1 The Internet

I.1

Fundamental Concepts

What is the Internet?

- a network of networks connecting millions of computing devices throughout the world
- each network is administered independently of all other networks

I.1

Fundamental Concepts

Client-Server Network Architecture

- a network model designed for the end users called clients, to access resources from a central system known as server
- responsibilities of a server
  - performs all the major operations such as security and network management
  - manages all the resources
- all the clients communicate with each other through a server

Fundamental Concepts

Client-Server Network Architecture

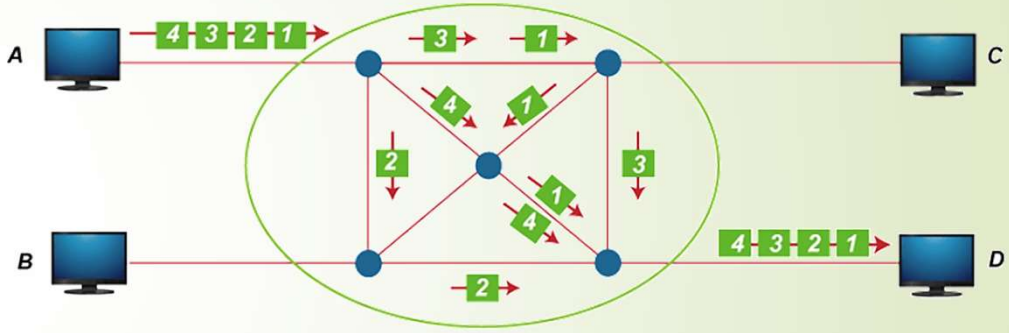
- advantages
  - a client/server network contains a centralized system
  - a client/server network has a dedicated server that improves the overall performance of the whole system
  - security is better in client/server network as a single server administers the shared resources
- disadvantages
  - client/server network is expensive as it requires the server with large resources
  - a server has a Network Operating System(NOS) to provide the resources to the clients, but the cost of NOS is very high
  - it requires a dedicated network administrator to manage all the resources

I.1

Fundamental Concepts

The Internet Model

- a packet-switched network
- packet structure
- key points
  - each packet is processed independently of all other packets
  - there is no need to set up a "connection" or "circuit" with another node before sending it a packet



I.1

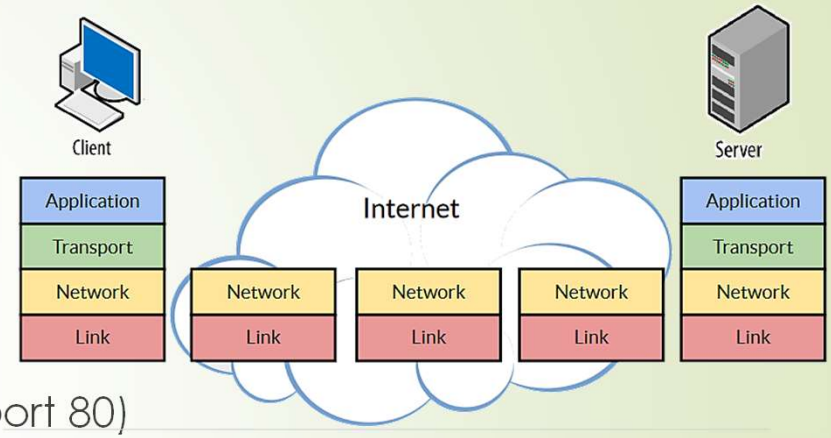
Fundamental Concepts

The Internet Protocols

- all computers connected to the Internet adhere to a set of Internet Standards – the protocols used for communication across the network
- two basic abstractions
  - a reliable, in-order byte stream to an application running on another computer
  - an unreliable message (finite datagrams) service to an application running on another computer

Fundamental Concepts

The Internet Layers



- Application layer
  name of document (e.g., index.html)
- Transport layer
  port identifying what service (e.g., HTTP on port 80)
- Network/Internet layer
  global IP address identifying which computer (e.g., 171.67.76.12)
- Network Access/Link layer
  MAC address specifying which device on a link (e.g., 00:25:90:e7:10:48

Fundamental Concepts

TCP: Reliable, in-order byte stream

- a connection-oriented protocol
  a connection is established between both the ends of the transmission
- a TCP connection behaves much like a bidirectional pipe
  - both sides can read, both sides can write
  - usually, one computer (a server) waits for connection requests from clients
- transfers the data in the form of contiguous stream of bytes
- assigns a sequence number to each byte transmitted and expects a positive acknowledgement. If ACK is not received within a timeout interval, then the data is retransmitted to the destination.

I.1

Fundamental Concepts

Naming of internet nodes

- every host machine connected to the Internet has a globally unique IP-address
- since IP-addresses are not particularly convenient for humans, each host may be assigned a hostname
- DNS (domain name service) is a directory service that provides a mapping between the name of a host on the network and its IP-address

Fundamental Concepts

Working of DNS

- DNS is a client/server network communication protocol
- DNS implements a distributed database to store the name of all the hosts available on the internet
  - if a client sends a request containing a hostname, then a piece of software such as DNS resolver sends a request to the DNS server to obtain the IP-address of a hostname
  - if DNS server does not contain the IP-address associated with a hostname, then it forwards the request to another DNS server
  - if IP-address has arrived at the resolver, which in turn completes the request over the internet protocol

1

The Internet

- Fundamental Concepts
- Network Programming
- Security Concerns
- Session Handling
- Network Programming II

2

The Hypertext Transfer Protocol

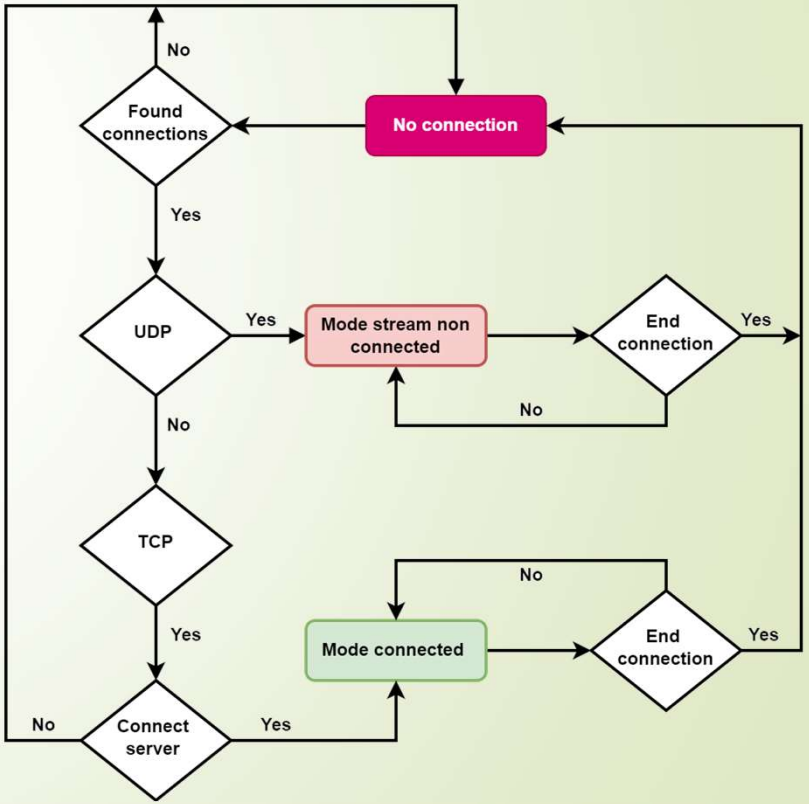
- A Typical HTTP-Session
- Requests and Responses
- Content Negotiation
- Access Control/Password-Protected Pages
- Caching (Proxies)
- State Management
- Authorization

I.1

Network Programming

Interfaces to Internet-Services - Sockets

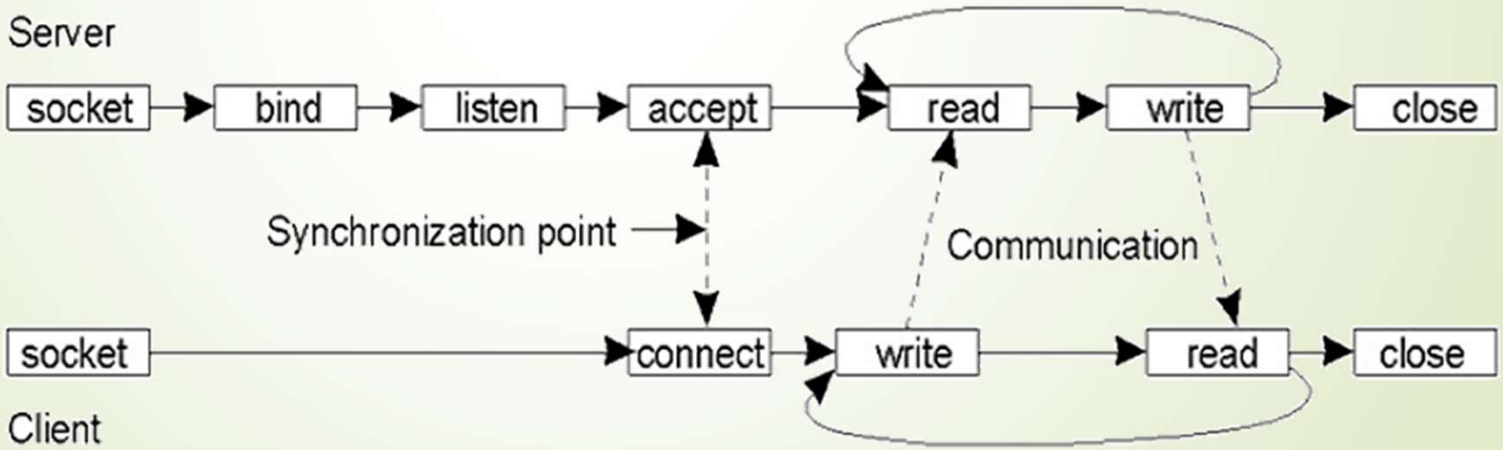
- an application programming interface (API) used for inter-process communication (IPC)
- socket is a network communication endpoint, obtained through a system call
- can use either TCP or UPD



Network Programming

Interfaces to Internet-Services - Sockets

- an application programming interface (API) used for inter-process communication (IPC)
- socket is a network communication endpoint, obtained through a system call



Network Programming

Interfaces to Internet-Services - Sockets

- Python's socket module provides an interface to the Berkeley sockets API
- socket/address families
  - socket.AF_UNIX address of the socket is a path on the filesystem
  - socket.AF_INET address of the socket is a pair (IPv4-host, port)
  - socket.AF_INET6 address of the socket is a pair (IPv6-host, port, flowinfo, scope_id)
  - ...
- creating a socket using a given address family, socket type and protocol number

```
sock = socket.socket(
           socket.AF_INET,
           socket.SOCK_STREAM | socket.SOCK_NONBLOCK)
```

I.1

Network Programming

Interfaces to Internet-Services - Sockets

- example

```python
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as serverSock:
    serverSock.bind((HOST, PORT))
    print("Listening ...")
    serverSock.listen()
    conn, addr = serverSock.accept()
    with conn:
        print(f"Connected by {addr}")
        while True:
            data = conn.recv(1024)
            if not data:
                break
            print(f"Received {len(data)} bytes")
            conn.sendall(data)
```

```python
with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
    s.connect((HOST, PORT))
    s.sendall(b"Hello, world")
    data = s.recv(1024)

print(f"Received {data!r}")
```

Network Programming

Interfaces to Internet-Services - Sockets

- improvements
  - generally: dealing with messages longer than the buffer
    - client `sendall` ok
    - server `recv` returns number of received bytes
  - server: handling multiple connections
    - concurrent solution, e.g. threads
    - controlled, multiplexed solution
      - `selectors`
        high-level I/O multiplexing, used to wait for I/O readiness notification on multiple objects
      - `select`
        wait until some registered objects become ready, or a timeout expires
        returns a list of (key, events)-tuples

Network Programming

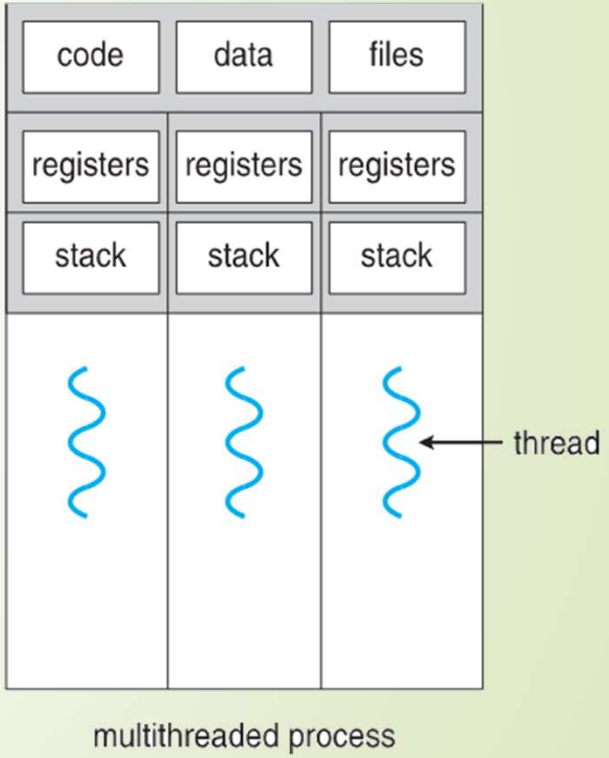
Interfaces to Internet-Services - Sockets

- `sel.select(timeout=None)` blocks until there are sockets ready for I/O
  - returns a list of tuples, one for each socket
  - each tuple contains a key and a mask
  - key is a `SelectorKey namedtuple` that contains
    - a `fileobj` attribute, i.e. the socket
    - a `data` attribute
      - if none, then event is related to listening socket and a new socket has to be created for communication
      - otherwise, then event is related to a client's socket
  - mask is an event mask of the operations that are ready

Network Programming

Exploiting Parallelism – Threads

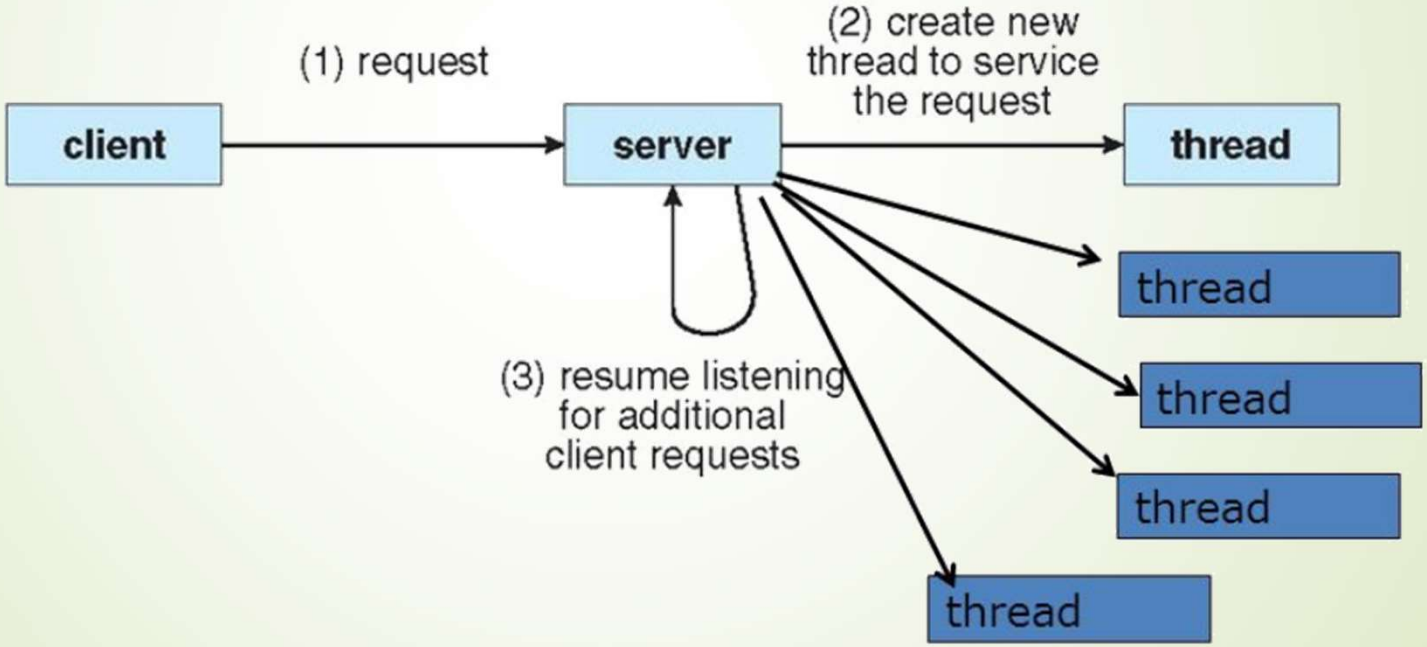
- a component of the process
- the smallest sequence of instructions that can be managed by the scheduler
- multiple threads can exist within one process, executing concurrently and sharing resources such as memory



multithreaded process

Network Programming

Exploiting Parallelism – Threads



Network Programming

Threads in Python

- Python standard library provides `threading`
- start a separate thread

```
ath = threading.Thread(target=thread_function, args=(1,))
ath.start()
```

- waiting for a thread

```
ath.join()
```