamp links!

Administrative autonomy

- Internet = network of networks
- Each network admin may want to control routing in its own network

Dealing with wcale

Question: What are the *advantages* of large scale?

- Take advantage of having to do similar things for others (caching)
- Fault tolerance:
  - Large number of servers
  - We have redundancy; multiple routes between sites
- Metcalfe's law:
  - "Value" of a network is proportional to square of number of things connected (bigger is better)
- Law of large numbers:
  - Allocation of resources based on average usage rather than peak
- Amortizing upgrade maintenance over a large population:
  - Popular network and services likely to be upgraded/improved
- Denial of service:
  - Size/replication makes it harder to attack
  - More generally, a system with replicated components is more survivable.

Dealing with scale

Discussion: "For every type of animal there is a most convenient size, and a large change in size inevitably carries with it a change of form."

Question: True for networks? Why? How so? Examples?

- Ethernet doesn't scale up: geographical distance, speed of light delays degrade performance of random access protocols. (geographic scaling). Maybe scale with # users in geographically narrow net if bandwidth scales with users.
- As number of communicants scales, need to change/improve manner in which to access communication channel
  - Example: Small number of students, versus 500-class lecture.

Dealing with scale

Discussion: "For every type of animal there is a most convenient size, and a large change in size inevitably carries with it a change of form."

Question: True for networks? Why? How so? Examples?

- Routing:
  - Large number of users and optimal routes
    => requires lots of info to compute routes, etc. ...
  - Doesn't scale
- Certain services become necessary when you get big
  - Name storage/translation: dns, phone books
- A single centralized site eventually breaks
  - Need replication or other form of distribution

End of "Design Principles"!

Goals

- Review
- Material that is timely, timeless: hot now but also long shelf life
- Synthesis: Deeper understanding; "see the forest for the trees"