CS2105

An *Awesome* Introduction to Computer Networks

Lecture 3 discussion

NUS National University of Singapore
Department of Computer Science
School of Computing

Domain Name System [RFC 1034, 1035]

- Two ways to identify a host:
  - **Hostname**, e.g., www.example.org
  - **IP address**, e.g., 93.184.216.34

- DNS (Domain Name System) translates between the two.
  - A client must carry out a DNS query to determine the IP address corresponding to the server name (e.g., www.example.org) prior to the connection.

DNS: Resource Records (RR)

- Mapping between host names and IP addresses (and others) are stored as resource records (RR).

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

type = A

- **`name`** is hostname
- **`value`** is IP address

type = NS

- **`name`** is domain (e.g., **`nus.edu.sg`**)
- **`value`** is hostname of authoritative name server for this domain

type = CNAME

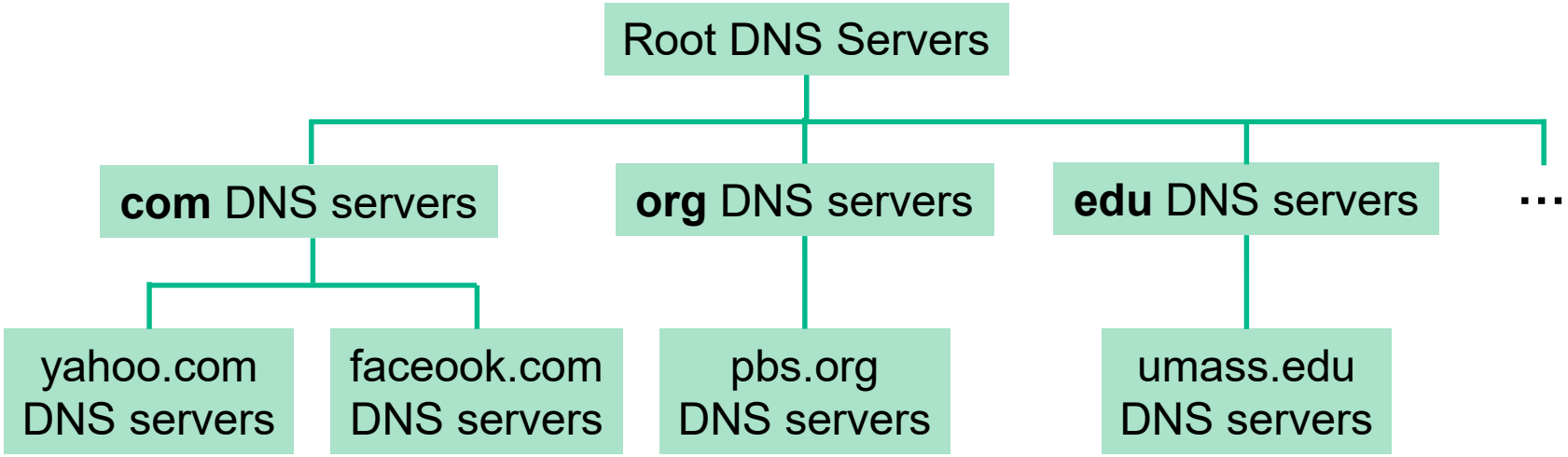
- **`name`** is alias name (e.g. **`www.nus.edu.sg`**) for some "canonical" (the real) name
- **`value`** is canonical name (e.g. **`mgnzsqc.x.incapdns.net`**)

type = MX

- **`value`** is name of mail server associated with **`name`**

Distributed, Hierarchical Database

- DNS stored RR in distributed databases implemented in hierarchy of many name servers.



A client wants IP address for www.facebook.com:

- client queries root server to find .com DNS server
- client queries .com DNS server to get facebook.com DNS server
- client queries facebook.com DNS server to get IP address for www.facebook.com

Local DNS Server

- Does not strictly belong to hierarchy
- Each ISP (residential ISP, company, university) has one local DNS server.
  - also called “default name server”
- When host makes a DNS query, query is sent to its local DNS server
  - Retrieve name-to-address translation from local cache
  - Local DNS server acts as proxy and forwards query into hierarchy if answer is not found locally

DNS Caching

- Once a name server learns mapping, it *caches* mapping.
  - cached entries may be out-of-date (best effort name-to-address translation!)
  - cached entries expire after some time (TTL).
  - 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 runs over **UDP**.

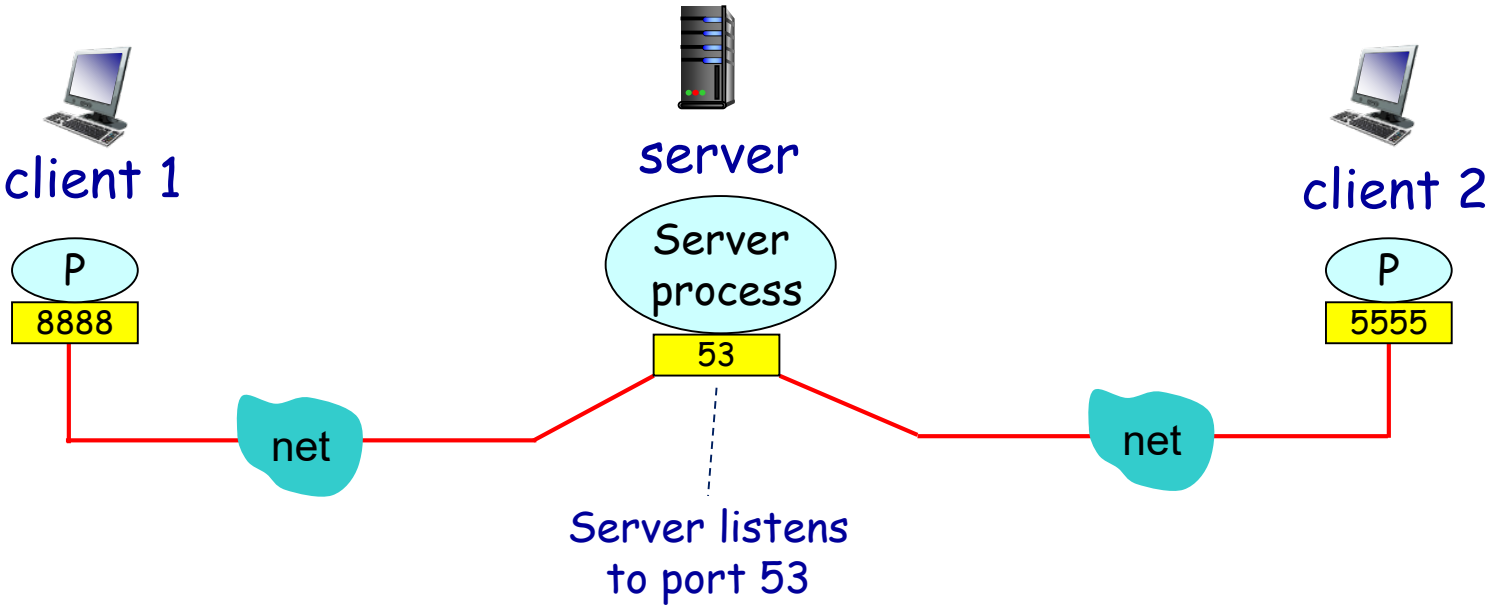
Socket Programming

- Applications (or processes) treat the Internet as a black box, sending and receiving messages through sockets.
- Two types of sockets
  - TCP: reliable, byte stream-oriented socket
  - UDP: unreliable datagram socket

Socket Programming with *UDP*

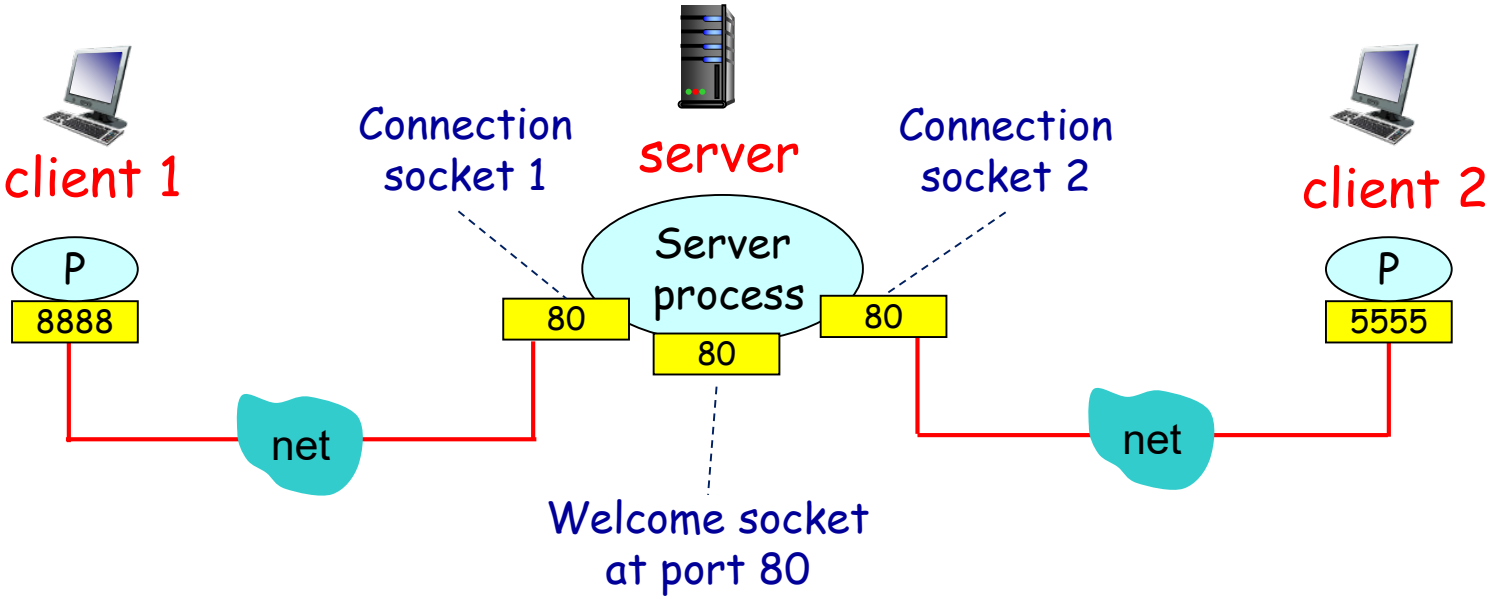
UDP: no "connection" between client and server

- Sender (client) explicitly attaches destination IP address and port number to each packet.
- Receiver (server) extracts sender IP address and port number from the received packet.



Socket Programming with *TCP*

- When client creates socket, client TCP establishes a connection to server TCP.
- When contacted by client, server TCP creates a new socket for server process to communicate with that client.
  - allows server to talk with multiple clients individually.



Example: UDP Echo Server

```
from socket import *
```
include Python's socket library

```
serverPort = 2105

# create a socket
serverSocket = socket(AF_INET, SOCK_DGRAM)

# bind socket to local port number 2105
serverSocket.bind(('', serverPort))
print('Server is ready to receive message')

# extract client address from received packet
message, clientAddress = serverSocket.recvfrom(2048)

serverSocket.sendto(message, clientAddress)

serverSocket.close()
```

IPv4

UDP socket

receive datagram
buffer size: 2048B

Example: TCP Echo Server

```
from socket import *

serverPort = 2105

serverSocket = socket(AF_INET, SOCK_STREAM)

serverSocket.bind(('', serverPort))

serverSocket.listen()
print('Server is ready to receive message')

connectionSocket, clientAddr = serverSocket.accept()
message = connectionSocket.recv(2048)

connectionSocket.send(message)

connectionSocket.close()
```

TCP socket

listens for incoming TCP request
(not available in UDP socket)

returns a *new* socket
to communicate with
client socket