University of Duisburg-Essen
Networked Embedded Systems Group
Institute for Computer Science and
Business Information Systems (ICB)
Schützenbahn 70
D-45127 Essen, Germany

Kommunikationsnetze 2
7 – Infrastructure

Prof. Dr. Pedro José Marrón

IP Addresses: How to Get One?

- How does a host get an IP address?

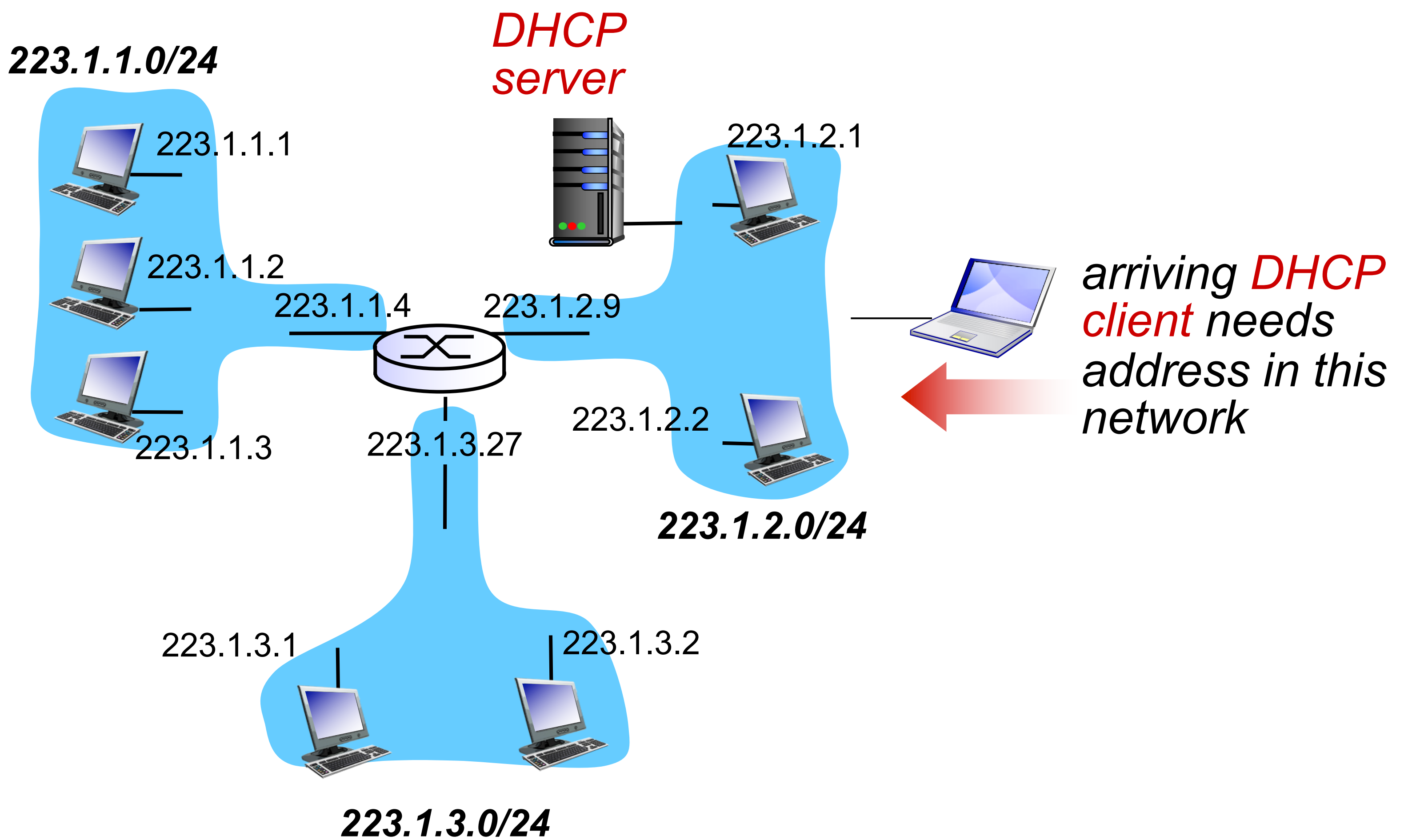
- Hard-coded by system admin in a file

- DHCP: Dynamic Host Configuration Protocol
  - Dynamically get address from a server
    - “plug-and-play”

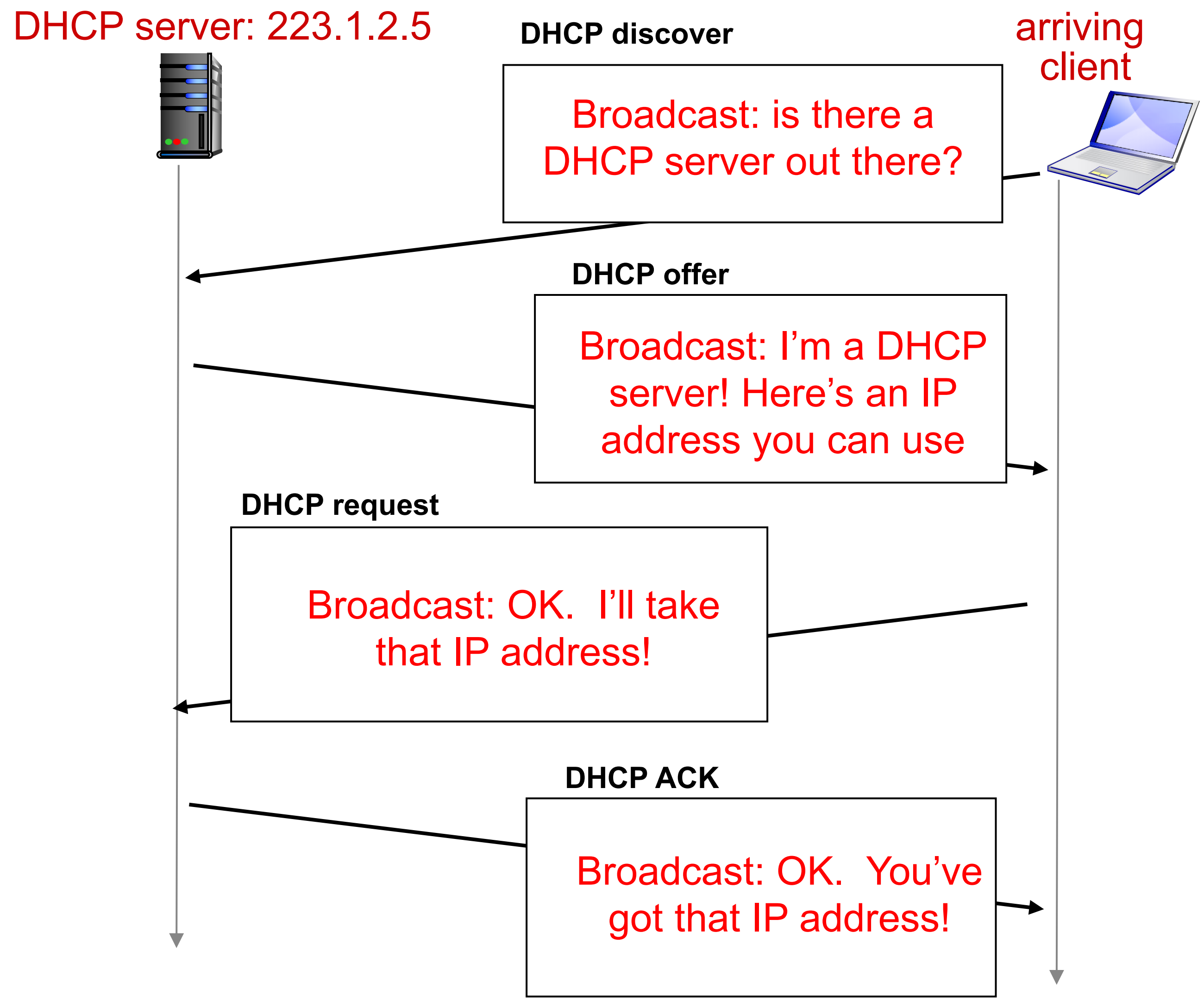
DHCP: Dynamic Host Configuration Protocol

- Goal: allow host to dynamically obtain its IP address from network server when it joins network
  - Can renew its lease on address in use
  - Allows reuse of addresses (only hold address while connected/“on”)
  - Support for mobile users who want to join network (more shortly)
- DHCP overview
  - Host broadcasts “DHCP discover” msg [optional]
  - DHCP server responds with “DHCP offer” msg [optional]
  - Host requests IP address: “DHCP request” msg
  - DHCP server sends address: “DHCP ack” msg

DHCP Client-Server Scenario



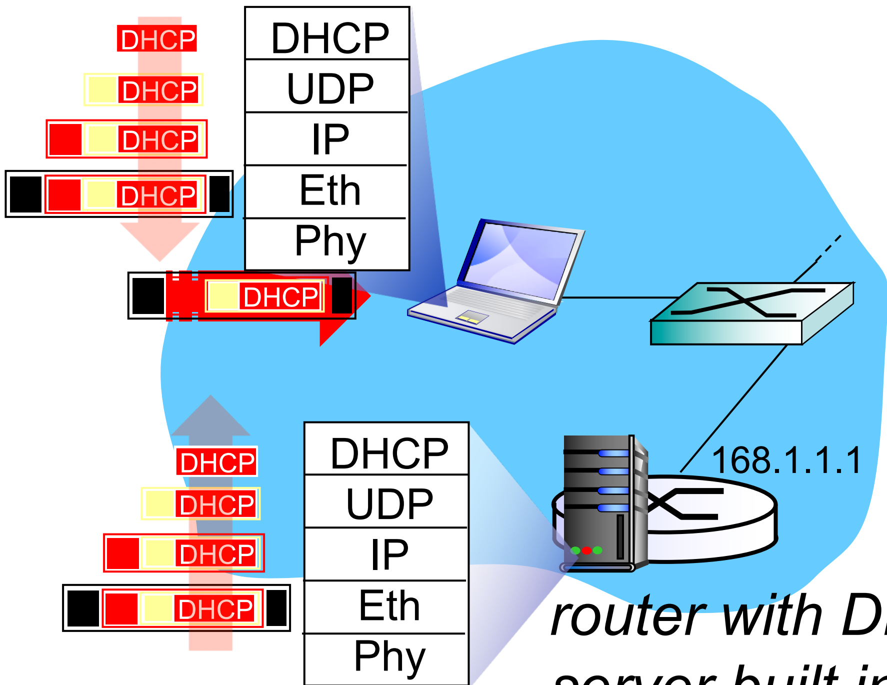
DHCP Client-Server Scenario



DHCP: More than IP Addresses

- DHCP can return more than just allocated IP address on subnet
  - Address of first-hop router for client
  - Name and IP address of DNS server
  - Network mask (indicating network versus host portion of address)

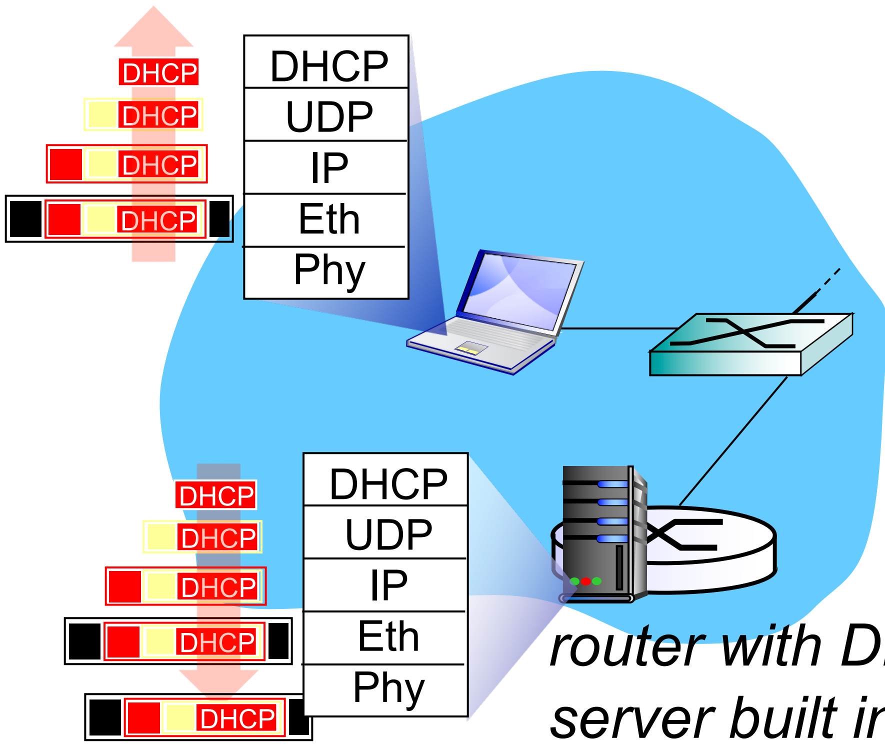
DHCP: Example



router with DHCP server built into router

- Connecting laptop needs its 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 demuxed to IP, demuxed to UDP, demuxed to DHCP

DHCP: Example



router with DHCP server built into router

- 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 of DHCP server, frame forwarded to client, demuxing up to DHCP at client
- Client now knows its IP address, name and IP address of DNS server, IP address of its first-hop router

DHCP: Wireshark Output (Home LAN)

request

Message type: **Boot Request (1)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 0.0.0.0 (0.0.0.0)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 0.0.0.0 (0.0.0.0)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) **DHCP Message Type = DHCP Request**
Option: (61) Client identifier
    Length: 7; Value: 010016D323688A;
    Hardware type: Ethernet
    Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Option: (t=50,l=4) Requested IP Address = 192.168.1.101
Option: (t=12,l=5) Host Name = "nomad"
Option: (55) Parameter Request List
    Length: 11; Value: 010F03062C2E2F1F21F92B
    1 = Subnet Mask; 15 = Domain Name
    3 = Router; 6 = Domain Name Server
    44 = NetBIOS over TCP/IP Name Server
    ……

reply

Message type: **Boot Reply (2)**
Hardware type: Ethernet
Hardware address length: 6
Hops: 0
Transaction ID: 0x6b3a11b7
Seconds elapsed: 0
Bootp flags: 0x0000 (Unicast)
Client IP address: 192.168.1.101 (192.168.1.101)
Your (client) IP address: 0.0.0.0 (0.0.0.0)
Next server IP address: 192.168.1.1 (192.168.1.1)
Relay agent IP address: 0.0.0.0 (0.0.0.0)
Client MAC address: Wistron_23:68:8a (00:16:d3:23:68:8a)
Server host name not given
Boot file name not given
Magic cookie: (OK)
Option: (t=53,l=1) DHCP Message Type = DHCP ACK
Option: (t=54,l=4) Server Identifier = 192.168.1.1
Option: (t=1,l=4) Subnet Mask = 255.255.255.0
Option: (t=3,l=4) Router = 192.168.1.1
Option: (6) Domain Name Server
    Length: 12; Value: 445747E2445749F244574092;
    IP Address: 68.87.71.226;
    IP Address: 68.87.73.242;
    IP Address: 68.87.64.146
Option: (t=15,l=20) Domain Name = "hsd1.ma.comcast.net."

IP Addresses: How to Get One?

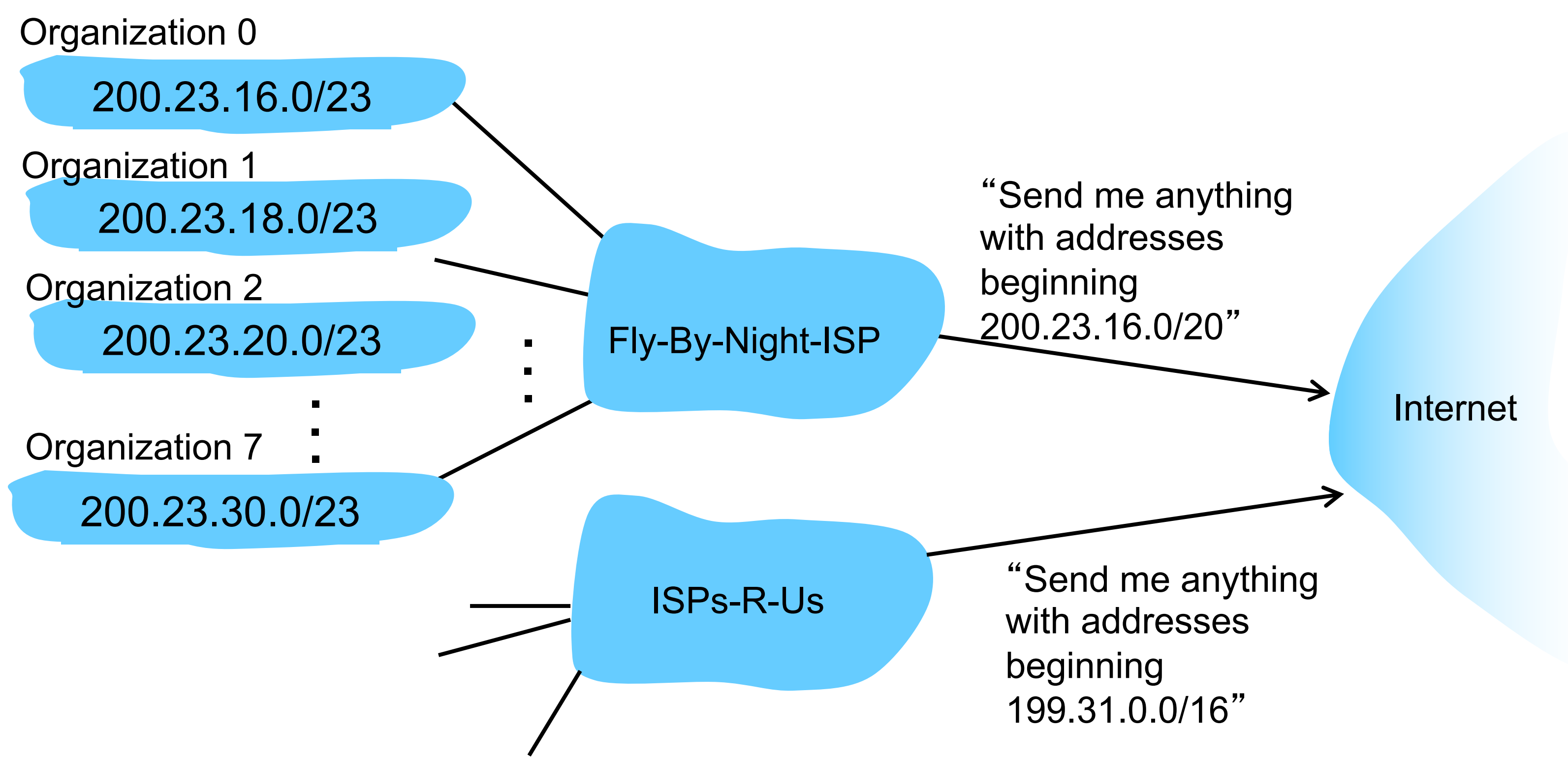
- How does network get subnet part of IP addr?

- Gets allocated portion of its provider ISP's address space

ISP's block	11001000 00010111 00010000 00000000	200.23.16.0/20
Organization 0	11001000 00010111 00010000 00000000	200.23.16.0/23
Organization 1	11001000 00010111 00010010 00000000	200.23.18.0/23
Organization 2	11001000 00010111 00010100 00000000	200.23.20.0/23
...	….. ….	….
Organization 7	11001000 00010111 00011110 00000000	200.23.30.0/23

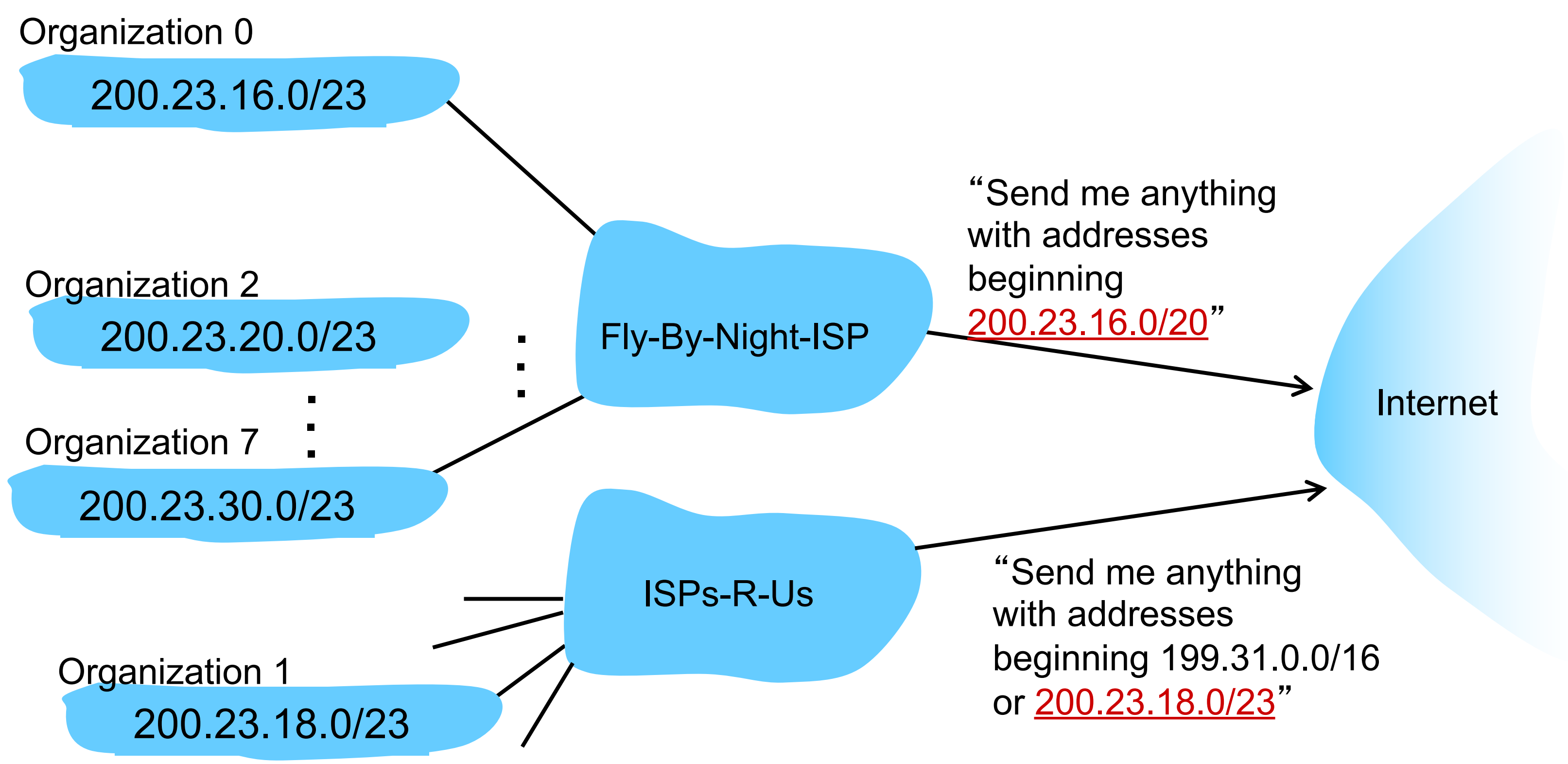
Hierarchical Addressing: Route Aggregation

- Hierarchical addressing allows efficient advertisement of routing information



Hierarchical Addressing: More Specific Routes

- ISPs-R-Us has more specific route to Organization 1

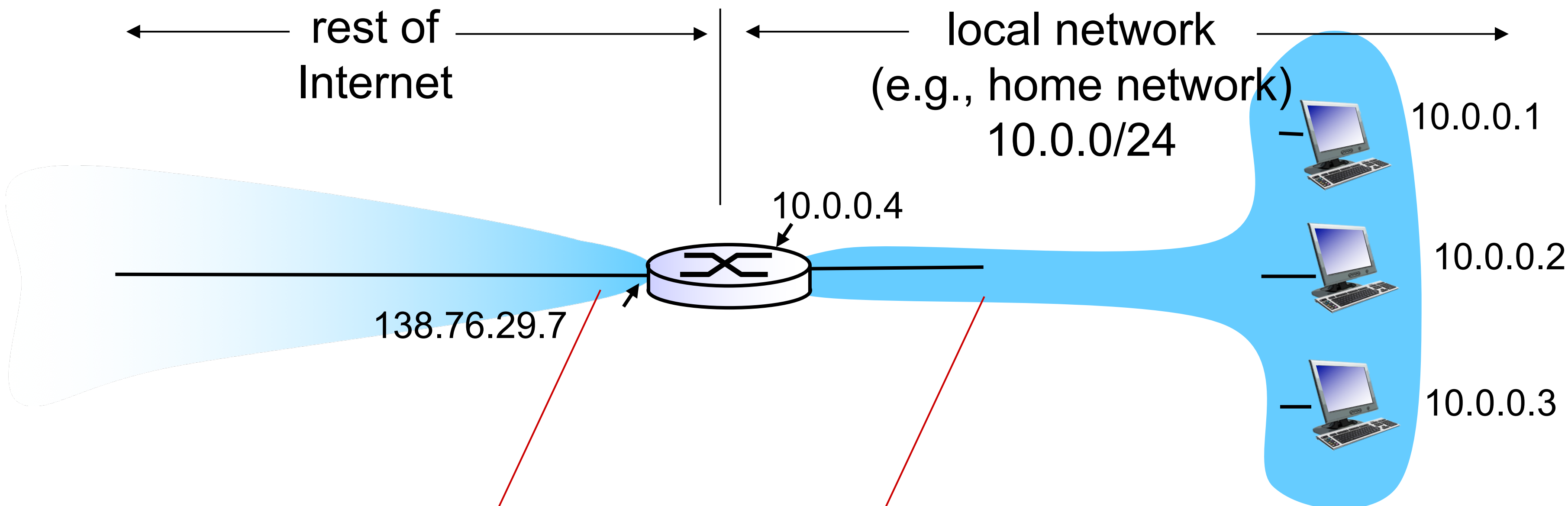


IP Addressing: The Last Word…

- How does an ISP get block of addresses?

- ICANN: Internet Corporation for Assigned Names and Numbers
  - http://www.icann.org
  - Allocates addresses
  - Manages DNS
  - Assigns domain names, resolves disputes

NAT/PAT: Network Address Translation



all datagrams *leaving* local network have *same* single source NAT IP address: 138.76.29.7,different source port numbers

datagrams with source or destination in this network have 10.0.0/24 address for source, destination (as usual)

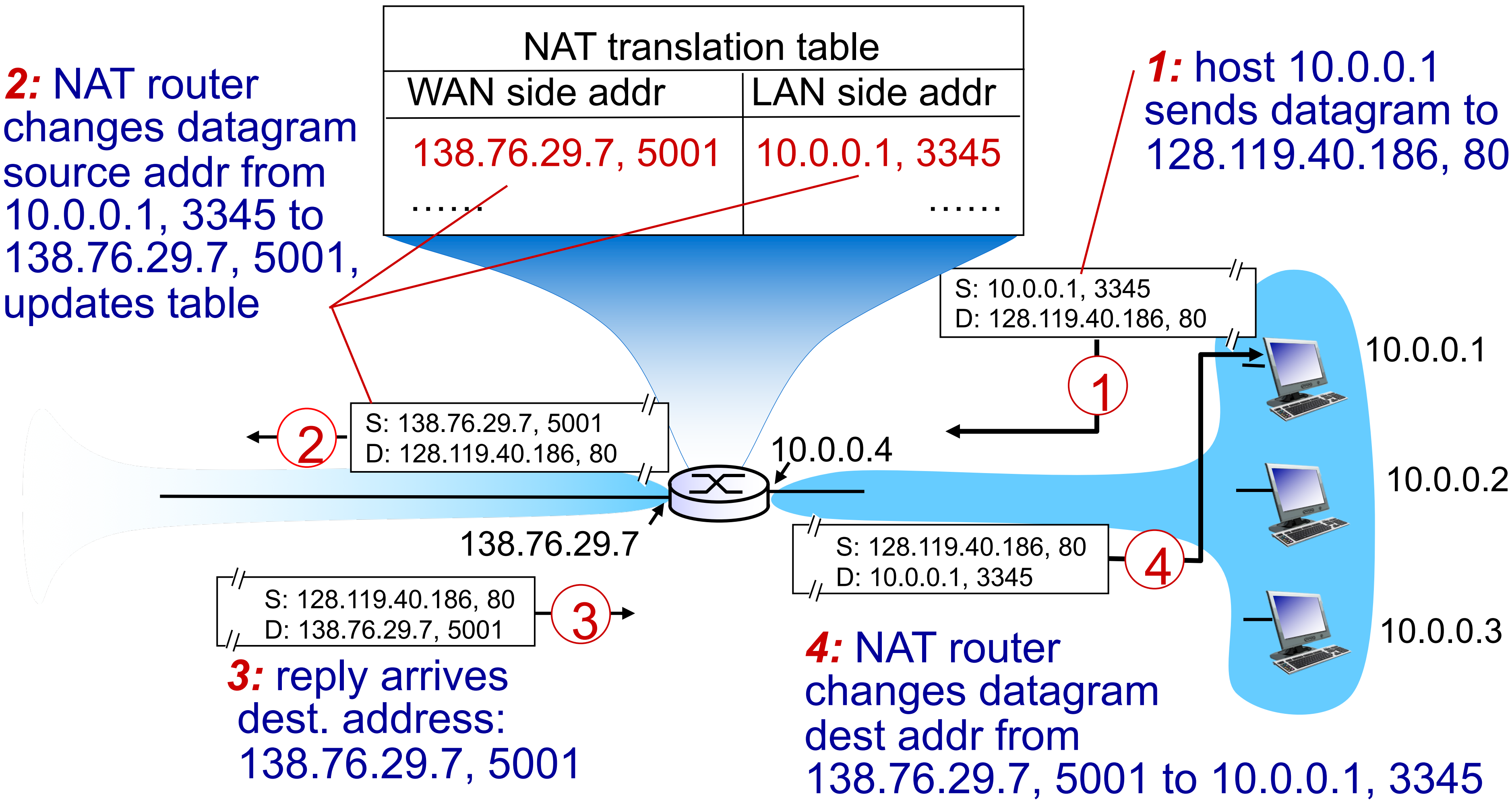
NAT/PAT: Network Address Translation

- Motivation: Local network uses just one IP address as far as outside world is concerned
  - Range of addresses not needed from ISP: just one IP address for all devices
  - Can change addresses of devices in local network without notifying outside world
  - Can change ISP without changing addresses of devices in local network
  - Devices inside local net not explicitly addressable, visible by outside world (a security bonus)

NAT/PAT: Implementation

- Implementation: NAT router must
  - Replace (source IP address, port #) of every outgoing datagram to (NAT IP address, new port #)
    - Remote clients/servers will respond using (NAT IP address, new port #) as destination addr
  - Remember (in NAT translation table) every (source IP address, port #) to (NAT IP address, new port #) translation pair
  - Replace (NAT IP address, new port #) in dest fields of every incoming datagram with corresponding (source IP address, port #) stored in NAT table

NAT/PAT: Example



NAT/PAT: Network Address Translation

- 16-bit port-number field
  - 60.000 simultaneous connections with a single LAN-side address
- NAT is controversial
  - Routers should only process up to layer 3
  - Address shortage should be solved by IPv6
  - Violates end-to-end argument
    - NAT possibility must be taken into account by app designers, e.g., P2P applications
  - NAT traversal: what if client wants to connect to server behind NAT?

NAT: Types

- PAT (or NAT Overload)
  - Mapping between multiple local IP addresses to a single public IP address (many-to-one) using different Layer 4 ports
  - (What we have seen until now)
- Static NAT
  - One-to-one mapping between a local IP address and a global IP address
  - Static configuration allowing services to be accessed from outside
- Dynamic NAT
  - A pool of public IP addresses is available
  - At run-time, local IP addresses are mapped to one of the free public IP addresses in the pool
  - After a timeout, the mapping is purged from the translation table

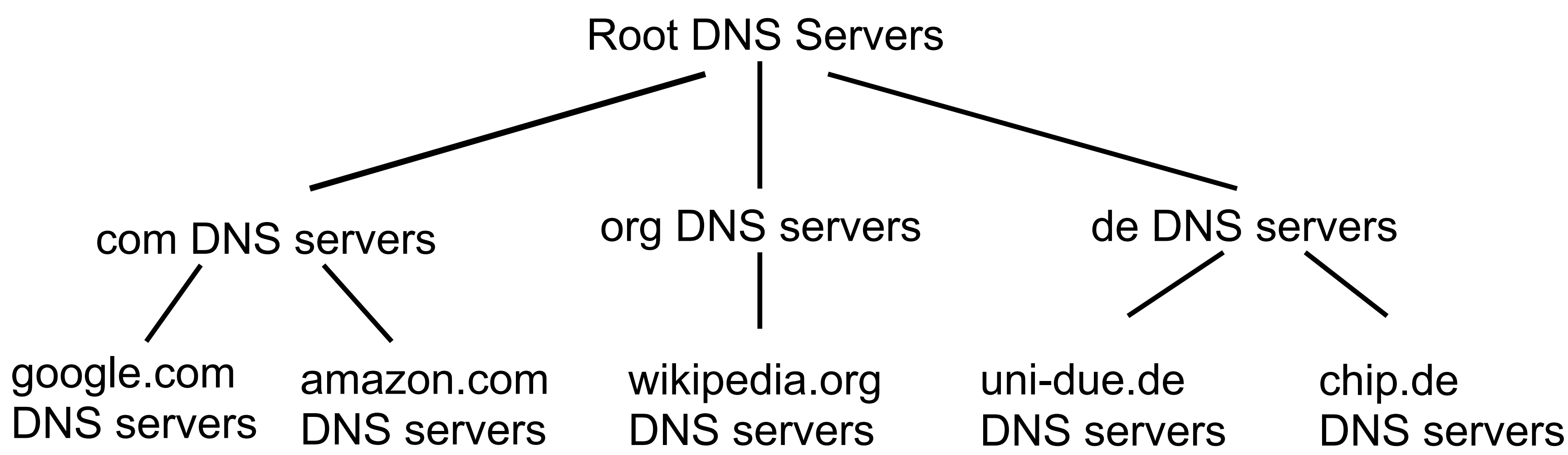
DNS: Domain Name System

- Internet hosts, routers identifiers:
  - IP address – used for addressing datagrams
  - "name", e.g., uni-due.de – used by humans
- Hot to map between IP address and name, and vice versa?
- Domain Name System
  - Distributed database implemented in hierarchy of many name servers
  - Application-layer protocol: hosts, name servers communicate to resolve names (address/name translation)
    - Note: core Internet function, implemented as application-layer protocol

DNS: Services, Structure

- DNS services
  - Hostname to IP address translation
  - Host aliasing
    - Canonical, alias names
  - Mail server aliasing
  - Load distribution
    - Replicated Web servers
    - Many IP addresses correspond to one name
- Why not centralize DNS?
  - Single point of failure
  - Traffic volume
  - Distant centralized database
  - Maintenance
  - Does not scale!

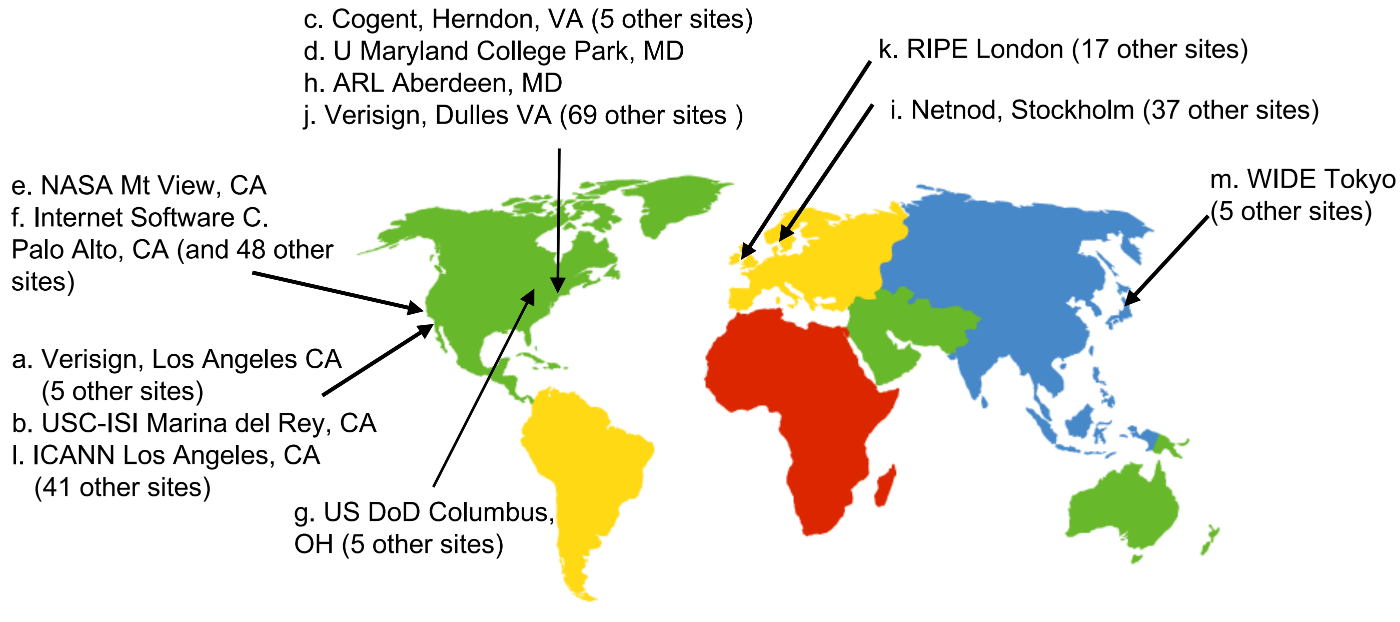
DNS: A Distributed, Hierarchical Database



- Client wants IP for nes.uni-due.de; 1st approximation:
  - Client queries root server to find de DNS server
  - Client queries de DNS server to get uni-due.de DNS server
  - Client queries uni-due.de DNS server to get IP address of nes.uni-due.de

DNS: Root Name Servers

- Contacted by local name server that can not resolve name
- Root name server:
  - Contacts authoritative name server if name mapping not known
  - Gets mapping
  - Returns mapping to local name server
- 13 logical root name “servers” worldwide
  - Each server replicated multiple times



Top-Level Domain and Authoritative Servers

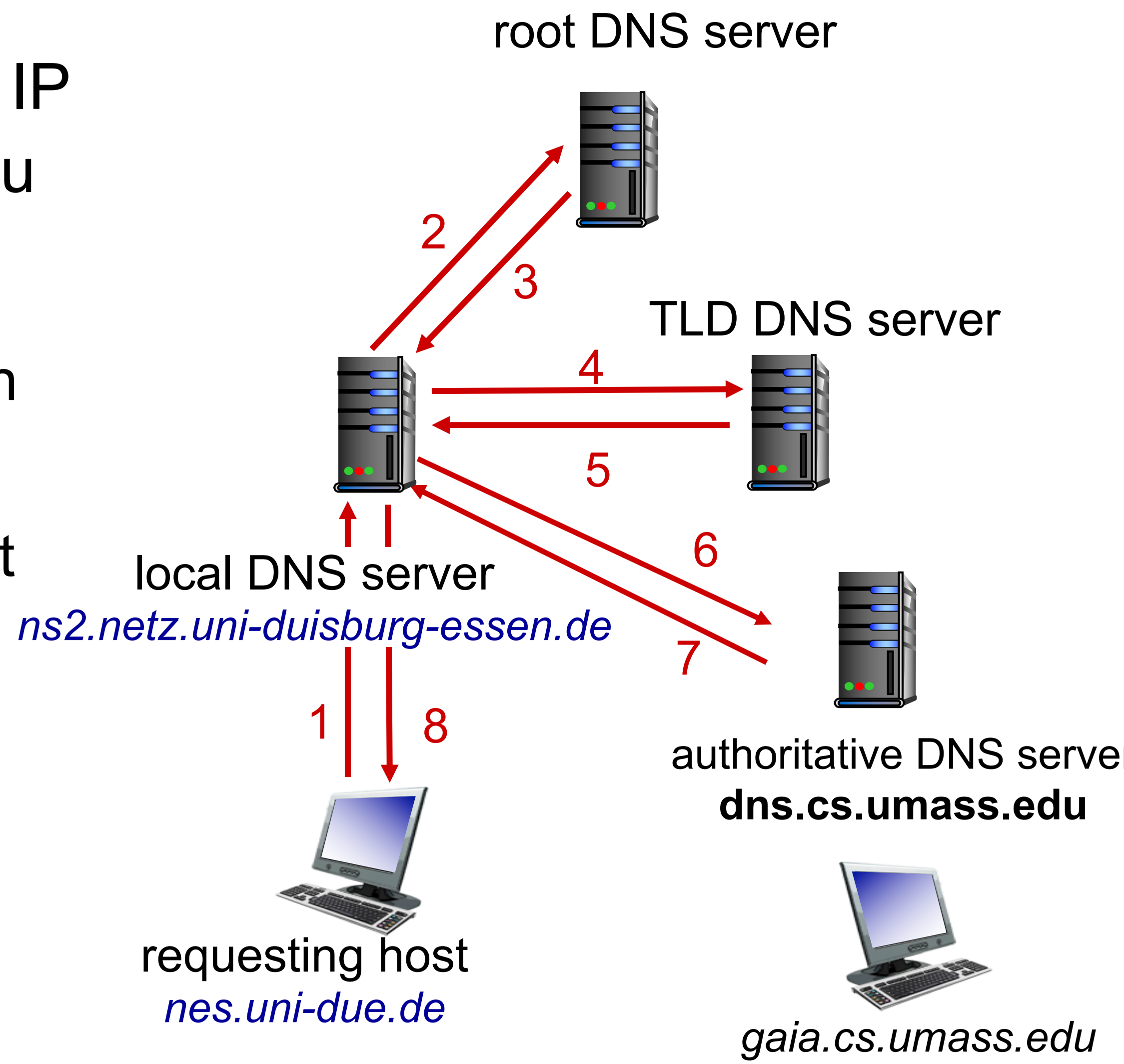
- Top-level domain (TLD) servers
  - Responsible for com, org, net, edu, aero, jobs, museums, and all top-level country domains, e.g., uk, de, fr, ca, it, jp
  - Verisign (after Network Solutions) maintains servers for .com TLD
  - DENIC eG for .de TLD
- Authoritative DNS servers
  - Organization’s own DNS server(s) providing authoritative hostname to IP mappings for organization’s named hosts
  - Can be maintained by organization or service provider

Local DNS Name Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one
  - Also called “default name server”
- When host makes DNS query, query is sent to its local DNS server
  - Has local cache of recent name-to-address translation pairs (but may be out-of-date)
  - Acts as proxy, forwards query into hierarchy

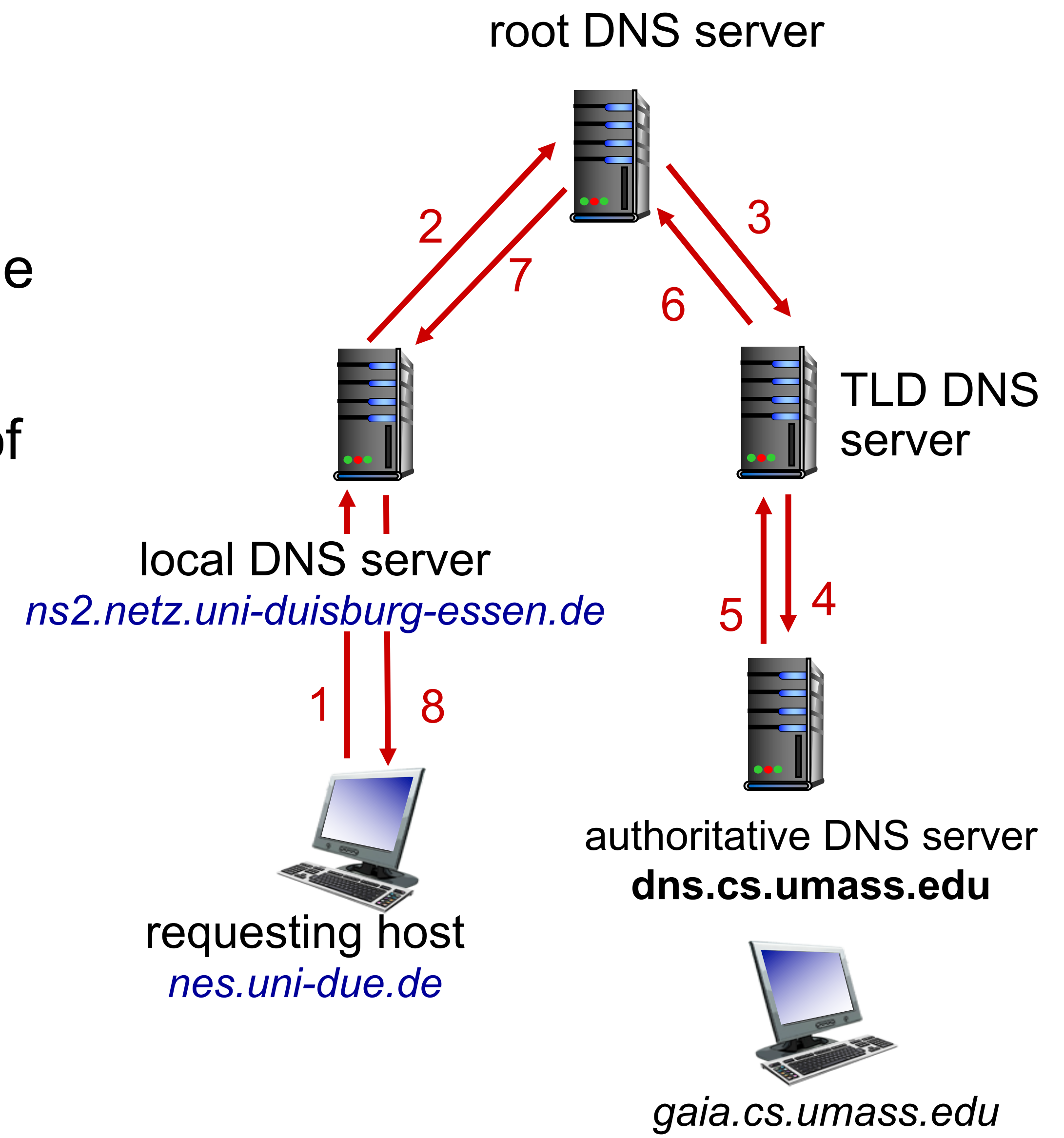
DNS Name Resolution Example

- Host at nes.uni-due.de wants IP address for gaia.cs.umass.edu
- Iterated query:
  - Contacted server replies with name of server to contact
  - “I do not know this name, but ask this server”



DNS Name Resolution Example

- Recursive query:
  - Puts burden of name resolution on contacted name server
  - Heavy load at upper levels of hierarchy?



DNS Lookup Trace Example

dig +nocmd +additional +trace +nodnssec www.nes.uni-due.de A

```
.                          364237   IN   NS   e.root-servers.net.
.                          364237   IN   NS   a.root-servers.net.
.                          364237   IN   NS   d.root-servers.net.
.                          364237   IN   NS   c.root-servers.net.
.                          364237   IN   NS   b.root-servers.net.
.                          364237   IN   NS   l.root-servers.net.
.                          364237   IN   NS   f.root-servers.net.
.                          364237   IN   NS   i.root-servers.net.
.                          364237   IN   NS   g.root-servers.net.
.                          364237   IN   NS   m.root-servers.net.
.                          364237   IN   NS   j.root-servers.net.
.                          364237   IN   NS   h.root-servers.net.
.                          364237   IN   NS   k.root-servers.net.
;; Received 239 bytes from 134.91.76.7#53(134.91.76.7) in 1 ms

de.                        172800   IN   NS   a.nic.de.
de.                        172800   IN   NS   f.nic.de.
de.                        172800   IN   NS   l.de.net.
de.                        172800   IN   NS   n.de.net.
de.                        172800   IN   NS   s.de.net.
de.                        172800   IN   NS   z.nic.de.
;; Received 389 bytes from 192.58.128.30#53(j.root-servers.net) in 15 ms

uni-due.de.          86400   IN   NS   dns-2.dfn.de.
uni-due.de.          86400   IN   NS   ns1.uni-duisburg-essen.de.
uni-due.de.          86400   IN   NS   ns2.uni-duisburg-essen.de.
;; Received 258 bytes from 194.246.96.1#53(z.nic.de) in 6 ms

nes.uni-due.de.      86400   IN   NS   ns1.netz.uni-duisburg-essen.de.
nes.uni-due.de.      86400   IN   NS   ns2.netz.uni-duisburg-essen.de.
;; Received 107 bytes from 193.174.75.54#53(dns-2.dfn.de) in 14 ms

www.nes.uni-due.de.  86400   IN   A    134.91.76.7
;; Received 63 bytes from 132.252.1.7#53(ns2.netz.uni-duisburg-essen.de) in 1 ms
```

DNS: Caching, Updating Records

- Once (any) name server learns mapping, it caches mapping
  - Cache entries timeout (disappear) after some time (TTL)
  - TLD servers typically cached in local name servers
    - Thus root name server not often visited
- Cached entries may be out-of-date (best effort name-to-address translation!)
  - If name host changes IP address, may not be known Internet-wide until all TTLs expire
- Update/notify mechanisms proposed IETF standard
  - RFC 2136

DNS Records

- DNS: distributed database storing resource records (RR)

  RR format: `(name, ttl, class, type, value)`

- Type = A
  - `name` is hostname
  - `value` is IP address
- Type = NS
  - `name` is domain
  - `value` is hostname of authoritative name server for this domain
- Type = CNAME
  - `name` is alias name for some “canonical” (the real) name
  - `value` is canonical name
- Type = MX
  - `value` is name of mailserver associated with `name`

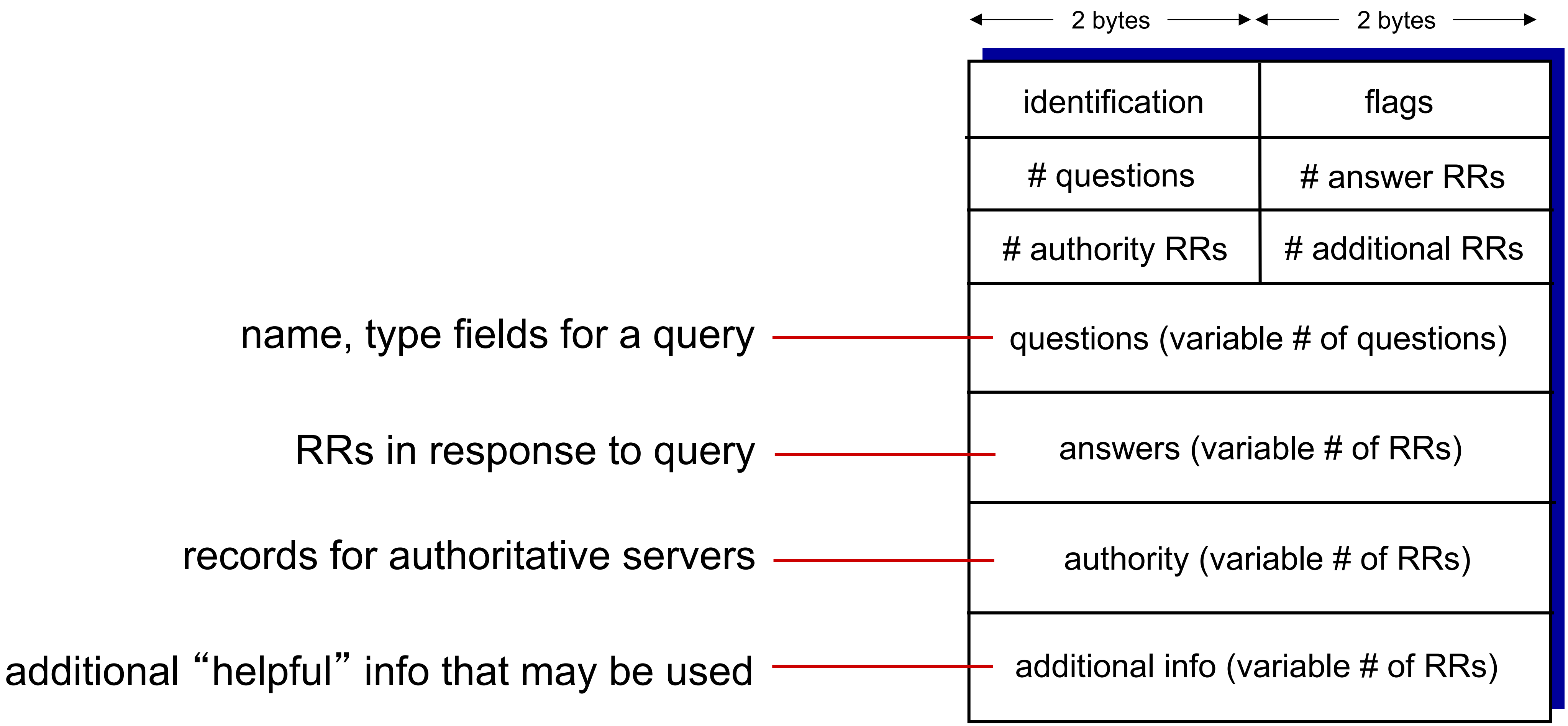
DNS Protocol, Messages

- Query and reply messages, both with same message format

- Message header
  - Identification: 16 bit # for query
    - Reply uses the same #
  - Flags:
    - Query or reply
    - Recursion desired
    - Recursion available
    - Reply is authoritative

← 2 bytes →	← 2 bytes →
identification	flags
# questions	# answer RRs
# authority RRs	# additional RRs
questions (variable # of questions)	
answers (variable # of RRs)	
authority (variable # of RRs)	
additional info (variable # of RRs)	

DNS Protocol, Messages



Attacking DNS

- DDoS attacks
  - Target root servers with traffic
    - Not successful to date
    - Traffic filtering
    - Local DNS servers cache IPs of TLD servers, allowing root server bypass
  - Target TLD servers
    - Potentially more dangerous
- Redirect attacks
  - Main-in-the-middle
    - Intercept queries
  - DNS poisoning
    - Send bogus replies to DNS server, which caches
- Exploit DNS for DDoS
  - Send queries with spoofed source address: target IP
  - Requires amplification