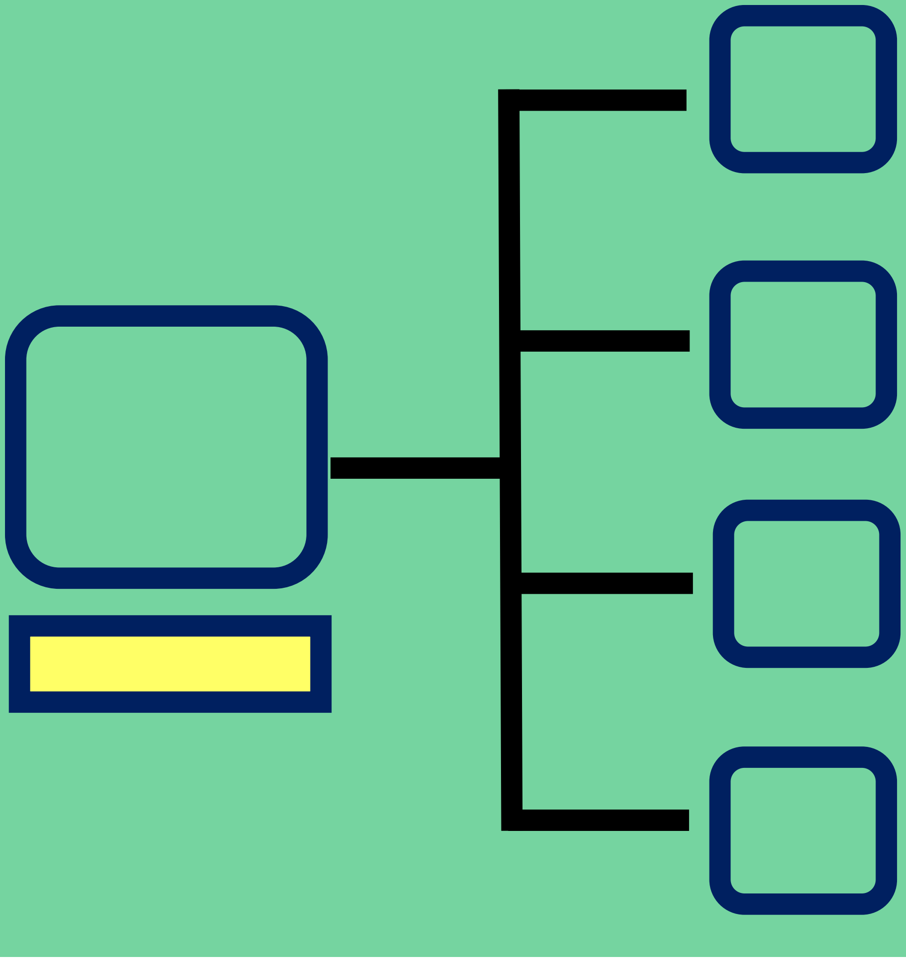
COMP2211: Networks and Systems



Lecture 3

Application Layer

Re-cap

- Packet switching vs. circuit switching
- OSI reference model
- Routing and forwarding
- Delay and loss in networks

Today’s Outline

- Network application architecture
  - Client-server architecture
  - P2P architecture
- Protocols
  - TCP (Transmission Control Protocol)
  - UDP (User Datagram Protocol)
- Socket programming
  - TCP
  - UDP

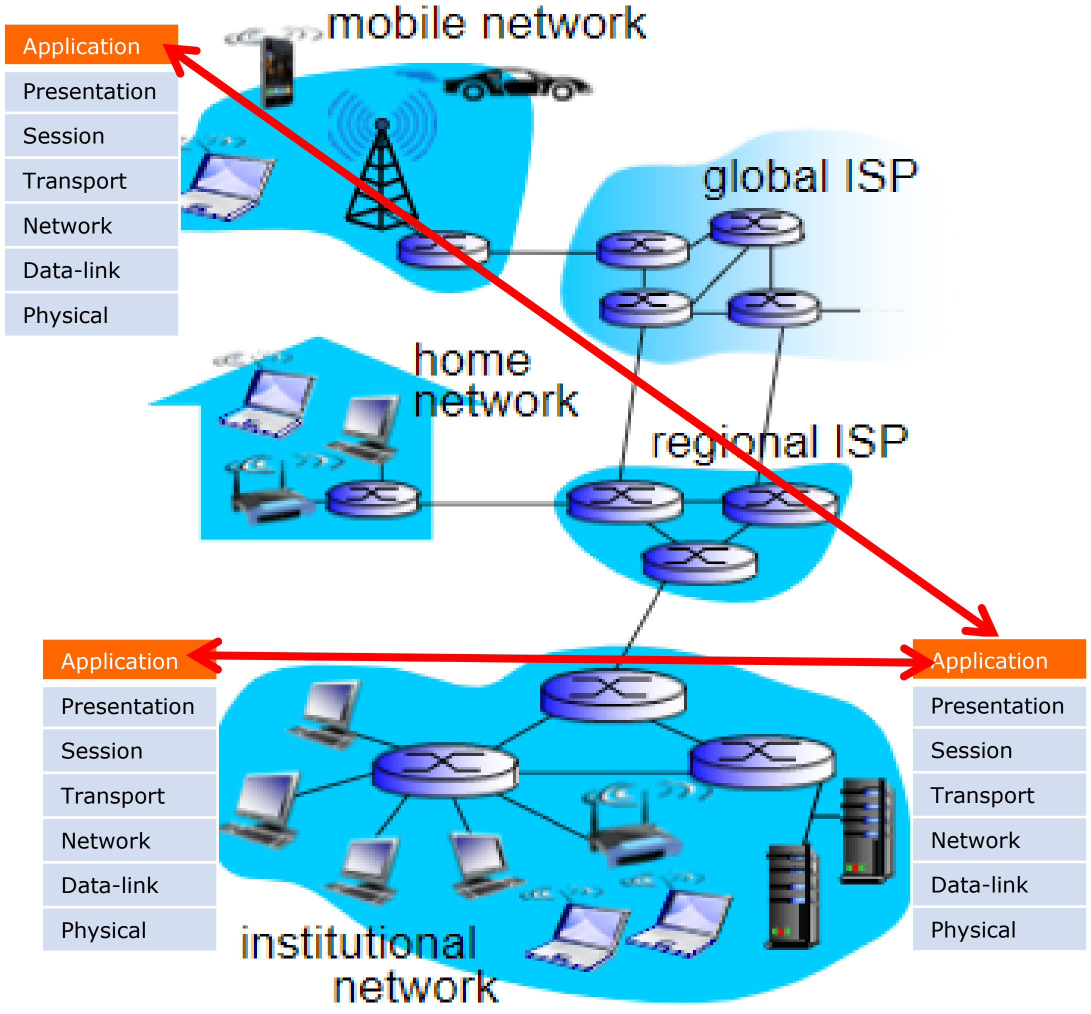
Creating a Network User Application

Write programs that:

- Run on (different) end systems
- Communicate over network:
  e.g., web server software communicates with browser software

No need to write software for network-core devices

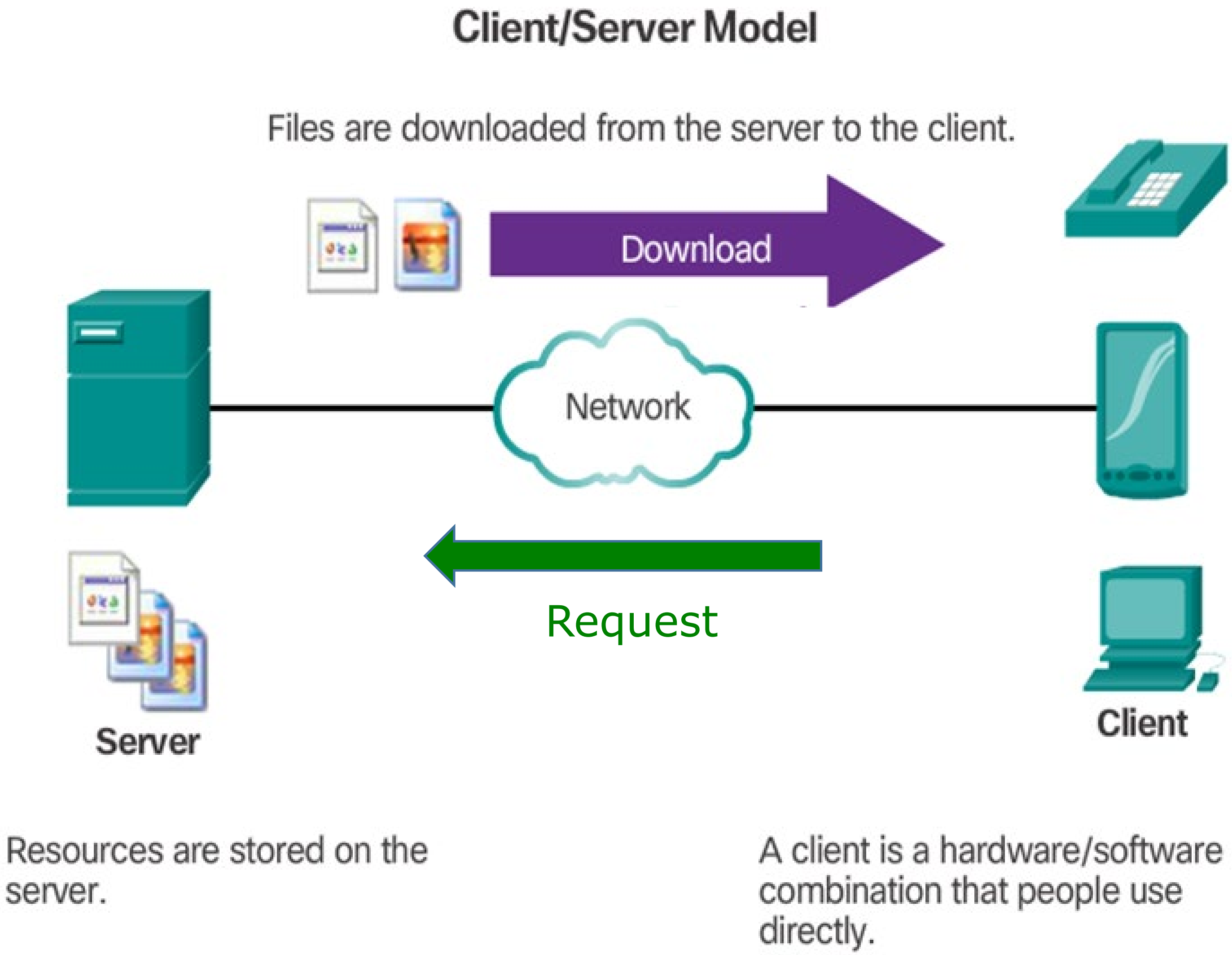
- Network-core devices do not run user applications
- Applications on end systems allow for rapid app development, propagation



Two application architectures:

- Client-server: HTTP, DNS, DHCP, Emails, FTP
- Peer-to-peer (P2P): eDonkey, G2, BitTorrent, Bitcoin

Client-Server Architecture


Client/Server Model
Files are downloaded from the server to the client.
Download
Network
Request
Server
Client
Resources are stored on the server.
A client is a hardware/software combination that people use directly.

Client:

- Devices requesting information/services
- May be intermittently connected
- May have dynamic IP addresses
- Do not communicate directly with each other

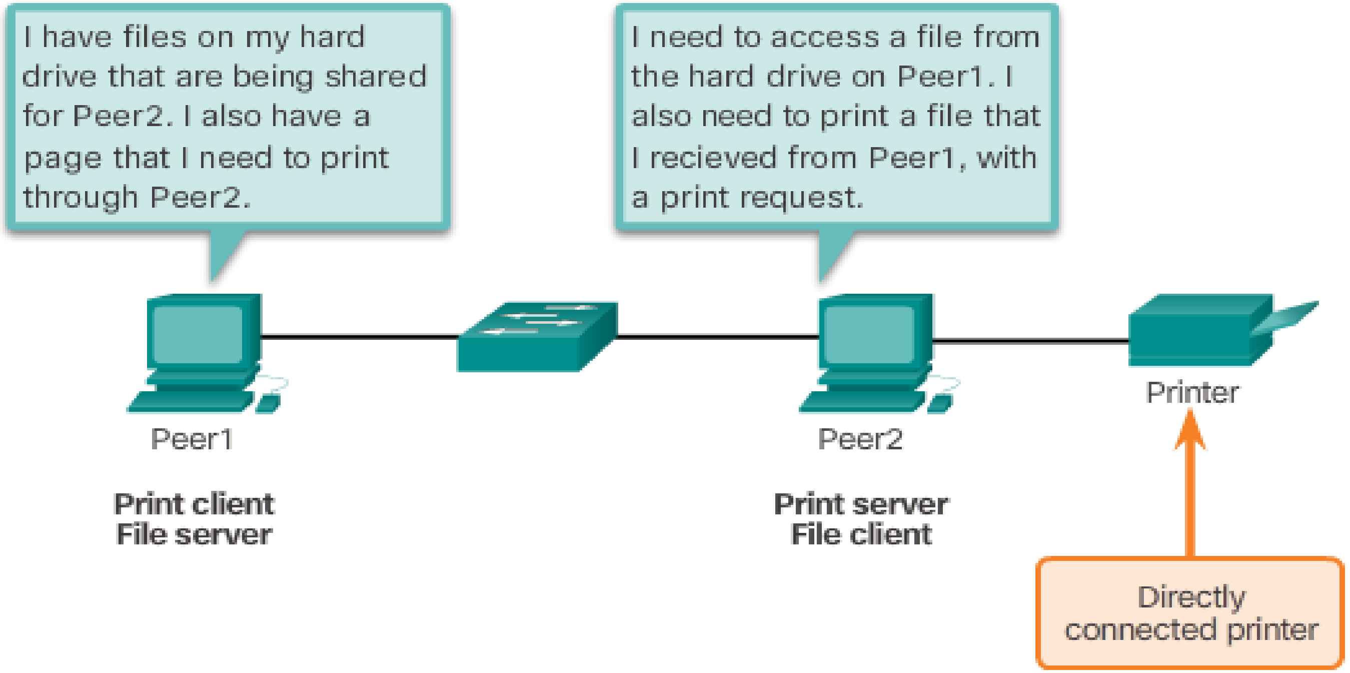
Server:

- Devices responding to the request by providing data/services
- Always-on devices
- Fixed (static) IP addresses
- Data centres for scaling

Peer-to-Peer Architecture

There is no a dedicated server in a peer-to-peer network.

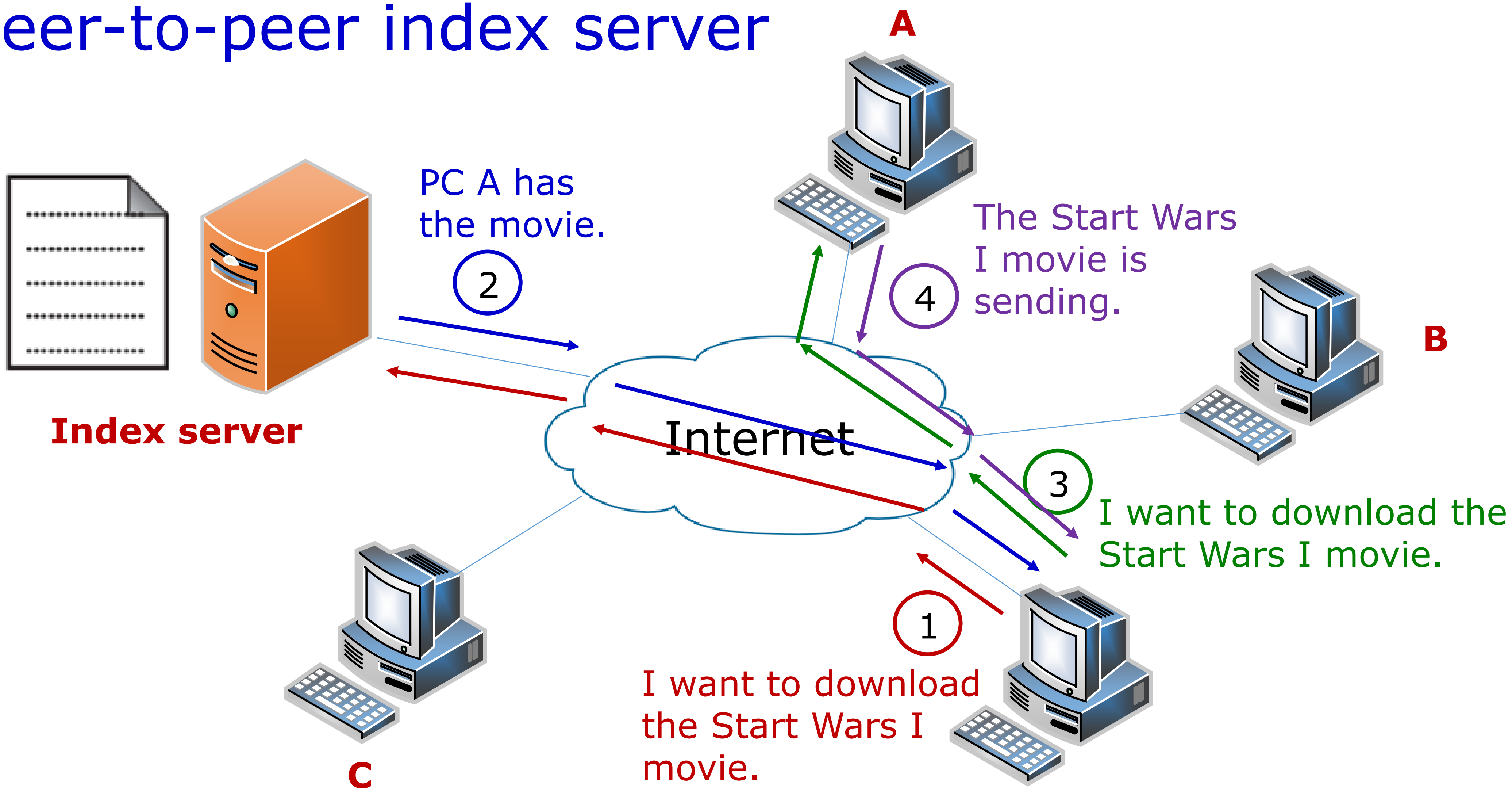
- Every connected device (a peer) can function as both a server and a client.
- The roles of client and server are set on a per request basis.
- Self scalability: new peers bring new service capacity, as well as new service demands.
- Peers are intermittently connected and change IP addresses.


I have files on my hard drive that are being shared for Peer2. I also have a page that I need to print through Peer2.
I need to access a file from the hard drive on Peer1. I also need to print a file that I recieved from Peer1, with a print request.
Peer1
Print client
File server
Peer2
Print server
File client
Printer
Directly connected printer

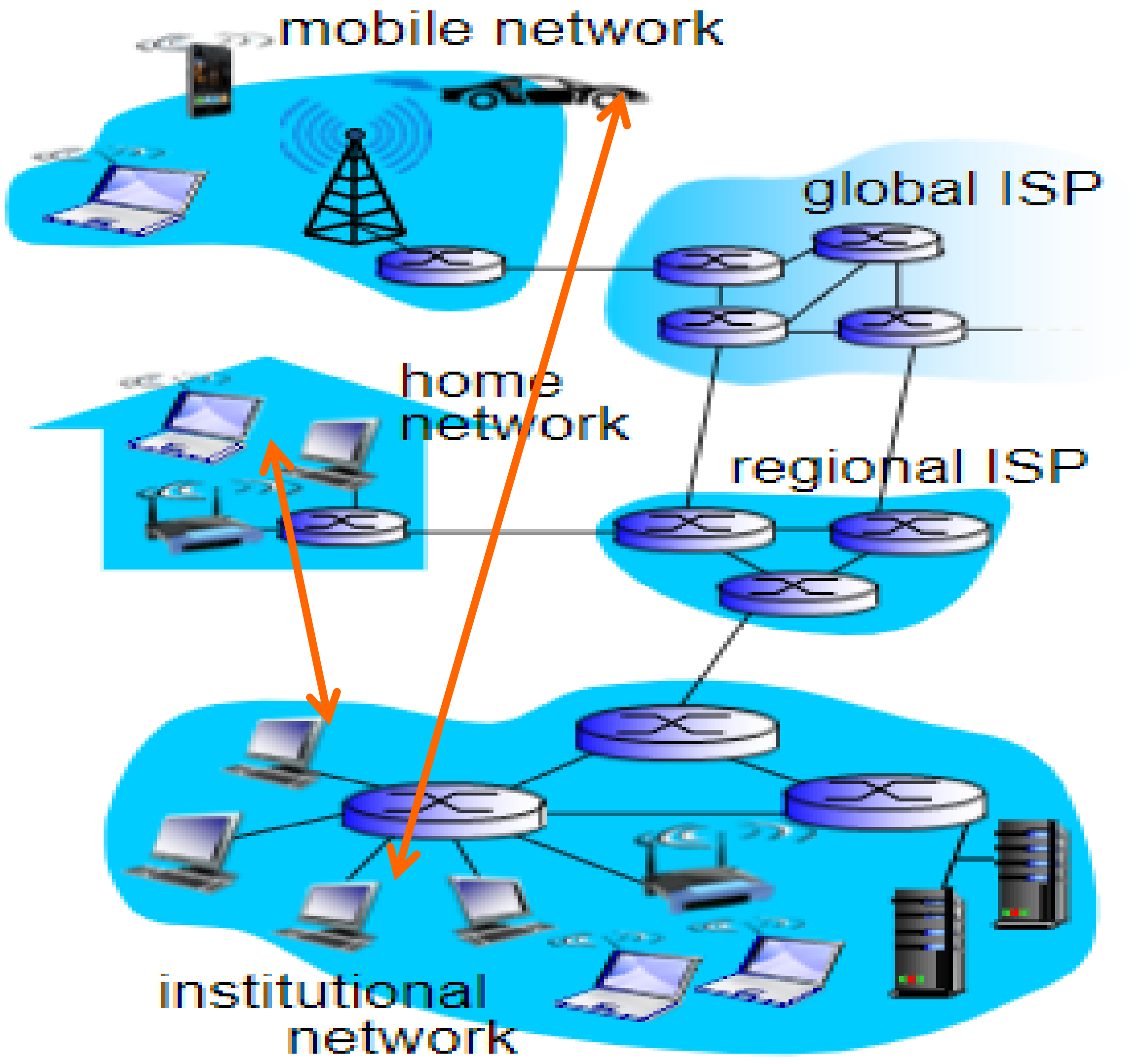
How does a device know where the resources/services that it needs are?

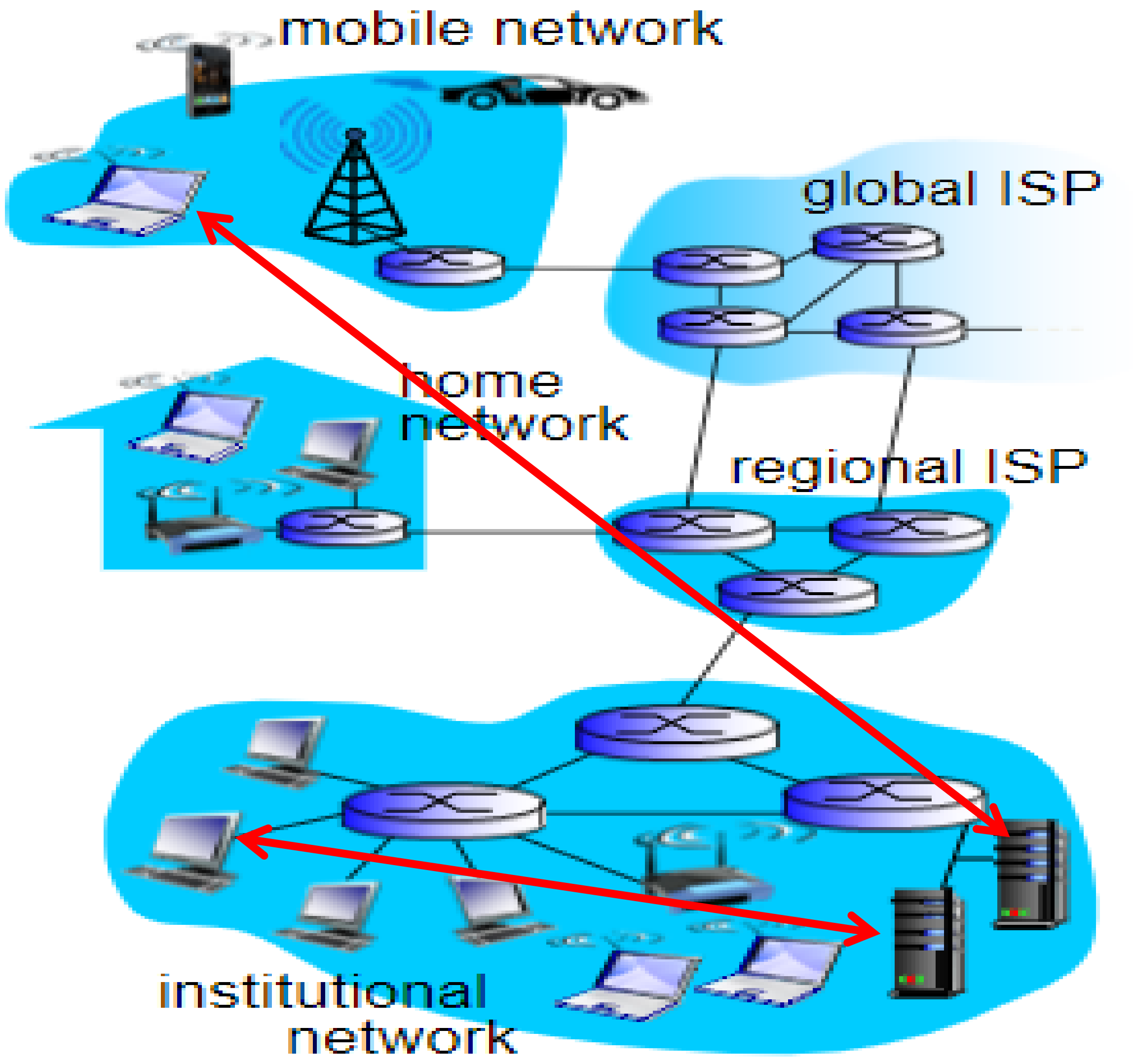
Peer-to-Peer Architecture (cont.)

Peer-to-peer index server


A
PC A has the movie.
2
The Start Wars I movie is sending.
4
B
Index server
Internet
3
I want to download the Start Wars I movie.
1
I want to download the Start Wars I movie.
C

Client-Server *vs.* P2P Architectures





Processes Communicating

Processes are programs running within a host.

- Within same host, two processes communicate using inter-process communication (defined by OS).
- Processes on different hosts communicate by exchanging messages.

A process is analogous to a house, and its socket is analogous to its door.

Socket is a software mechanism that allows a process to

- create and send messages into, and
- receive messages from the network.

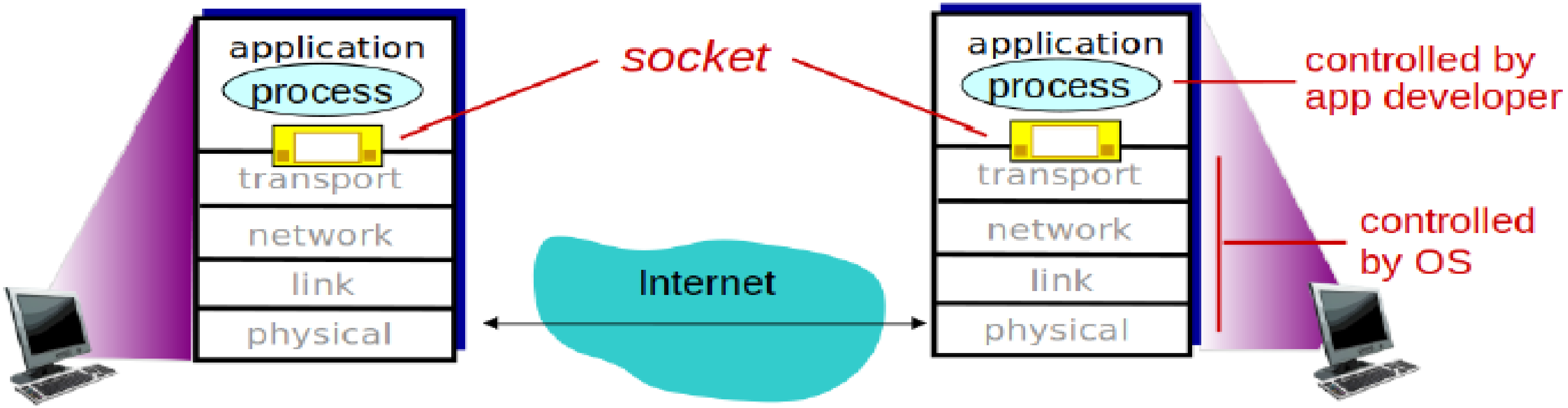
Interface between application layer and transport layer.

Sockets

- A process sends/receives messages to/from its socket.
- Socket analogous to door

  - Sending process shoves messages out of the door

  - Sending process relies on transport infrastructure (on the other side of the door) to deliver messages to a socket at the receiving process



What Transport Services Does An Application Need?

Reliability

- Some applications (e.g., file transfer, web transactions) require 100% reliable data transfer
- Other applications (e.g., audio, video) can tolerate some loss

Timing

- Some applications (e.g., Internet telephony, interactive games) require low delay to be “effective”

Security

- Encryption, data integrity, …

and many more other service requirements …

Transport-layer Protocol Services

Services provided by the TCP protocol:

- Connection-oriented: setup required between client and server processes
- Reliable transport between sending and receiving process
- Flow control: sender won’t overwhelm receiver
- Full-duplex connection: connection can send messages to each other at the same time

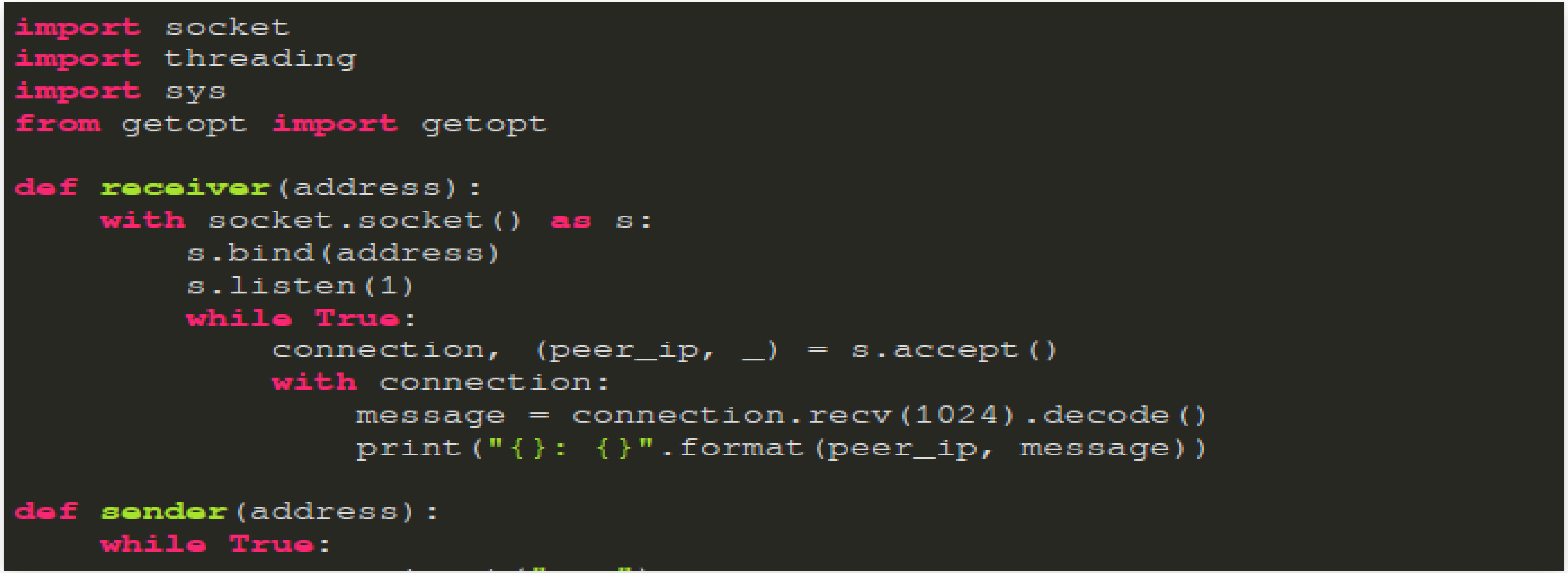
Transport-layer Protocol Services (cont.)

Services provided by the UDP protocol:

- Unreliable data transfer between sending and receiving processes
- Does not provide: reliability, flow control, congestion control, security, or connection setup
- Short-delay transmissions between sending and receiving processes

Peer-to-Peer Architecture: Example

P2P messenger - using TCP



```
import socket
import threading
import sys
from getopt import getopt

def receiver(address):
    with socket.socket() as s:
        s.bind(address)
        s.listen(1)
        while True:
            connection, (peer_ip, _) = s.accept()
            with connection:
                message = connection.recv(1024).decode()
                print("{}: {}".format(peer_ip, message))

def sender(address):
    while True:
```

Python Socket Programming Documentation: https://docs.python.org/3/library/socket.html

Socket programming HOWTO: https://docs.python.org/3/howto/sockets.html

Application-layer Protocols

- Types of messages exchanged, e.g., requests, responses
- Message syntax: What fields in messages & how fields are delineated
- Message semantics: Meaning of information in fields

Rules for when and how processes send and respond to messages

Types of Application-layer Protocols

Open protocols:

- Defined in Request For Comments (RFC)
- Allow for interoperability, e.g., HTTP, SMTP

Proprietary protocols

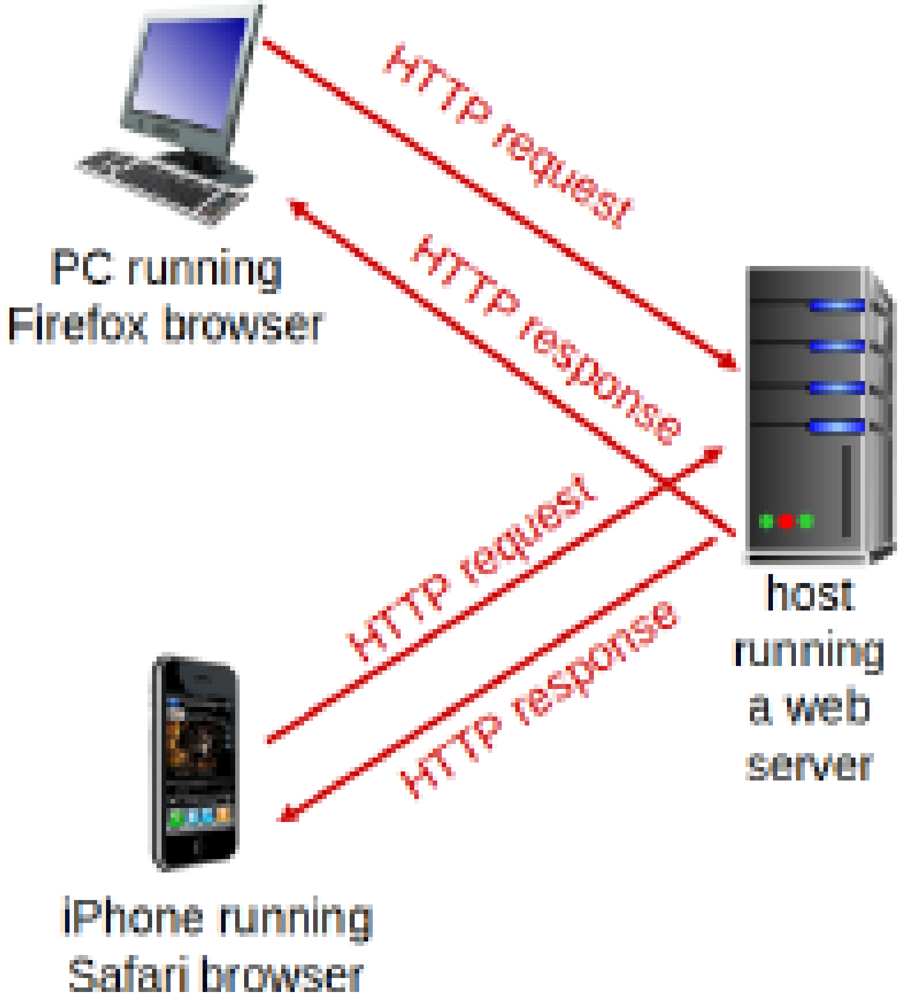
- Skype, Zoom, etc.

HTTP Overview

HTTP: Hypertext Transfer Protocol

- The application-layer protocol for web services
- Client-server model

Client	Browser that requests, receives, (using HTTP protocol) and displays web objects
Server	Web server sends (using HTTP protocol) objects in response to requests


HTTP request
HTTP response
PC running Firefox browser
HTTP request
HTTP response
host running a web server
iPhone running Safari browser

HTTP Overview (cont.)

Use TCP at the transport layer (i.e., HTTP/TCP)

- Client initiates TCP connection (creates socket) to server, port 80
- Server accepts TCP connection from client
- HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
- TCP connection closed
- HTTP is "stateless"
- Server maintains no information about past client requests

Protocols that maintain "state" are complex!

- Past history (state) must be maintained
- If server/client crashes, their views of "state" may be inconsistent, must be reconciled

HTTP Connections

Non-persistent HTTP

- At most one object sent over TCP connection
- Connection then closed
- Downloading multiple objects requires multiple connections

Persistent HTTP

- Multiple objects can be sent over single TCP connection between client and server

Non-persistent HTTP

Suppose a user enters an URL: *www.durham.ac.uk/cs*

1. HTTP client initiates a TCP connection to HTTP server at *durham.ac.uk* on port 80

2. HTTP server at host is waiting for TCP connections on port 80 and accepts to the connection notifying the client

3. HTTP client sends HTTP request message (containing URL) into the TCP connection socket. The message indicates that the client wants *www.durham.ac.uk/cs/index.html*

4. HTTP server receives request message, forms response message containing requested object, and sends message into its socket.

Non-persistent HTTP (cont.)

5. HTTP server closes TCP connection as soon as the client receives the response intact

6. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced objects (e.g., JPEG)

7. Steps 1-5 are repeated for each of the 10 jpeg objects

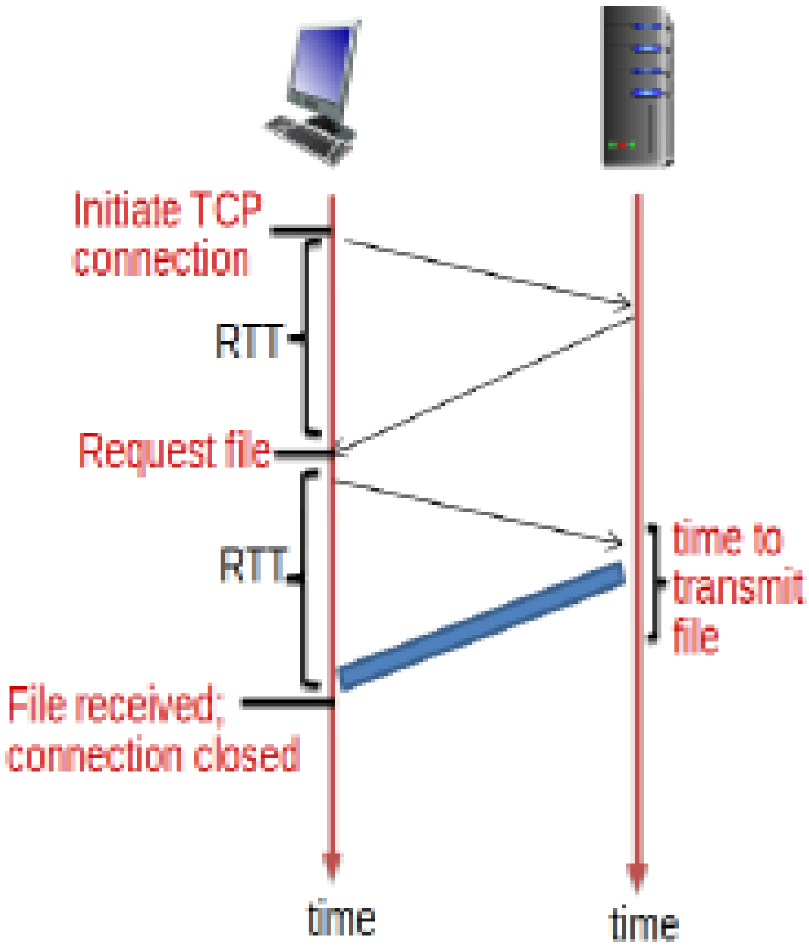
Non-persistent HTTP: Response Time

Round trip time (RTT): time for a small packet to travel from client to server and back

Non-persistent HTTP response time:

2RTT + file transmission time

- One RTT to initiate TCP connection
- One RTT for HTTP request and first few bytes of HTTP response to return



Persistent HTTP

- Server leaves connection open after sending response.
- Subsequent HTTP messages between same client/server sent over open connection.
- Client sends requests as soon as it encounters a referenced object.

Persistent HTTP response time:

RTT + file transmission time

- One RTT for HTTP request and first few bytes of HTTP response to return

- Assuming connections to server already established
- Assuming all files requested in parallel

HTTP Standards

- HTTP/1.1 introduced 1997

- HTTP/2 currently in use

  - requires Transport Layer Security (TLS) 1.2 or newer

- HTTP/3 proposed next standard

  - already in use by some browsers
  - uses UDP not TCP (see QUIC)
  - attempts to solve “head-of-line” blocking

Summary

- Application-layer architectures:
  - Client-server architecture
  - Peer-to-peer architecture
- Application service requirements
- Transport-layer protocols:
  - TCP: connection-oriented, reliable
  - UDP: connectionless, unreliable
- Application-layer protocols
- HTTP: non-persistent http, persistent http

Next Lecture:

- Transport layer