Network programming, DNS, and NAT

Copyright © University of Illinois CS 241 Staff

Today

- Network programming tips
- Domain name system
- Network Address Translation
- Bonus slides (for your reference)
  - Timers with select()
  - select() vs. poll()

Copyright © University of Illinois CS 241 Staff

Tip #1: Can't bind?

- Problem: How come I get "address already in use" from `bind()`?
  - You have stopped your server, and then re-started it right away
  - The sockets that were used by the first incarnation of the server are still active

Copyright ©: University of Illinois CS 241 Staff

setsockopt

```
int yes = 1;
setsockopt (fd, SOL_SOCKET,
   SO_REUSEADDR, (char *) &yes, sizeof
   (yes));
```

- Call just before `bind()`
- Allows bind to succeed despite the existence of existing connections in the requested TCP port
- Connections in limbo (e.g. lost final ACK) will cause bind to fail

Copyright ©: University of Illinois CS 241 Staff

Tip #2: Dealing with abruptly closed connection

[demo: server.c]

signal

- Problem: Socket at other end is closed
  - Write to your end generates **SIGPIPE**
  - This signal kills the program by default!

```
signal (SIGPIPE, SIG_IGN);
```

  - Call at start of main in server
  - Allows you to ignore broken pipe signals
  - Can ignore or install a proper signal handler
  - Default handler exits (terminates process)

Copyright ©: University of Illinois CS 241 Staff

Tip #3: Beej's guide

- Beej's Guide to Network Programming

http://beej.us/guide/bgnet/

Copyright © University of Illinois CS 241 Staff

The Domain Name System

Slides thanks in part to Jennifer Rexford,
Ion Stoica, Vern Paxson, and Scott Shenker

Host Names vs. IP addresses

- Host names
  - Mnemonic name appreciated by humans
  - Variable length, full alphabet of characters
  - Provide little (if any) information about physical location
  - Examples: www.cnn.com and bbc.co.uk

- IP addresses
  - Numerical address appreciated by routers
  - Fixed length, binary number
  - Hierarchical, related to host location
  - Examples: 64.236.16.20 and 212.58.224.131

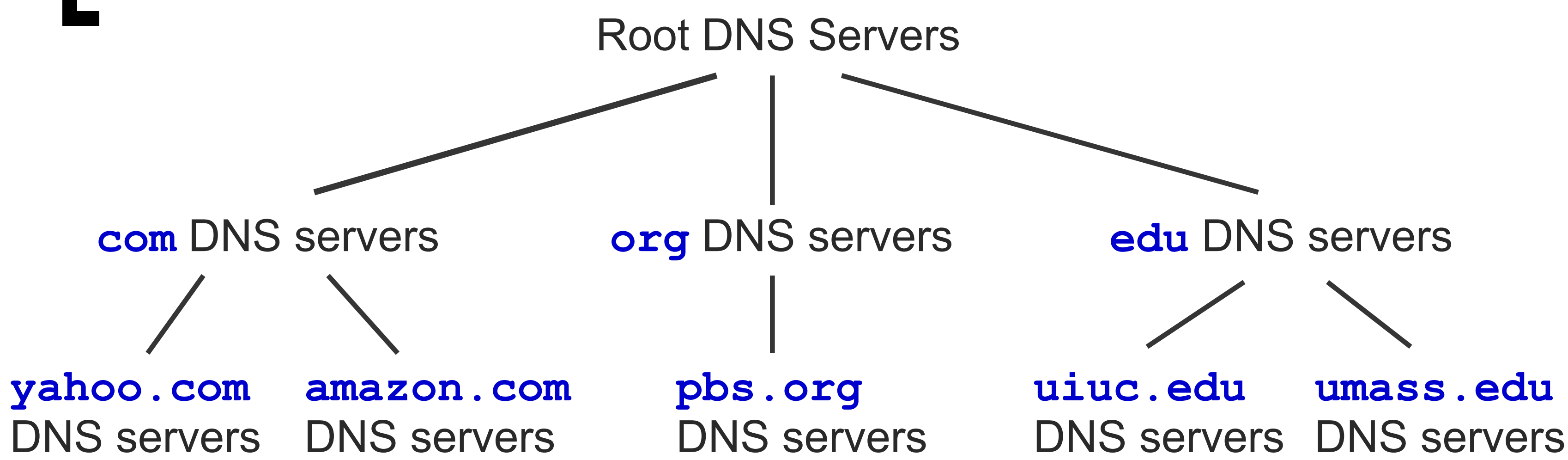
Separating Naming and Addressing

- Names are easier to remember
  - cnn.com vs. 64.236.16.20 *(but not shortened urls)*
- Addresses can change underneath
  - Move www.cnn.com to 4.125.91.21
  - E.g., renumbering when changing providers
- Name could map to multiple IP addresses
  - www.cnn.com to multiple (8) replicas of the Web site
  - Enables
    - Load-balancing
    - Reducing latency by picking nearby servers
    - Tailoring content based on requester's location/identity
- Multiple names for the same address
  - E.g., aliases like www.cnn.com and cnn.com

Domain Name System (DNS)

- Properties of DNS
  - Hierarchical name space divided into zones
  - Zones distributed over collection of DNS servers
- Hierarchy of DNS servers
  - Root (hardwired into other servers)
  - Top-level domain (TLD) servers
  - Authoritative DNS servers
- Performing the translations
  - Local DNS servers
  - Resolver software

Distributed, Hierarchical Database

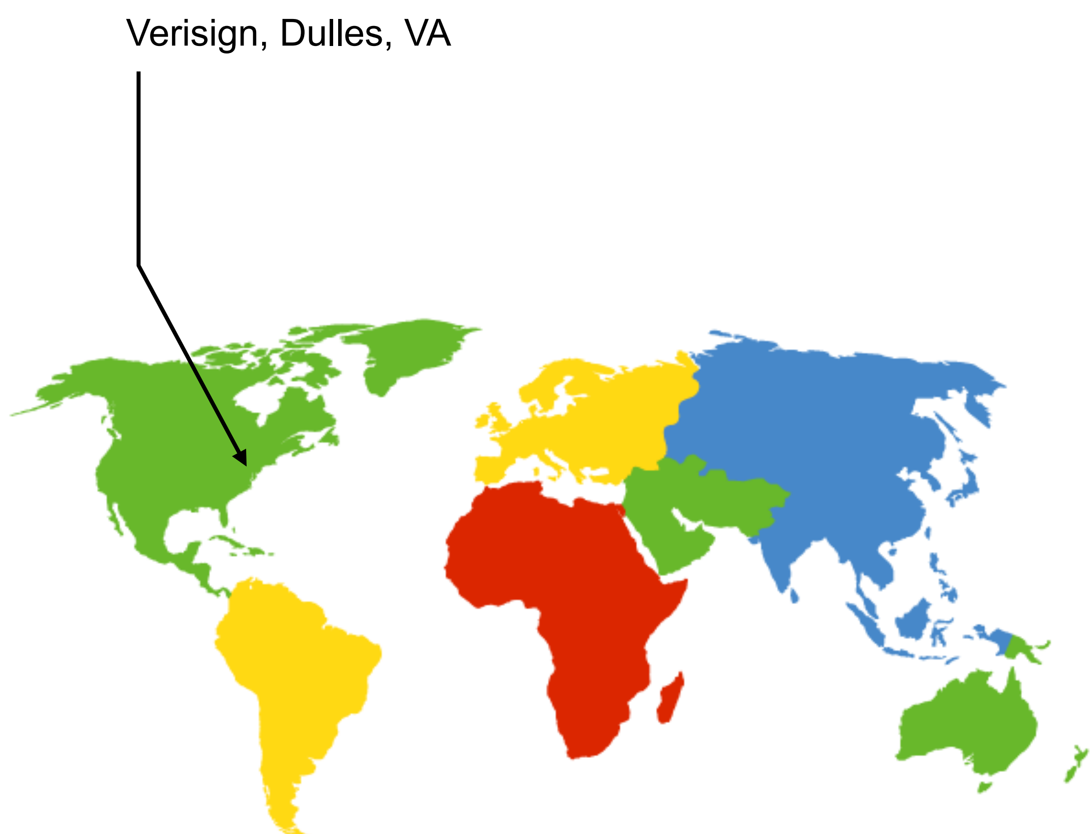


- Client wants IP for www.amazon.com
  - Client queries a root server to find **`com`** DNS server
  - Client queries **`com`** DNS server to get **`amazon.com`** DNS server
  - Client queries **`amazon.com`** DNS server to get IP address for **`www.amazon.com`**

Copyright ©: University of Illinois CS 241 Staff

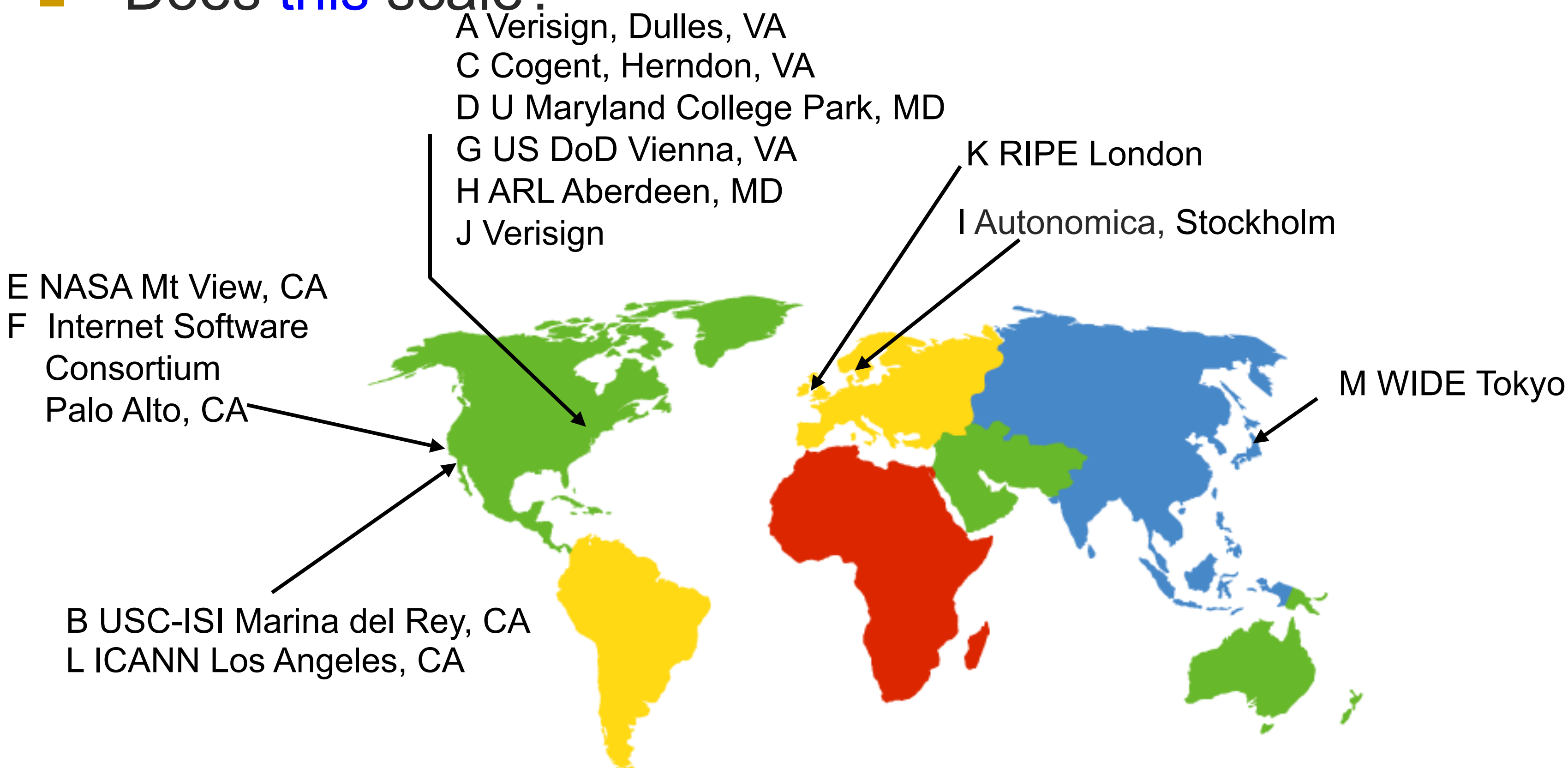
DNS Root

- Located in Virginia, USA
- How do we make the root scale?



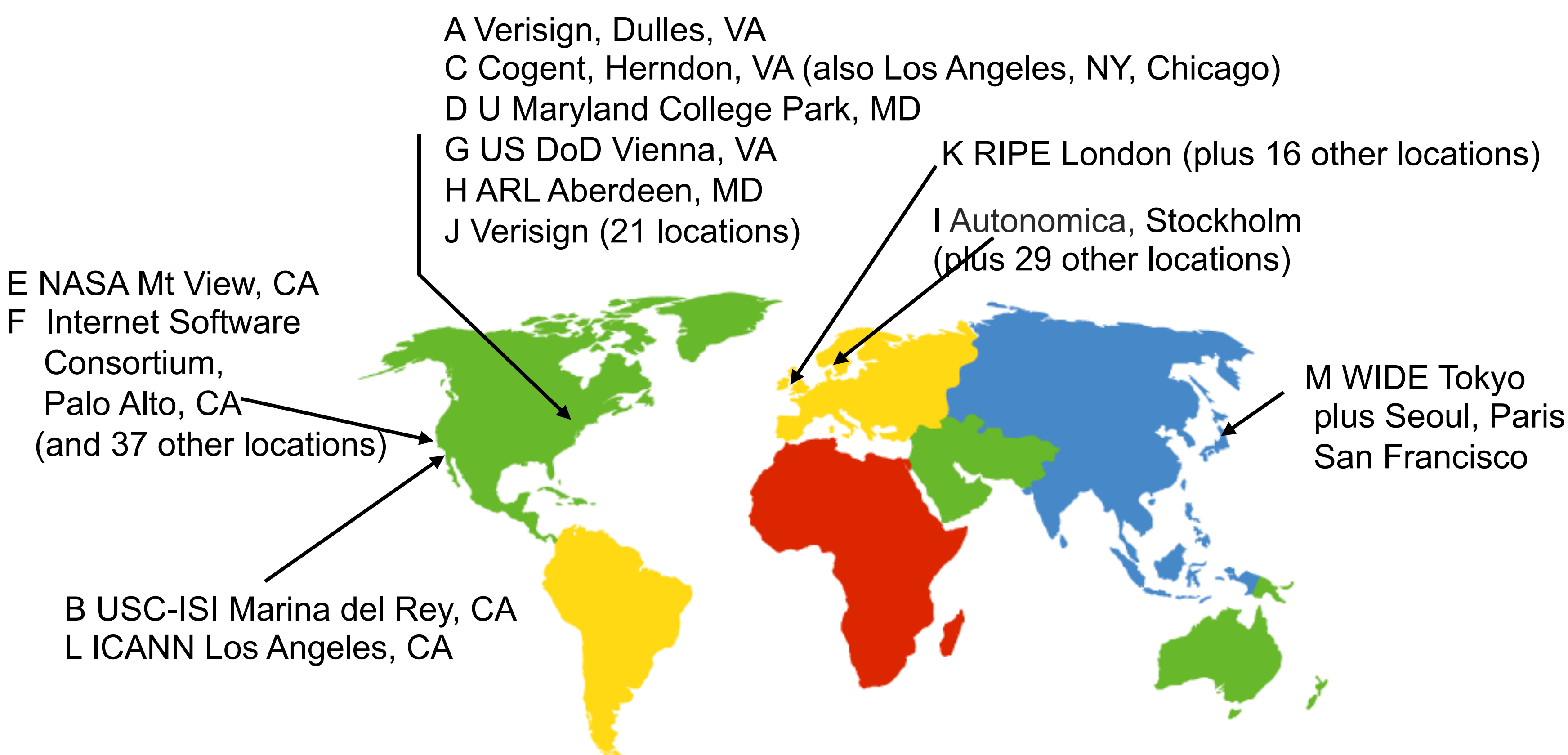
DNS Root Servers

- 13 root servers (see http://www.root-servers.org/)
  - Labeled A through M
- Does this scale?



DNS Root Servers

- 13 root servers each replicated via any-casting (localized routing for addresses)



TLD and Authoritative Servers

- Top-level domain (TLD) servers
  - Responsible for **`com`**, **`org`**, **`net`**, **`edu`**, etc, and all top-level country domains **`uk`**, **`fr`**, **`ca`**, **`jp`**.
    - Network Solutions maintains servers for **`com`** TLD
    - Educause for **`edu`** TLD
- Authoritative DNS servers
  - Organization's DNS servers
  - Provide authoritative hostname to IP mappings for organization's servers (e.g., Web, mail).
  - Can be maintained by organization or service provider

Copyright ©: University of Illinois CS 241 Staff

Local Name Server

- One per ISP (residential ISP, company, university)
  - Also called "default name server"
- When host makes DNS query, query is sent to its local DNS server
  - Acts as proxy, forwards query into hierarchy
  - Reduces lookup latency for commonly searched hostnames
- Hosts learn local name server via...
  - DHCP (same protocol that tells host its IP address)
  - Static configuration (e.g., can use Google's "local" name service at 8.8.8.8 or 8.8.4.4)

Copyright ©: University of Illinois CS 241 Staff

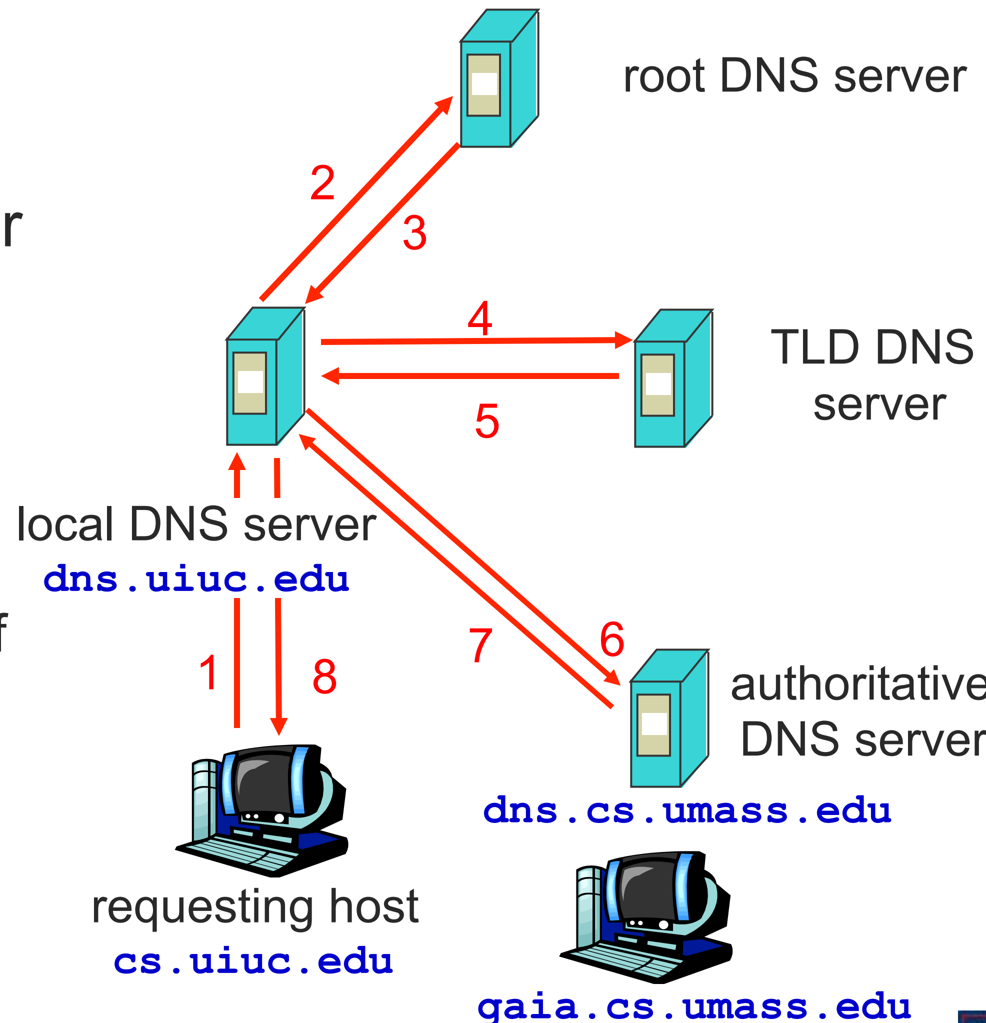
Applications' use of DNS

- Client application (e.g., web browser)
  - Extract server name (e.g., from the URL)
  - Do *gethostbyname()* to trigger resolver code, sending message to local name server

- Server application (e.g. web server)
  - Extract client IP address from socket
  - Optional *gethostbyaddr()* to translate into name

DNS name resolution example

- Host at cs.uiuc.edu wants IP address for gaia.cs.umass.edu
- Iterated query
  - Contacted server replies with name of server to contact
  - "I don't know this name, but ask this server"



Copyright ©: University of Illinois CS 241 Staff

DNS: Caching

- Once (any) name server learns mapping, it caches mapping
  - Cache entries timeout (disappear) after some time
  - TLD servers typically cached in local name servers
    - Thus root name servers not often visited

Copyright ©: University of Illinois CS 241 Staff

Network Address Translation

Copyright © University of Illinois CS 241 Staff

NAT: Network Address Translation

- Approach
  - Assign one router a global IP address
  - Assign internal hosts local IP addresses
- Change IP Headers
  - IP addresses (and possibly port numbers) of IP datagrams are replaced at the boundary of a private network
  - Enables hosts on private networks to communicate with hosts on the Internet
  - Run on routers that connect private networks to the public Internet

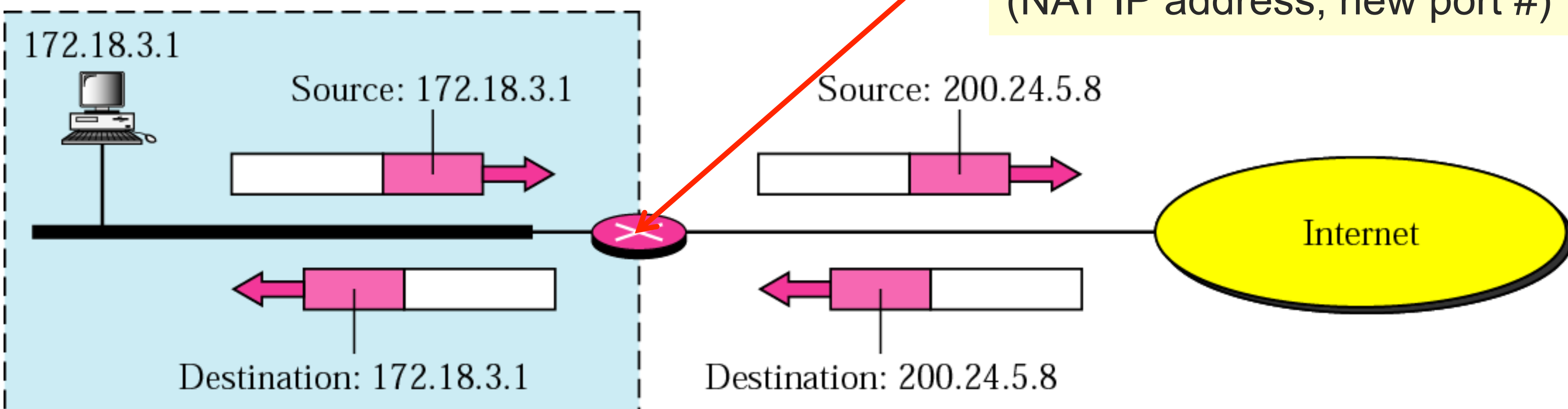
Copyright ©: University of Illinois CS 241 Staff

NAT: Network Address Translation

- Outgoing packet
  - Source IP address (private IP) replaced by global IP address maintained by NAT router
- Incoming packet
  - Destination IP address (global IP of NAT router) replaced by appropriate private IP address

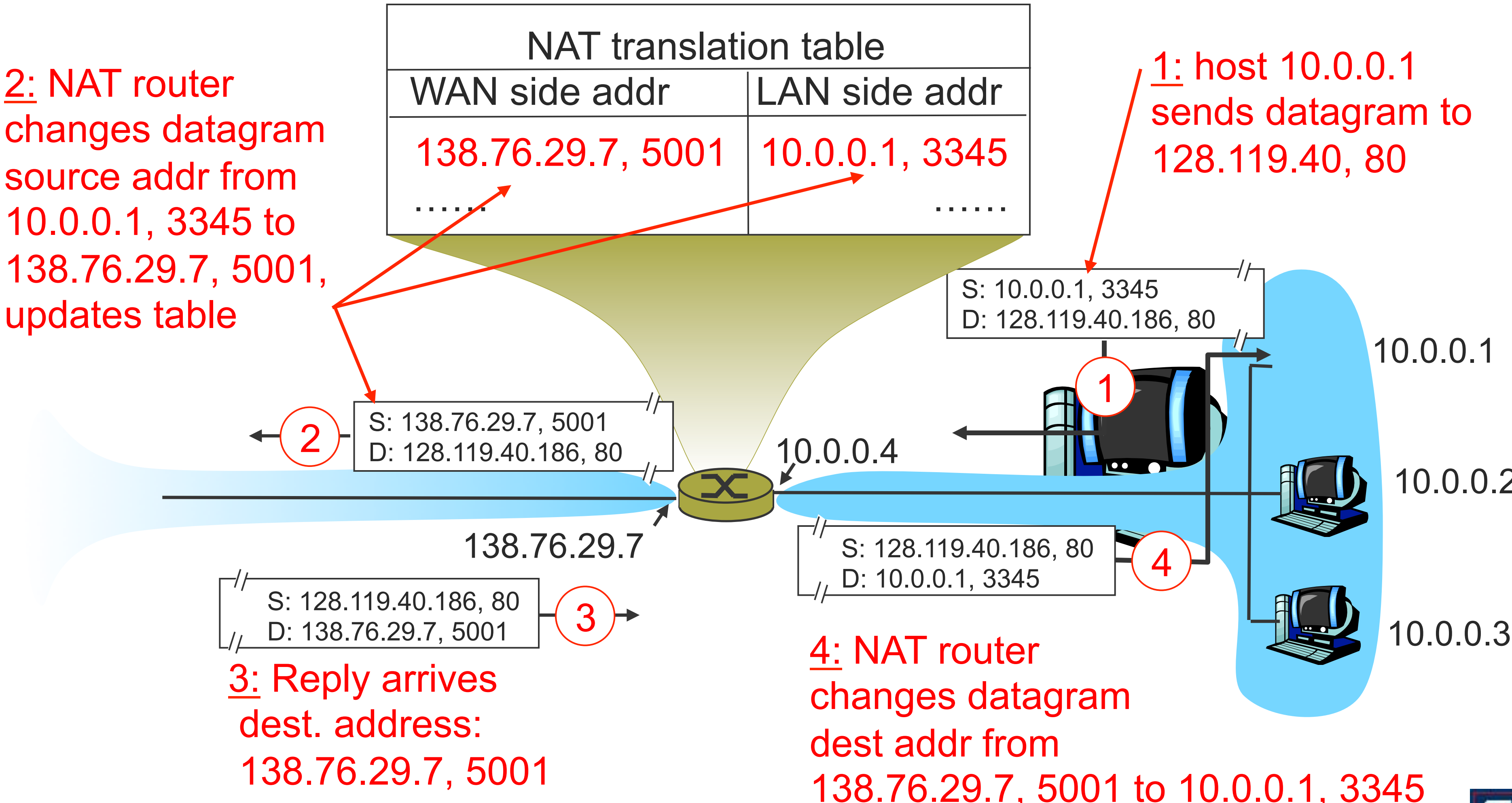
What address do the remote hosts respond to?

NAT router caches translation table:
(source IP address, port #) ➔ (NAT IP address, new port #)



Copyright ©: University of Illinois CS 241 Staff

NAT: Network Address Translation



Copyright ©: University of Illinois CS 241 Staff

NAT: Benefits

- Local network uses just one (or a few) IP address as far as outside world is concerned
  - No need to be allocated range of addresses from ISP
    - Just one IP address is used for all devices
    - Or a few, in a large private enterprise network
    - 16-bit port-number field: 60,000 simultaneous connections with a single LAN-side address!
  - 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 plus)

Copyright ©: University of Illinois CS 241 Staff

NAT: Benefits

- Load balancing
  - Balance the load on a set of identical servers, which are accessible from a single IP address
- NAT solution
  - Servers are assigned private addresses
  - NAT acts as a proxy for requests to the server from the public network
  - NAT changes the destination IP address of arriving packets to one of the private addresses for a server
  - Balances load on the servers by assigning addresses in a round-robin fashion

Copyright ©: University of Illinois CS 241 Staff

NAT: Consequences

- End-to-end connectivity broken
  - NAT destroys universal end-to-end reachability of hosts on the Internet
  - A host in the public Internet often cannot initiate communication to a host in a private network
  - Even worse when two hosts that are in different private networks need to communicate with each other

Copyright ©: University of Illinois CS 241 Staff

NAT: Consequences

- Performance worsens
  - Modifying the IP header by changing the IP address requires that NAT boxes recalculate the IP header checksum
  - Modifying port number requires that NAT boxes recalculate TCP checksum
- Fragmentation issues
  - Datagrams fragmented before NAT device must not be assigned different IP addresses or different port numbers

Copyright ©: University of Illinois CS 241 Staff

NAT: Consequences

- Broken if IP address in application data
  - Applications often carry IP addresses in the payload of the application data
  - No longer work across a private-public network boundary
  - Hack: Some NAT devices inspect the payload of widely used application layer protocols and, if an IP address is detected in the application-layer header or the application payload, translate the address according to the address translation table

Copyright ©: University of Illinois CS 241 Staff

NAT: Consequences

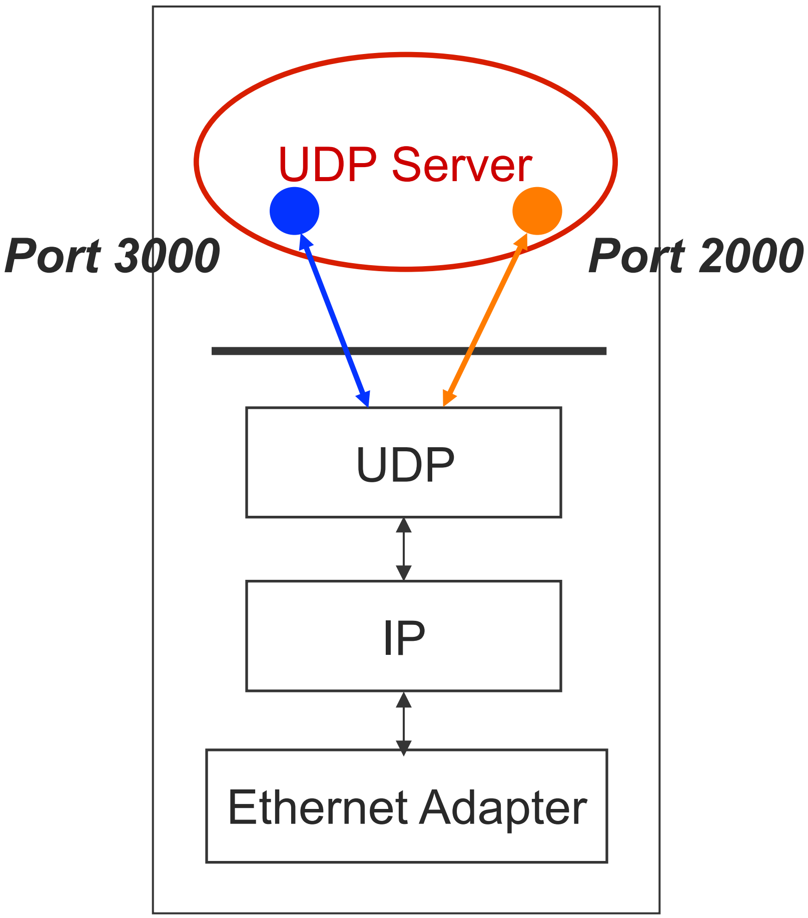
- Ossification of Internet protocols
  - NAT must be aware of port numbers which are inside transport header
  - Existing NATs don't support your fancy new transport protocol
    - and might even block standard protocols like UDP
  - Result: Difficult to invent new transport protocols
    - ...unless they just pretend to be TCP

Copyright ©: University of Illinois CS 241 Staff

Bonus slides

Copyright ©: University of Illinois CS 241 Staff

A UDP Server



- How can a UDP server service multiple ports simultaneously?

Copyright ©: University of Illinois CS 241 Staff

UDP Server: Servicing Two Ports

```
int s1;                          /* socket descriptor 1 */
int s2;                          /* socket descriptor 2 */

/* 1) create socket s1 */
/* 2) create socket s2 */
/* 3) bind s1 to port 2000 */
/* 4) bind s2 to port 3000 */

while(1) {
    recvfrom(s1, buf, sizeof(buf), ...);
    /* process buf */
    recvfrom(s2, buf, sizeof(buf), ...);
    /* process buf */
}
```

What problems does this code have?

Copyright ©: University of Illinois CS 241 Staff

Building Timeouts with Select and Poll

- Time structure

Number of seconds since midnight, January 1, 1970 GMT

```
struct timeval {
    long tv_sec;           /* seconds */
    long tv_usec;  /* microseconds */
};
```

unix will have its own "Y2K" problem one second after 10:14:07pm, Monday January 18, 2038 (will appear to be 3:45:52pm, Friday December 13, 1901)

Copyright ©: University of Illinois CS 241 Staff

Select

Which file descriptors are set and what should the timeout value be?

- High-resolution sleep function
  - All descriptor sets **NULL**
  - Positive **timeout**
- Wait until descriptor(s) become ready
  - At least one descriptor in set
  - **timeout** **NULL**
- Wait until descriptor(s) become ready or timeout occurs
  - At least one descriptor in set
  - Positive **timeout**
- Check descriptors immediately (poll)
  - At least one descriptor in set
  - 0 **timeout**

Copyright ©: University of Illinois CS 241 Staff

Select: Example

```
fd_set my_read;
FD_ZERO(&my_read);
FD_SET(0, &my_read);

if (select(1, &my_read, NULL, NULL) == 1) {
   ASSERT(FD_ISSET(0, &my_read);
   /* data ready on stdin */
```

What went wrong: after select indicates data available on a connection, read returns no data?

Copyright ©: University of Illinois CS 241 Staff

Select: Timeout Example

```
int main(void) {
    struct timeval tv;
    fd_set readfds;

    tv.tv_sec = 2;
    tv.tv_usec = 500000;

    FD_ZERO(&readfds);
    FD_SET(STDIN, &readfds);

    // don't care about writefds and exceptfds:
    select(1, &readfds, NULL, NULL, &tv);

    if (FD_ISSET(STDIN, &readfds))
        printf("A key was pressed!\n");
    else
        printf("Timed out.\n");

    return 0;
}
```

Wait 2.5 seconds for something to appear on standard input

Copyright ©: University of Illinois CS 241 Staff

Poll

- High-resolution sleep function
  - 0 **`nfds`**
  - Positive **`timeout`**
- Wait until descriptor(s) become ready
  - **`nfds`** > 0
  - **`timeout`** **`INFTIM`** or -1
- Wait until descriptor(s) become ready or timeout occurs
  - **`nfds`** > 0
  - Positive **`timeout`**
- Check descriptors immediately (poll)
  - **`nfds`** > 0
  - 0 **`timeout`**

Copyright ©: University of Illinois CS 241 Staff

`select()` vs. `poll()`

Which to use?

- BSD-family (e.g., FreeBSD, MacOS)
  - `poll()` just calls `select()` internally
- System V family (e.g., AT&T Unix)
  - `select()` just calls `poll()` internally

Copyright ©: University of Illinois CS 241 Staff