CS 755 – System and Network Architectures and Implementation

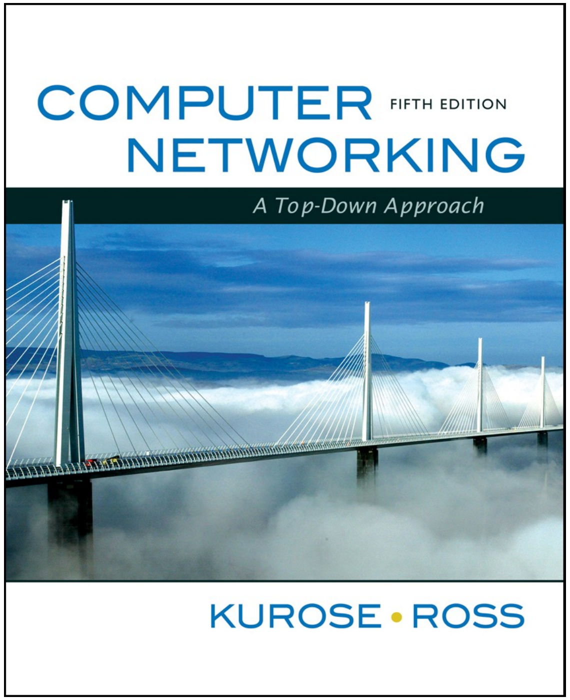
Module 5 – Naming and Mobility

Martin Karsten

mkarsten@uwaterloo.ca

Notice

Some slides and elements of slides are taken from third-party slide sets. In this module, parts are taken from the Kurose/Ross slide set. See detailed statement on next slide...



A note on the use of these ppt slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) in substantially unaltered form, that you mention their source (after all, we'd like people to use our book!)
- If you post any slides in substantially unaltered form on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

Computer Networking: A Top Down Approach
5th edition.
Jim Kurose, Keith Ross
Addison-Wesley, April 2009.

Overview

- naming
  - manage and find entities and services
  - examples: DNS, DHT
- mobility
  - naming + maintain connectivity
  - example: Mobile IP

Services – Review

- messaging middleware
  - persistent communication
  - asynchrounous communication
- remote procedure call
  - transparency
- security
  - privacy, integrity, authentication

Naming

“Of what one cannot speak,
one must pass over in silence.”

Ludwig Wittgenstein, *Tractatus*

Naming

- names in (distributed) computer systems
  - identification, (permanent) uniqueness – *scope*
  - facilitate communication / access – *resolution*
  - description of entity – *context*
- static vs. dynamic naming
  - online vs. offline *agreement*
  - online: distributed vs. centralized
  - manual configuration

Name Resolution

- access named object / entity
- direct access
  - forward message to destination
- indirect access
  - map name to other name using database (*lookup*)
  - might be a distributed database

  -> forwarding within distributed database?

Definition Attempt

- a *name* is a handle/reference, valid and unique in a *scope*, that can be used to access a (group of) object(s) via a *resolution* mechanism

- try your own...

Everything is a Name

- memory address
- file system inode / name, socket handle
- MAC address, IP address, port, DNS name
- URL
- email address, Skype ID, Twitter ID
- phone number
- service identifier: WSDL, WSIL, UDDI
- “plumbing contractor kitchener ontario”

Conflicting Goals

- uniqueness, permanence - *identifier*
  - numbering scheme, large range
- access / communication - *locator*
  - location-dependent name (distance metric?)
  - efficient processing for forwarding
- description - *descriptor*
  - precision
  - processing overhead

Discovery

- human-oriented description
  - expressive vs. concise

  vs. machine processing, efficiency
  - basic semantic schism: human vs. machine

- essential mechanism: search
  - range queries
  - typos, unclear intent
  - multiple results

Name Processing

- At any point during name processing:
  scope, resolution, and context are implicit!

- if explicit, they are described by a label
  ... which is mapped into something else
  ... therefore the label is a name
  ... which needs scope, resolution, and context
  => recursive contradiction!

Abstract Names

- role name
  - 'mkarsten' vs. 'root'
  - 'joe@email.com' vs. 'admin@isp.com'

- service name
  - print.cs.uwaterloo.ca
  - ipp://print.cs.uwaterloo.ca:631/printers/color

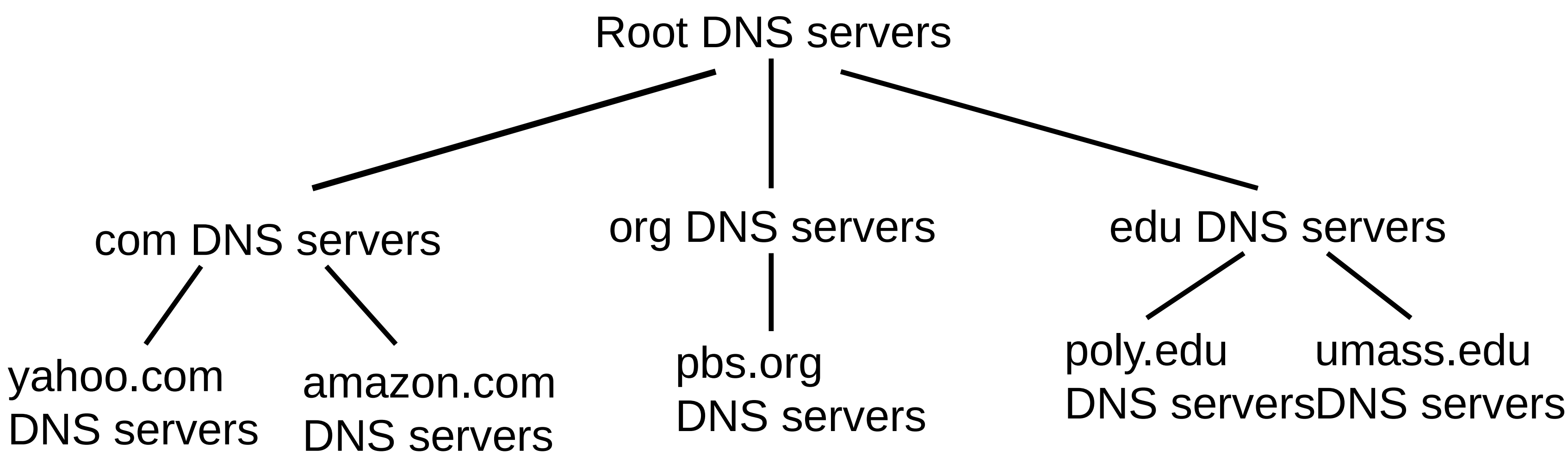
Distributed Naming System

- name assignment
  - alias support
- name resolution
  - resolution overhead
  - storage overhead
  - scalability, caching
    - size of name space, number of managed entities
    - frequency of updates and lookups
  - relative vs. absolute names

Example: DNS

- <u>D</u>omain <u>N</u>ame <u>S</u>ystem
- hierarchical host names in the Internet
  - example: `cpu08.student.cs.uwaterloo.ca`
    - sequence of labels, written with separator
    - cf. file system name: /home/mkarsten/cs755/network.pdf
  - naming conventions
    - top level domain: country code or orga-type code
    - 2nd level: organizational name
    - etc. - local conventions (org units, depts, ...)
    - lowest level: role-based name (www, print, cpu, ...)

DNS Database



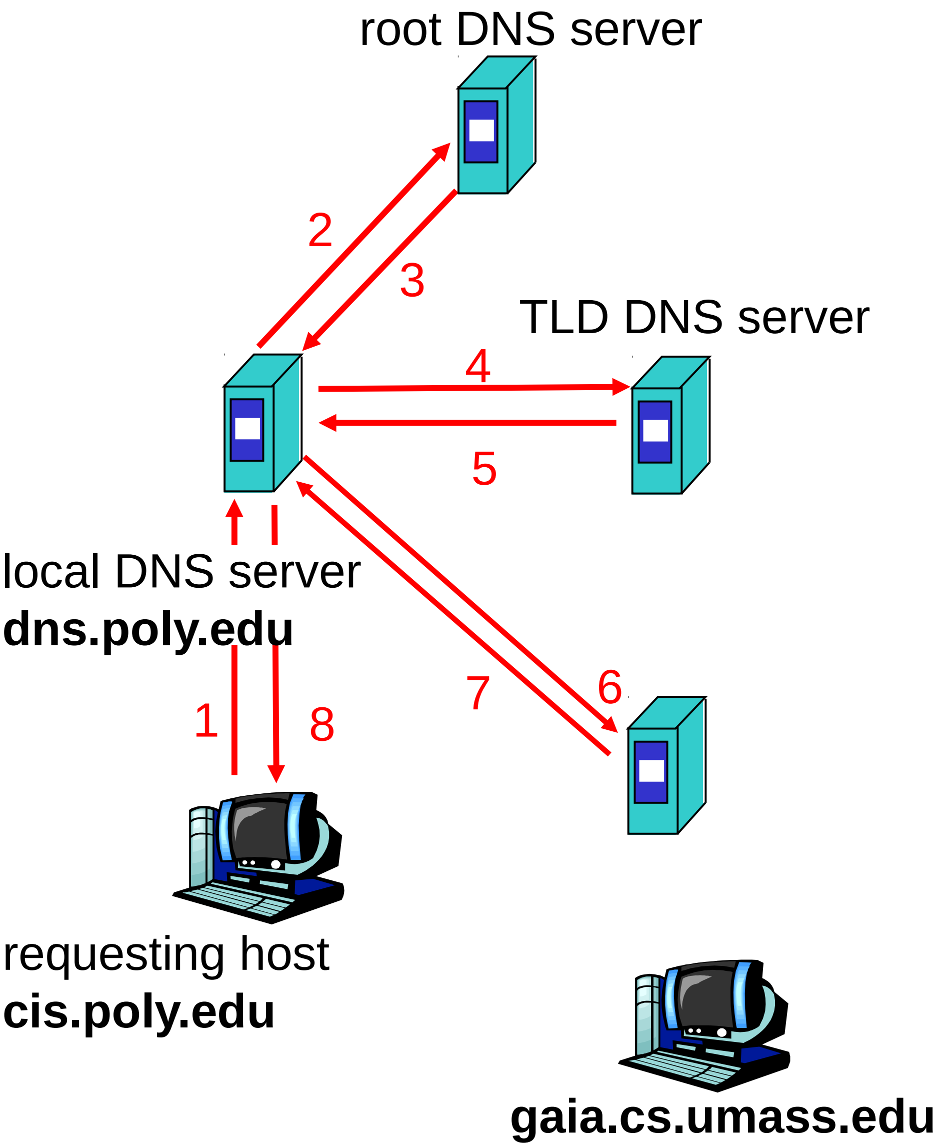
- distributed database - entry point: root server(s)
  - hierarchy (mostly) follows naming hierarchy
  - resolution requests traverse hierarchy
- local caching

Caching

- local DNS server
  - not part of authoritative hierarchy
  - caching proxy for DNS requests
  - often co-located with regular DNS server
  - “default name server”
- hierarchy of caching servers
- DNS updates are not frequent
  - and not expected to propagate fast

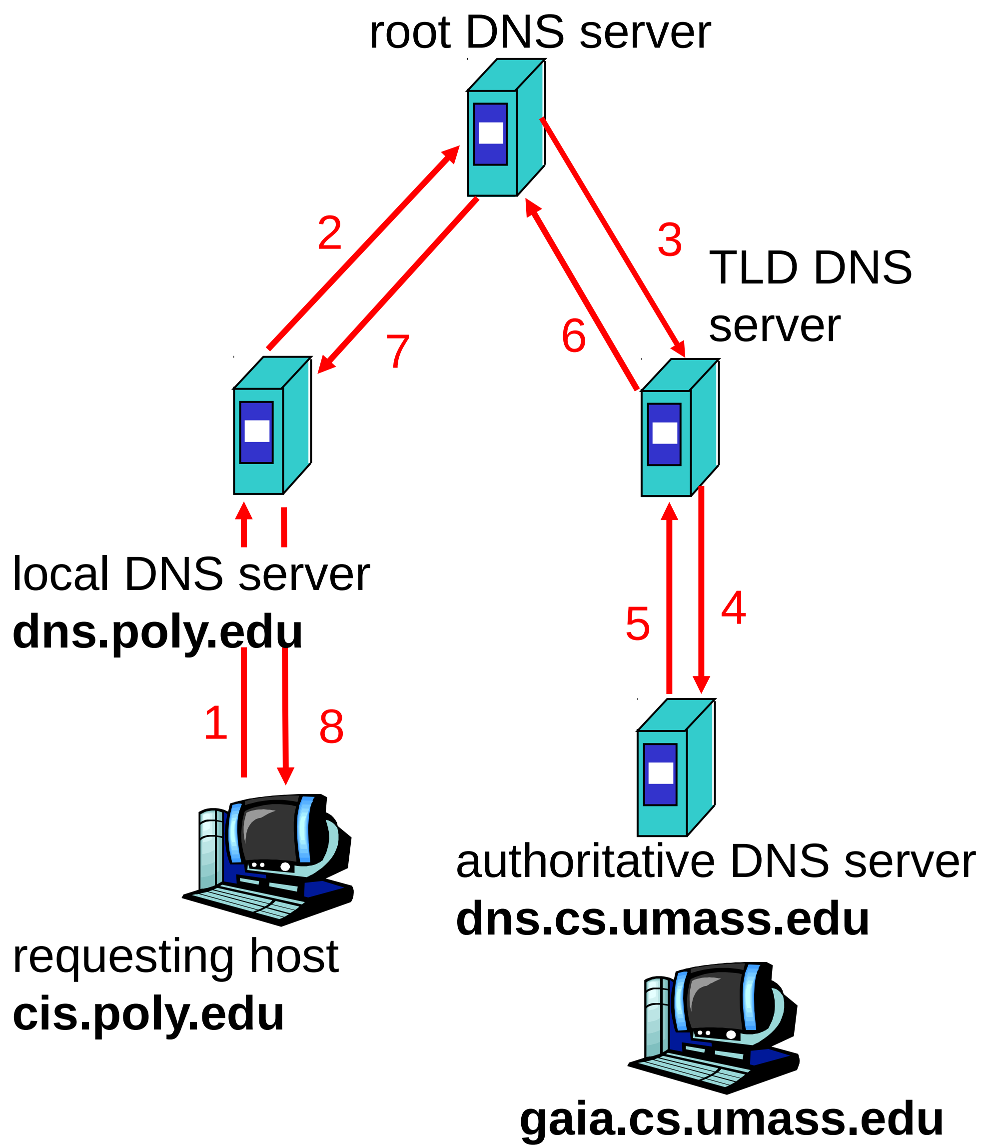
DNS Lookup

- iterative resolution
  - server replies with name/address of next server to contact
  - resolver can cache multiple intermediate results



DNS Lookup

- recursive resolution
  - server forwards request and replies with ultimate response
  - high-level servers can cache lots of results
  - load on high-level servers?
- recursive vs. iterative? => mix & match!



DNS Records

- DNS resource records (RR)

  name, value, type, ttl
- Type A: hostname -> IP address
- Type NS: domain name -> name server name
- Type CNAME: hostname -> hostname (alias)
- Type MX: email domain -> mail server name
- etc.

DNS – Notes

- simple request/reply protocol using UDP
  - retransmission, message identifiers
  - recursion optional
- ownership & regulation
  - IP address has technical meaning
  - DNS name has business value
- security & authentication
  - 'www.personalbank.com' redirected to bad party?

DNS – Notes

- replicated root servers
- relative names?
  - difficult, because resolution logically starts at root
- special functionality
  - multiple entries for name
    - return random value -> basic load balancing
    - server location -> geographical load balancing
- relevance in the age of Google?
  - how important is it to own www.mycompany.com?

Example: Distributed Hash Table

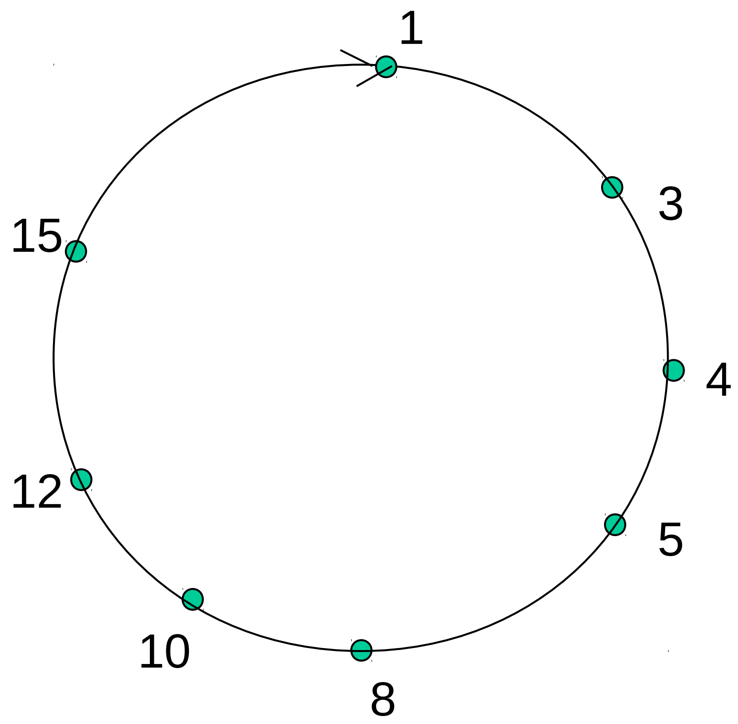
- flat identifier space for peer-to-peer applications
- DHT: distributed database for (key,value) pairs
- peers can insert (key,value) pairs
- peers query with key
  - DB returns with value that matches key
- identifier for each peer in [0...2^n-1]
- keys are taken from the same range
  - e.g., use hashing

Location of Values

- assign (key,value) to immediate successor peer of key, i.e., peer with smallest larger ID
- example: n = 4, peers: 1, 3, 4, 5, 8, 10, 12, 14
  - key = 14, successor peer = 14
  - key = 15, successor peer = 1

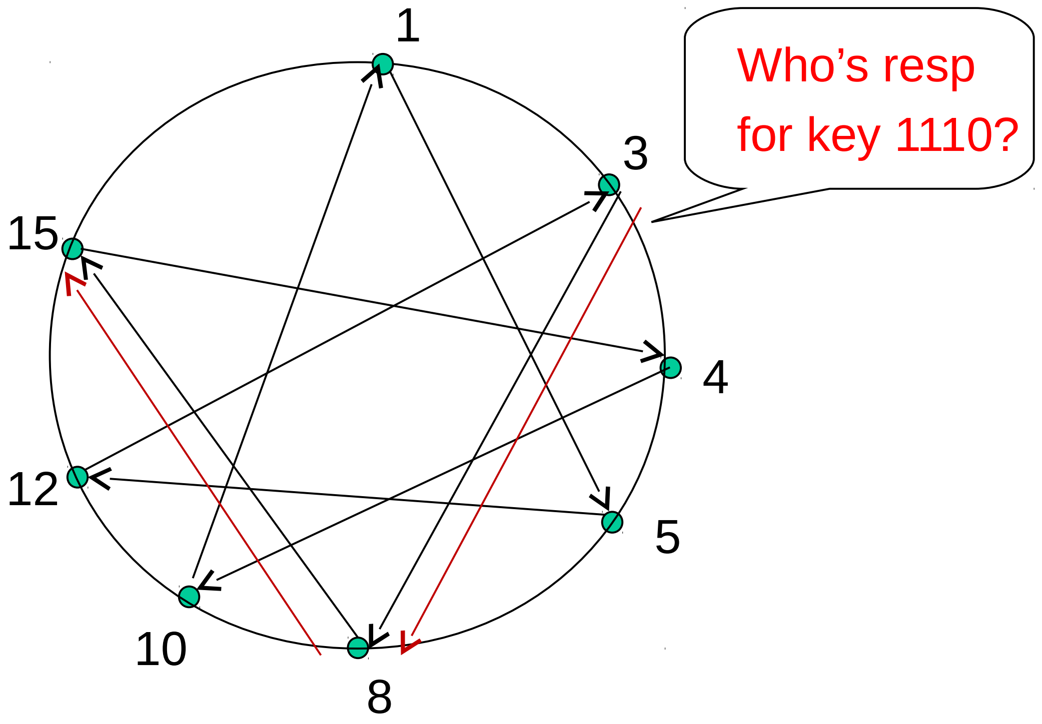
=> logical ring

Logical Ring Structure



- virtual network ("overlay")
- linear search?

Circular DHT with Shortcuts



- each peer has shortcuts to O(logN) peers
  - average distance 1, 2, 4, 8, 16, etc. (randomized)
- expected lookup in O(logN) steps

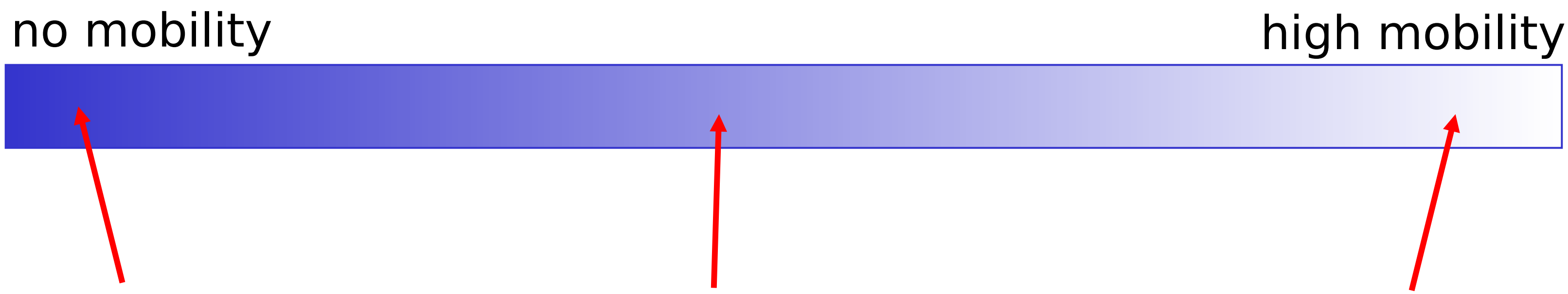
DHT Challenges

- decentralized management
- join / leave operations
  - periodic ping of successors
  - eliminate peers that have left from structure
  - join needs starting point and hook into the structure

- security?
- overhead with high churn?

Mobility

- spectrum of mobility, from network perspective



stationary wireless user, using same access point

mobile user, connecting/ disconnecting from network using DHCP.

mobile user, passing through multiple access point with ongoing sessions (cell phone)

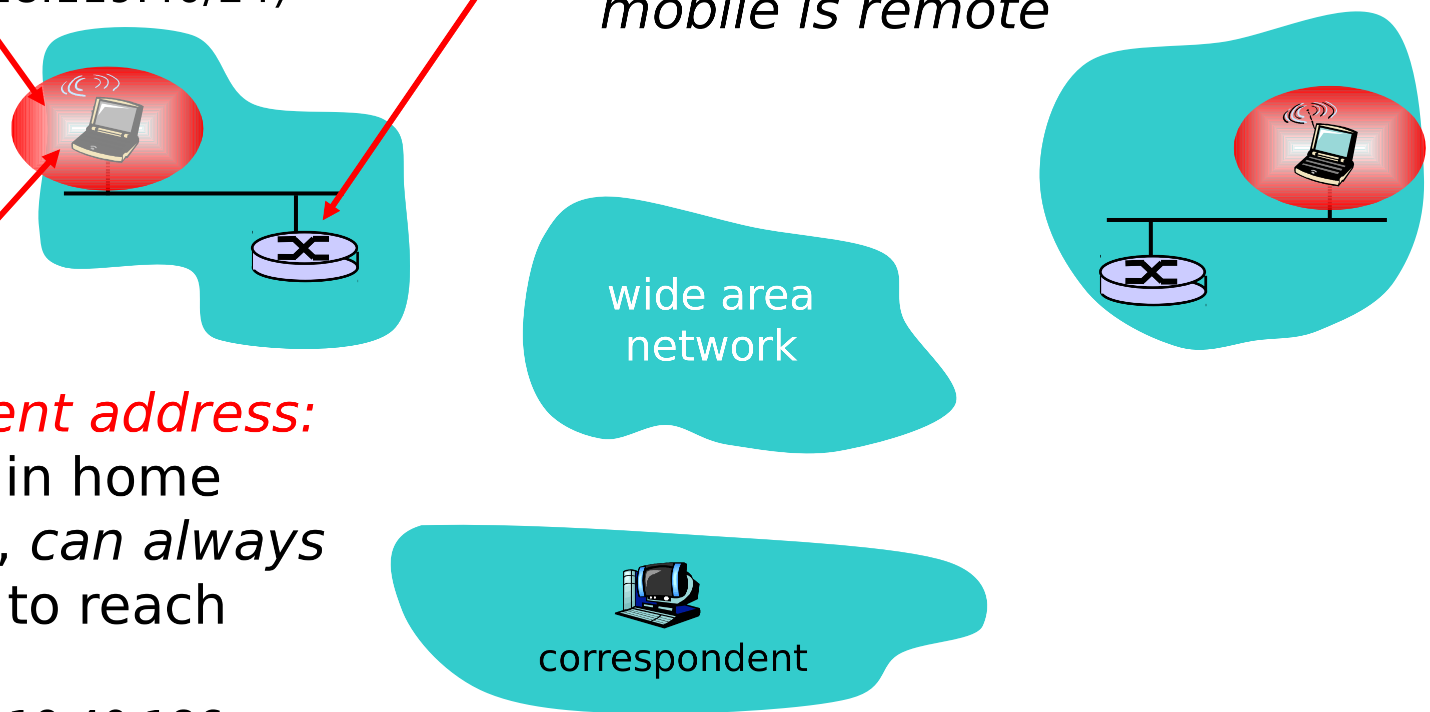
Mobility

- network access point
  - nearest network-level stationary router
  - might or might not be wireless access point
- challenges
  - initial lookup of responder
    - needs some form of registry
  - ongoing connectivity with movements
    - needs some form of update/redirection
  - similar, but different: time scale – session vs. packet

Example: Mobile IP

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



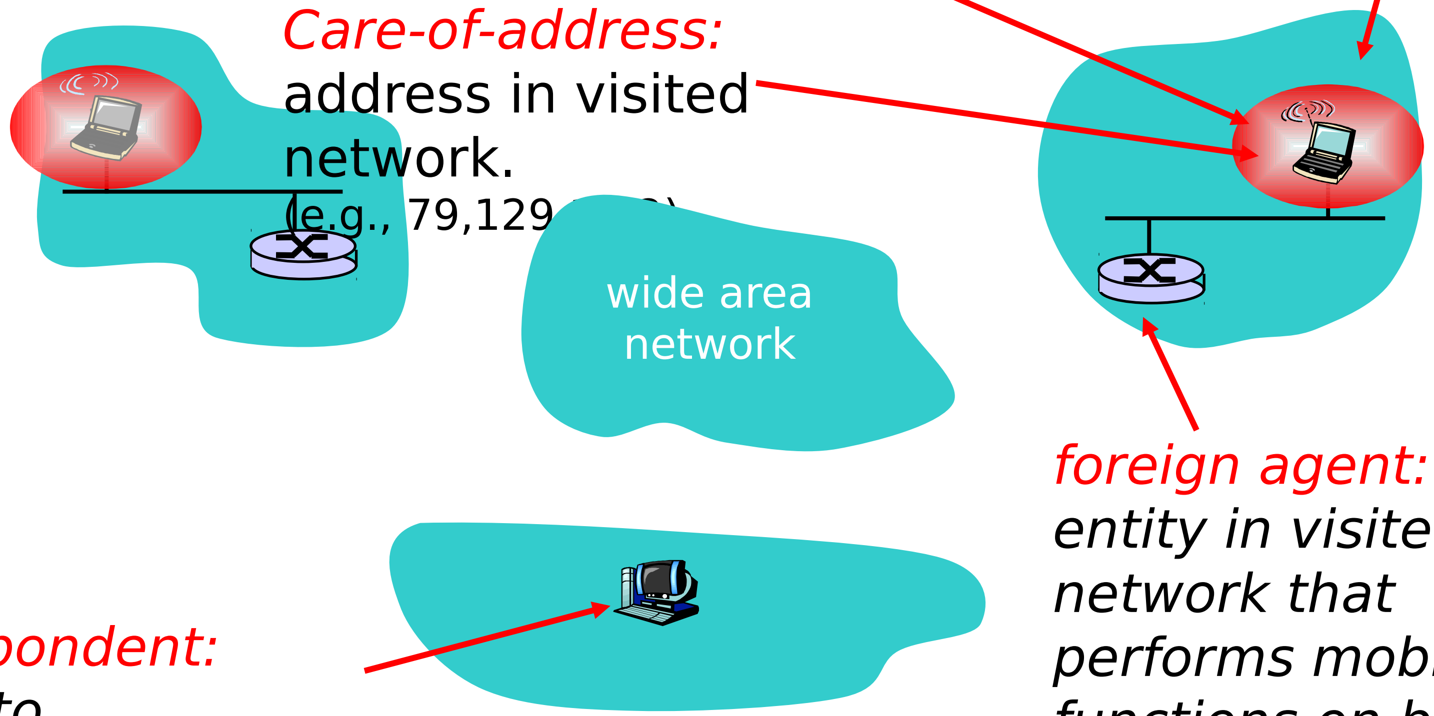
Permanent address: address in home network, *can always* be used to reach mobile
e.g., 128.119.40.186

Mobile IP

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

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

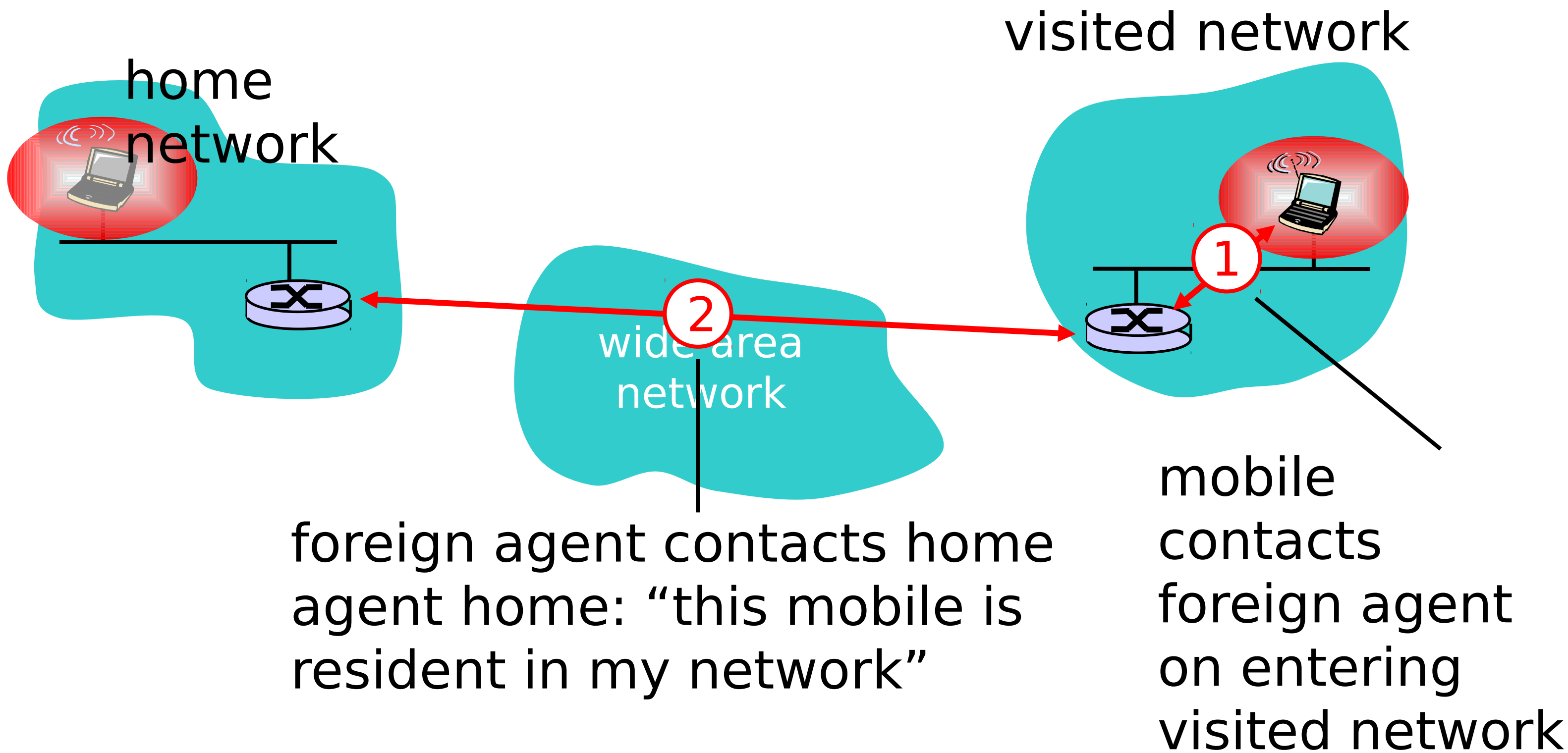
Care-of-address: address in visited network. (e.g., 79,129



foreign agent: entity in visited network that performs mobility functions on behalf of mobile.

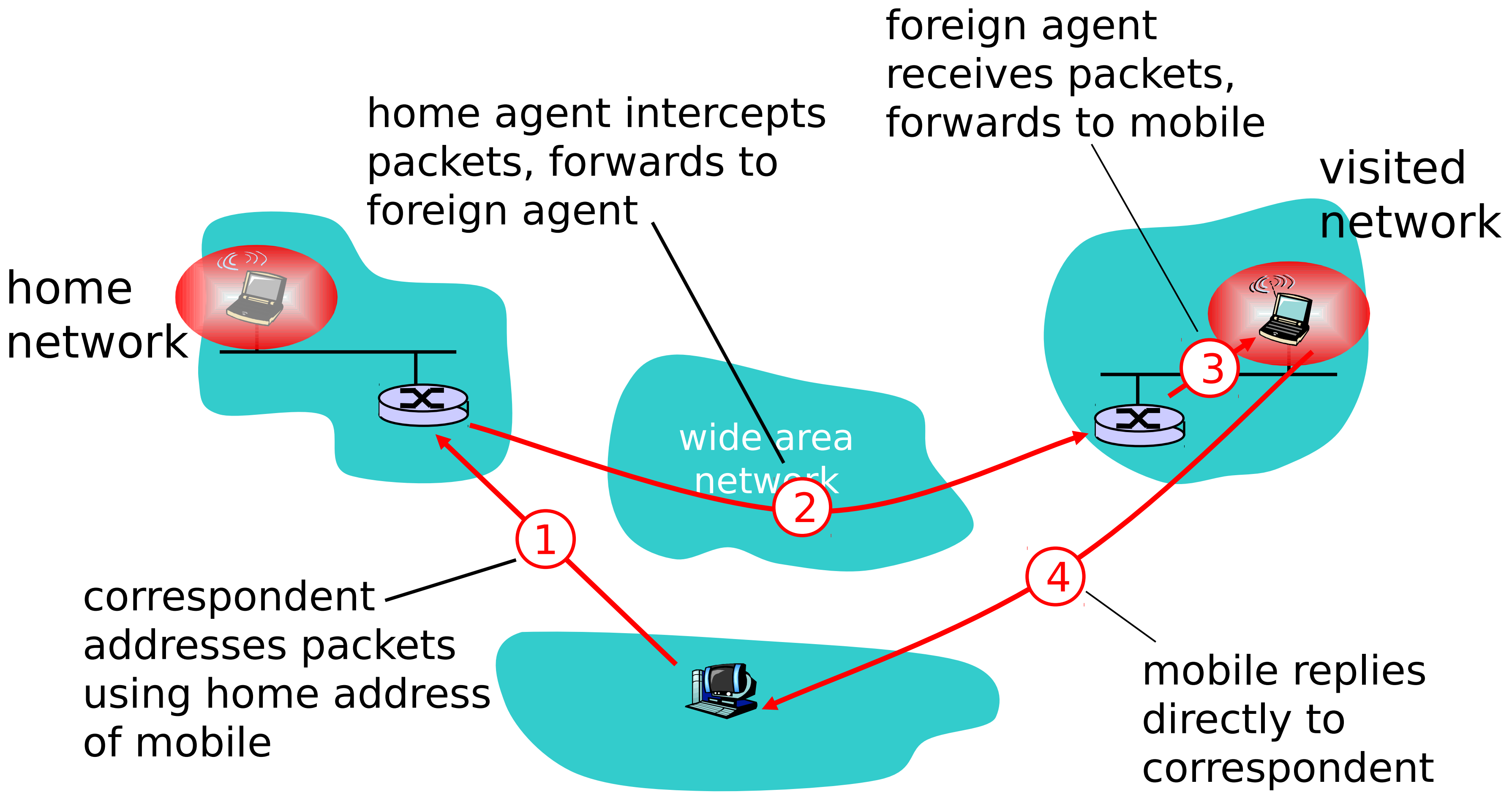
correspondent: wants to communicate with mobile

Mobile IP – Registration

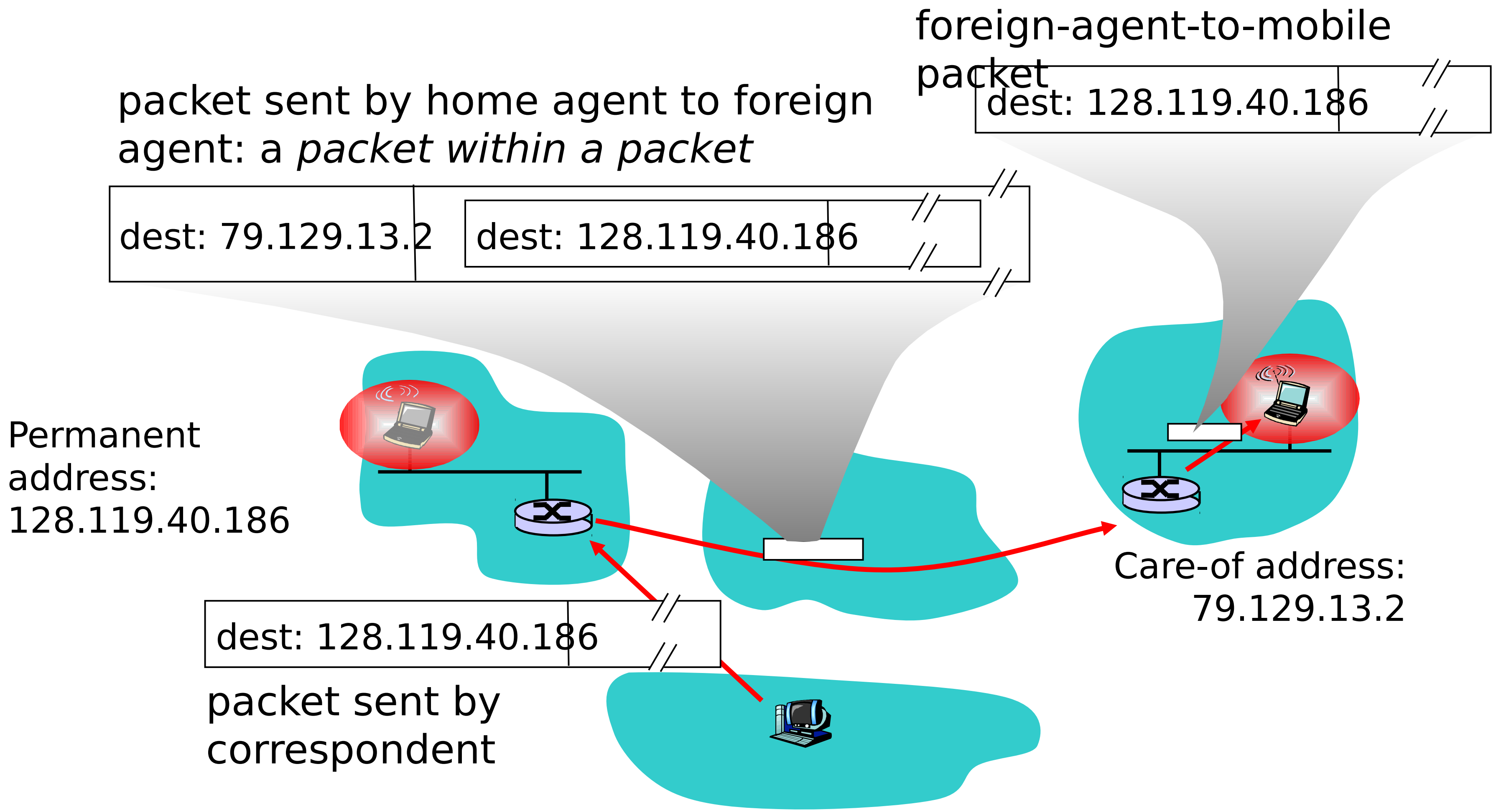


- foreign agent knows about mobile
- home agent knows location of mobile

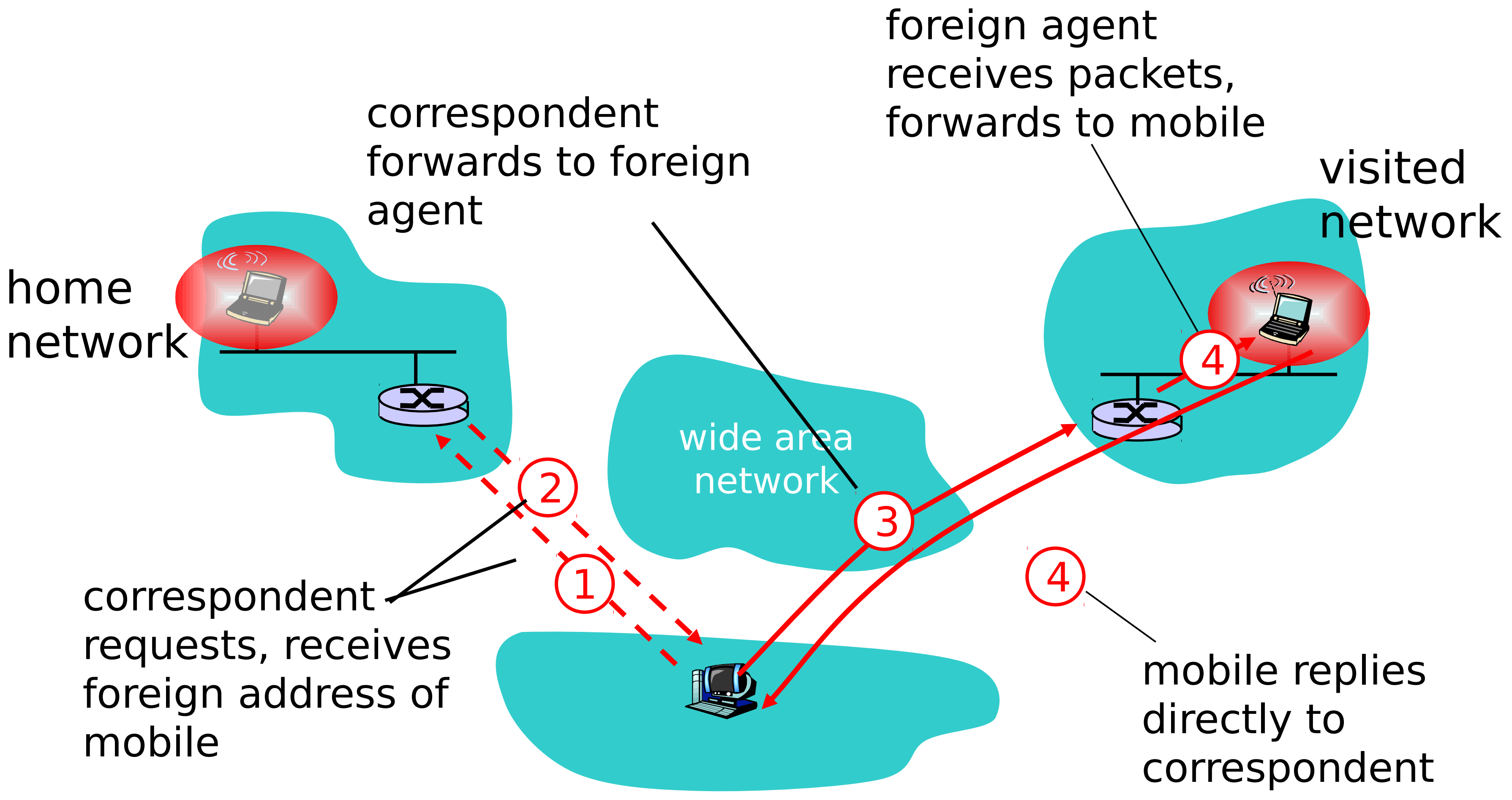
Mobile IP – Indirect Routing



Mobile IP – Indirect Routing

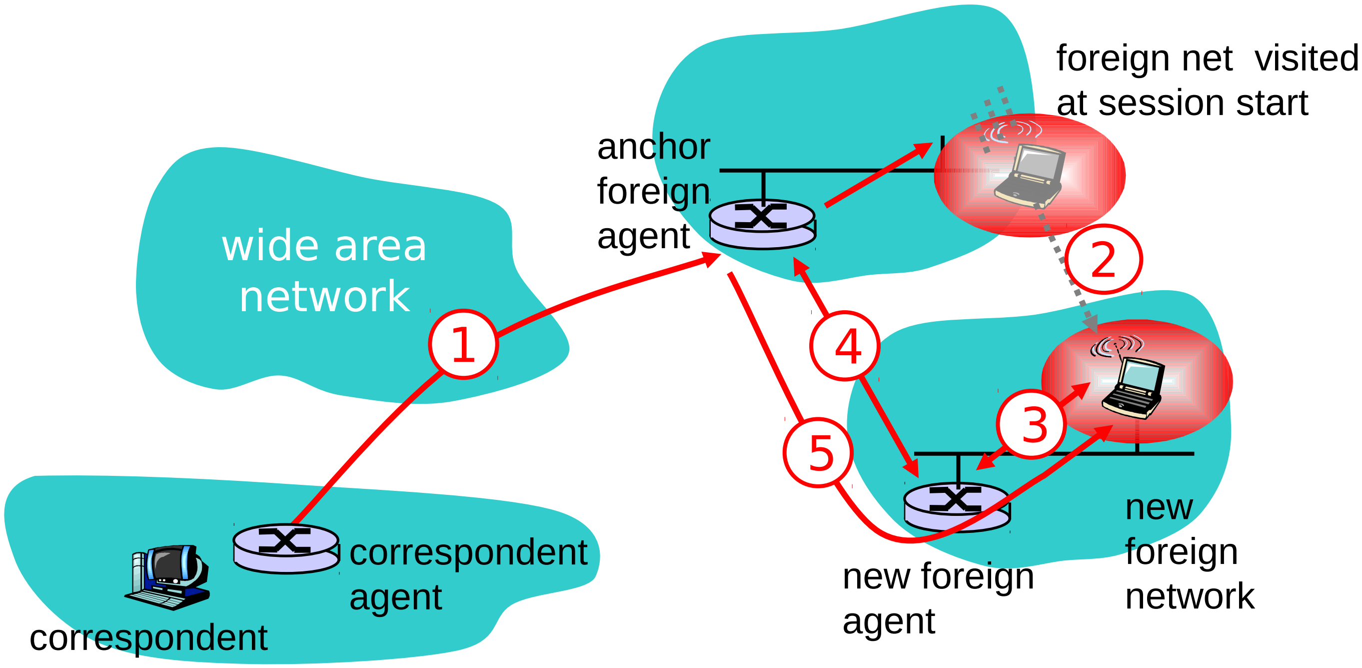


Mobile IP – Direct Routing



Mobile IP – Hierarchical Agents

- fast handover support – avoid end-to-end delay
- can be extended arbitrarily



Mobile IP – Discussion

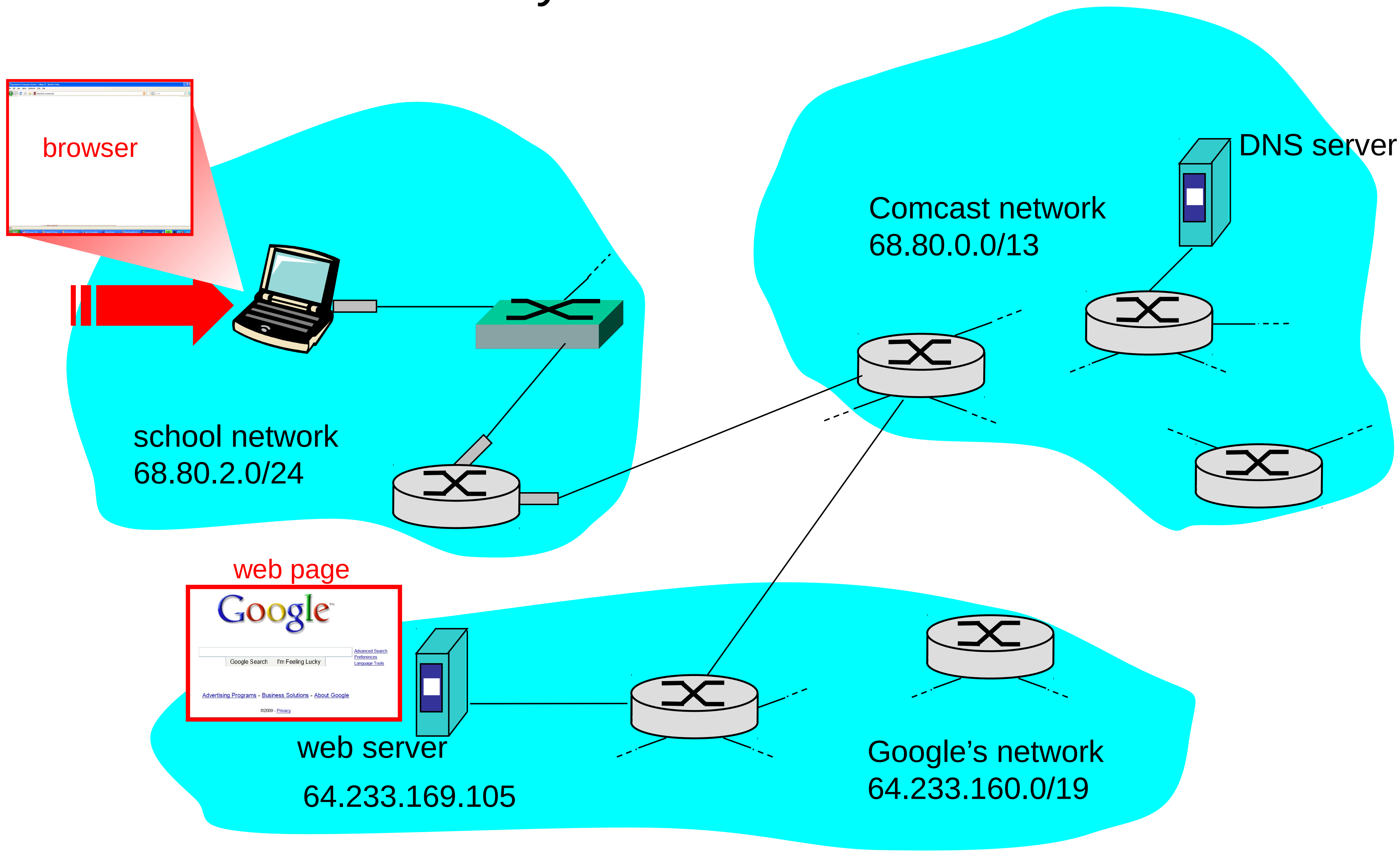
- overarching design consideration
  - transparency, backward-compatibility
  - triangle routing direct reply – source addr checks?
- foreign agent functionality can be with mobile
  - e.g., with address allocation via DHCP
- Mobile IP uses IP tunneling
  - could use direct forwarding with address rewriting
  - ... similar to reverse NAT
  - ... similar to virtual circuit -> similar to GSM

A Day in the Life of a Web Request

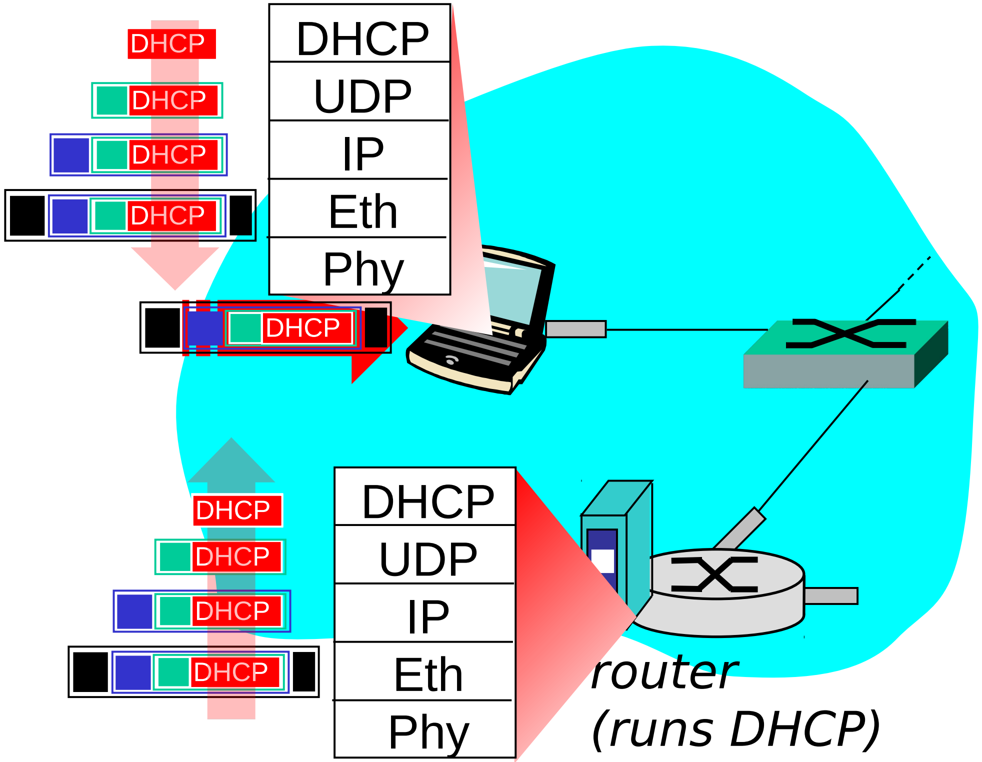
- journey through basic functionality complete!
  - channel, network, transport, naming

- putting-it-all-together: synthesis!
  - *goal:* identify, review, understand protocols (at all layers) involved in seemingly simple scenario: requesting www page
  - *scenario:* student attaches laptop to campus network, requests/receives www.google.com

A day in the life: scenario



A day in the life… connecting to the Internet

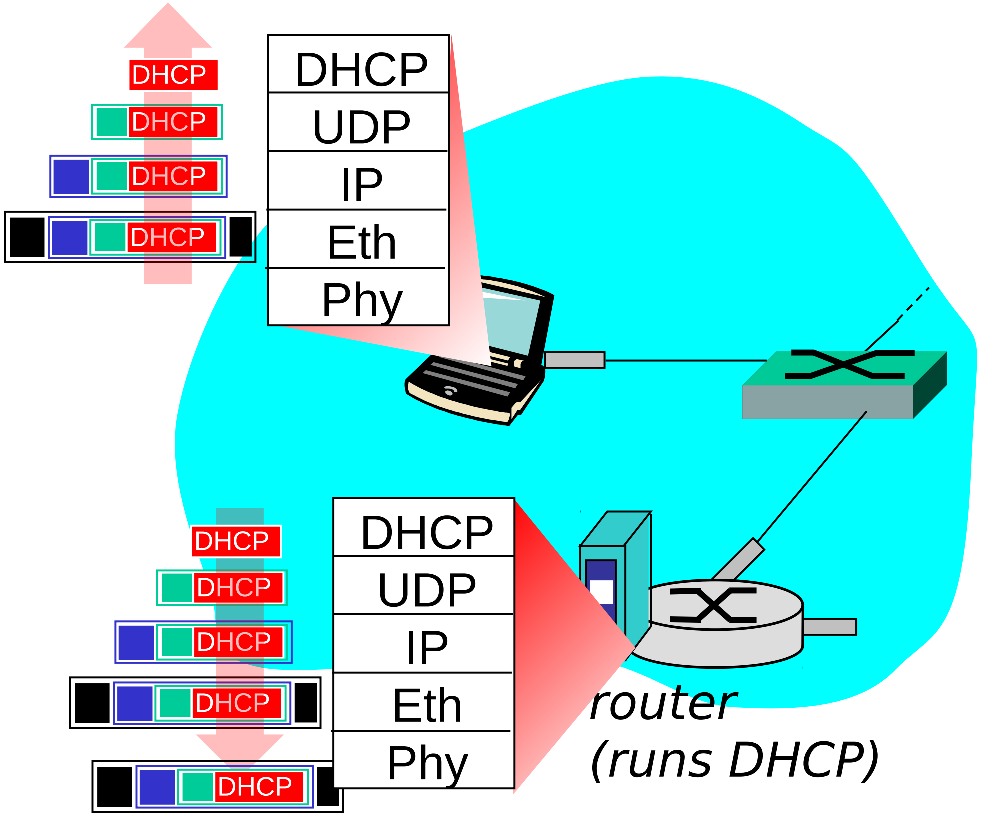


- connecting laptop needs to get its own IP address, addr of first-hop router, addr of DNS server: use ***DHCP***

- DHCP request ***encapsulated*** in ***UDP***, encapsulated in ***IP***, encapsulated in ***802.1*** Ethernet
- Ethernet frame ***broadcast*** (dest: FFFFFFFFFFFF) on LAN, received at router running ***DHCP*** server

- Ethernet ***demux'ed*** to IP demux'ed, UDP demux'ed to DHCP

A day in the life… connecting to the Internet

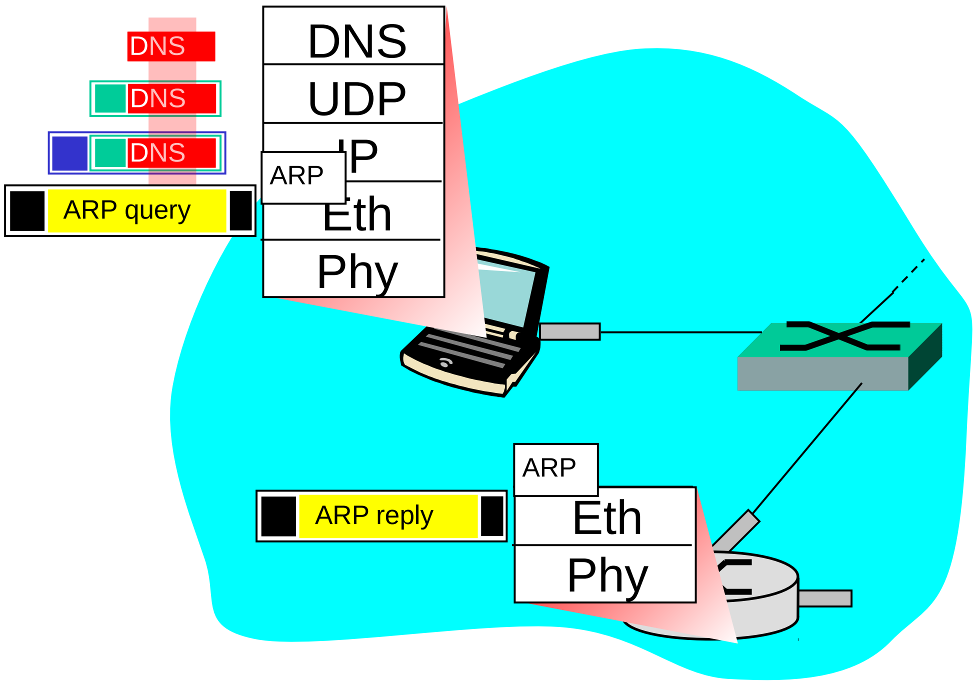


- DHCP server formulates ***DHCP ACK*** containing client's IP address, IP address of first-hop router for client, name & IP address of DNS server

- encapsulation at DHCP server, frame forwarded (***switch learning***) through LAN, demultiplexing at client
- DHCP client receives DHCP ACK reply

Client now has IP address, knows name & addr of DNS server, IP address of its first-hop router

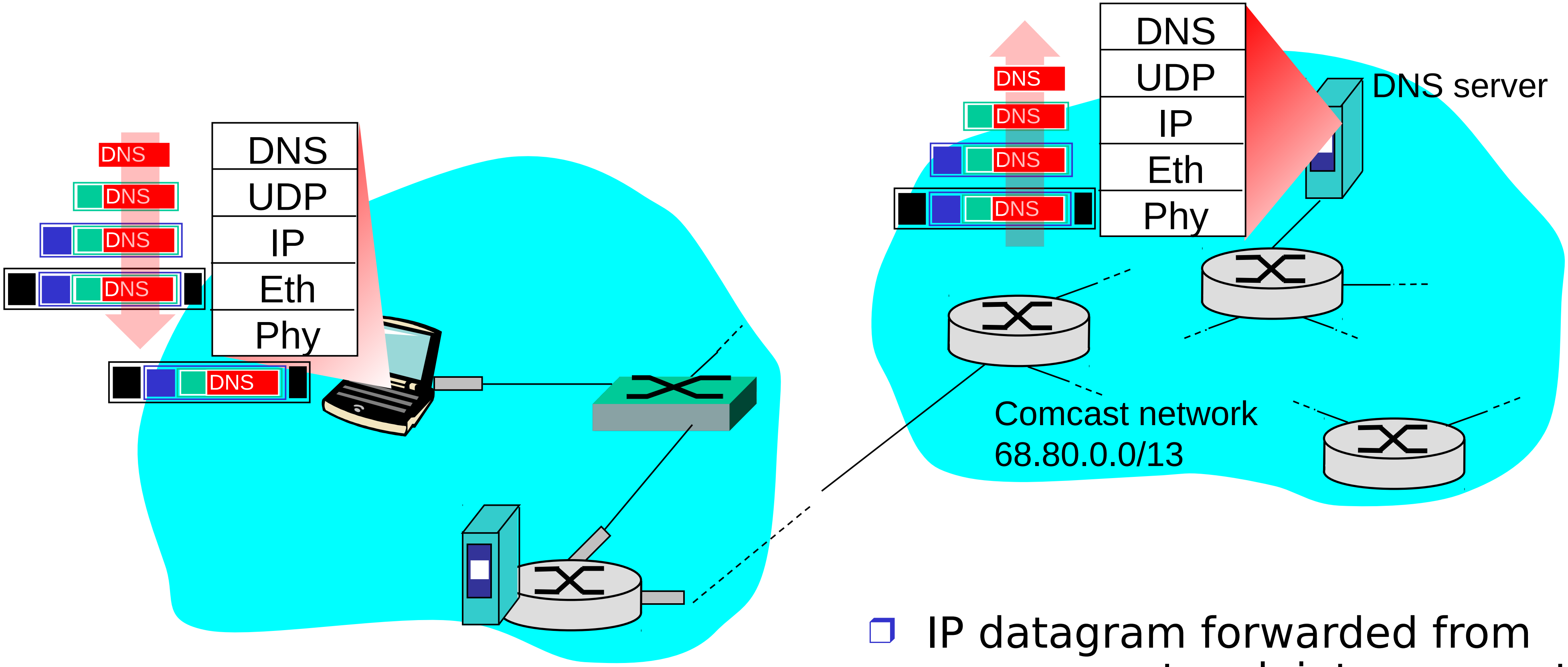
A day in the life… ARP (before DNS, before HTTP)



- before sending ***HTTP*** request, need IP address of www.google.com: ***DNS***

- DNS query created, encapsulated in UDP, encapsulated in IP, encasulated in Eth. In order to send frame to router, need MAC address of router interface: ***ARP***
- ***ARP query*** broadcast, received by router, which replies with ***ARP reply*** giving MAC address of router interface
- client now knows MAC address of first hop router, so can now send frame containing DNS query

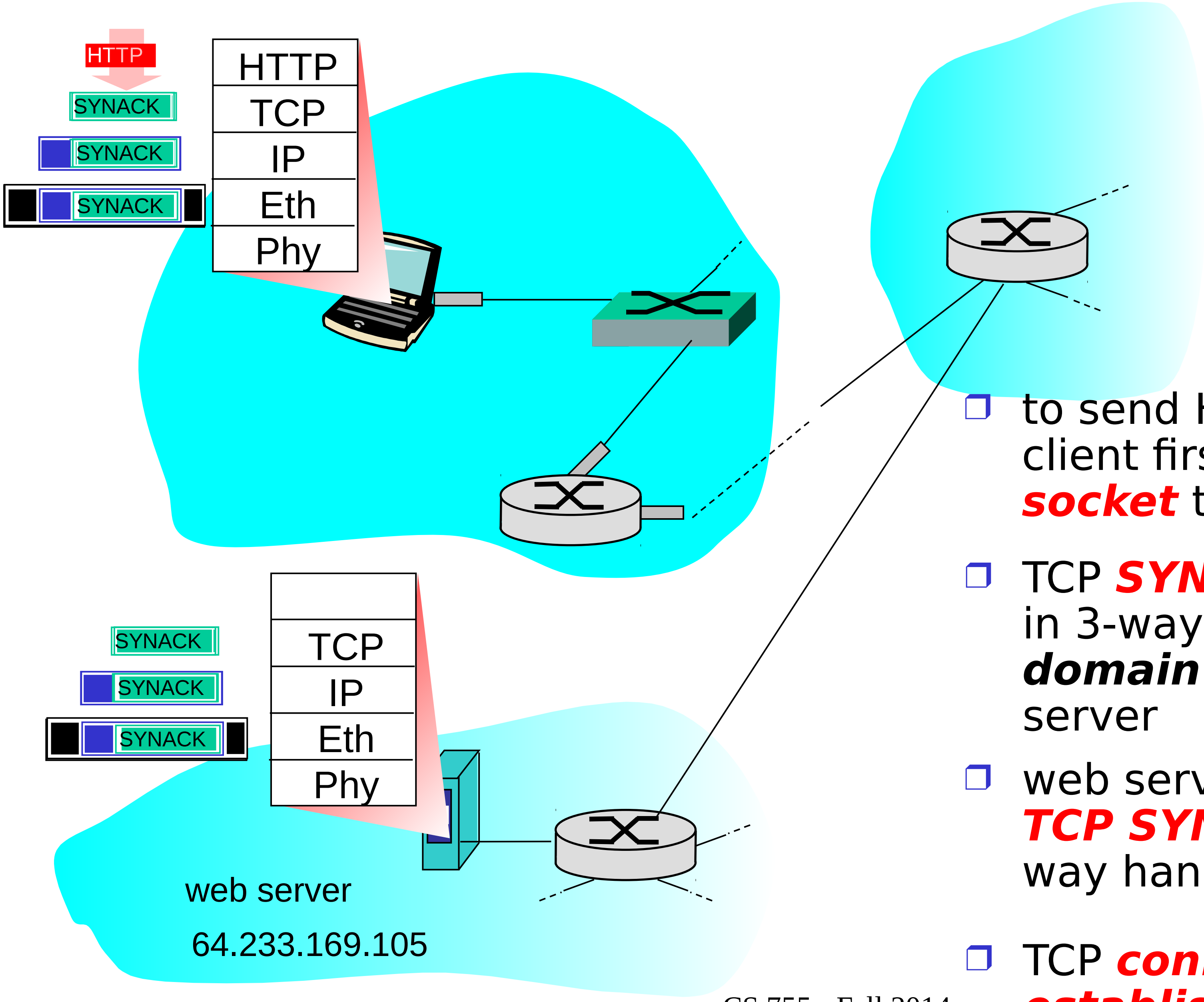
A day in the life… using DNS



- IP datagram containing DNS query forwarded via LAN switch from client to 1st hop router

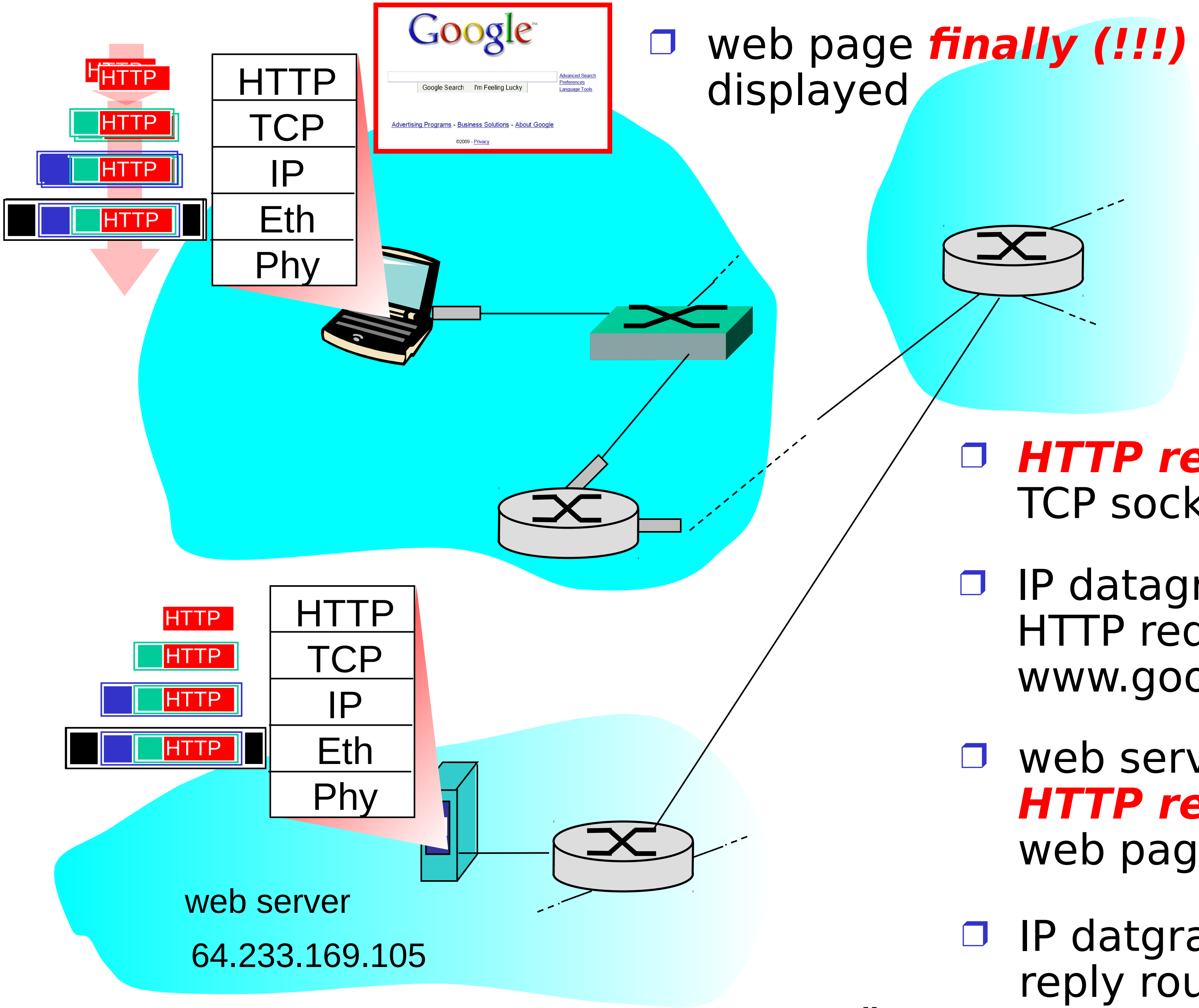
- IP datagram forwarded from campus network into comcast network, routed (tables created by ***RIP, OSPF, IS-IS*** and/or ***BGP*** routing protocols) to DNS server
- demux'ed to DNS server
- DNS server replies to client with IP address of www.google.com

A day in the life… TCP connection carrying HTTP



- to send HTTP request, client first opens ***TCP socket*** to web server
- TCP ***SYN segment*** (step 1 in 3-way handshake) ***inter-domain routed*** to web server
- web server responds with ***TCP SYNACK*** (step 2 in 3-way handshake)
- TCP ***connection established!***

A day in the life… HTTP request/reply



- web page ***finally (!!!)*** displayed
- ***HTTP request*** sent into TCP socket
- IP datagram containing HTTP request routed to www.google.com
- web server responds with ***HTTP reply*** (containing web page)
- IP datgram containing HTTP reply routed back to client