Network Programming

1. Computer Networks
   - transmission media and network topologies
   - client/server architecture
   - layers, protocols, and sockets
2. Network Programming
   - a simple client/server interaction
   - the module socket in Python
   - a game: player/dealer communication

MCS 260 Lecture 39
Introduction to Computer Science
Jan Verschelde, 28 July 2023

Network Programming

1. Computer Networks
   - transmission media and network topologies
   - client/server architecture
   - layers, protocols, and sockets

2. Network Programming
   - a simple client/server interaction
   - the module socket in Python
   - a game: player/dealer communication

Transmission Media

types of connections

The unit for transmission speed is `bps`, bits per second.

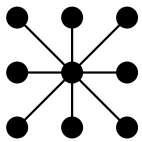
Four categories of media used for data transmission:

1. **twisted pair wire:** pair of copper wires used for telephone communications. Transmission speeds vary from 2,400 to 33,600 `bps`.
2. **coaxial cable:** for many telephone lines or television signal. Data transmission over short distances guarantees speeds up to 10^7 `bps`.
3. **optical fibre:** signals transmitted via light-emitting diodes encode bits as presence or absence of light. Transmission speeds reach up to 10^9 `bps`.
4. **electromagnetic waves:** wireless over short distances or via satellites across long distances.

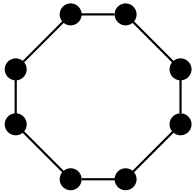
Network Topologies

structure of computer networks

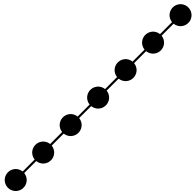
Three most common regular topologies:



a *star network*: one central node is connected to all other nodes



a *ring network*: nodes form a closed circuit, messages circle around the ring of nodes



a *bus network*: all nodes on a single bus, used in the Von Neumann architecture

Network Programming

1. Computer Networks
   - transmission media and network topologies
   - client/server architecture
   - layers, protocols, and sockets

2. Network Programming
   - a simple client/server interaction
   - the module socket in Python
   - a game: player/dealer communication

Client/Server Architecture

for software and hardware

Client/server architecture defines the communication between two computers: one is the server and the other acts as the client.

A client places a request or order to a server.
The server processes the request.

The client does not need to know how the server processes the requests.

We distinguish between software and hardware client/server architectures:

- web and database servers offer software services;
- file and print servers offer hardware services.

Network Programming

1. Computer Networks
   - transmission media and network topologies
   - client/server architecture
   - layers, protocols, and sockets

2. Network Programming
   - a simple client/server interaction
   - the module socket in Python
   - a game: player/dealer communication

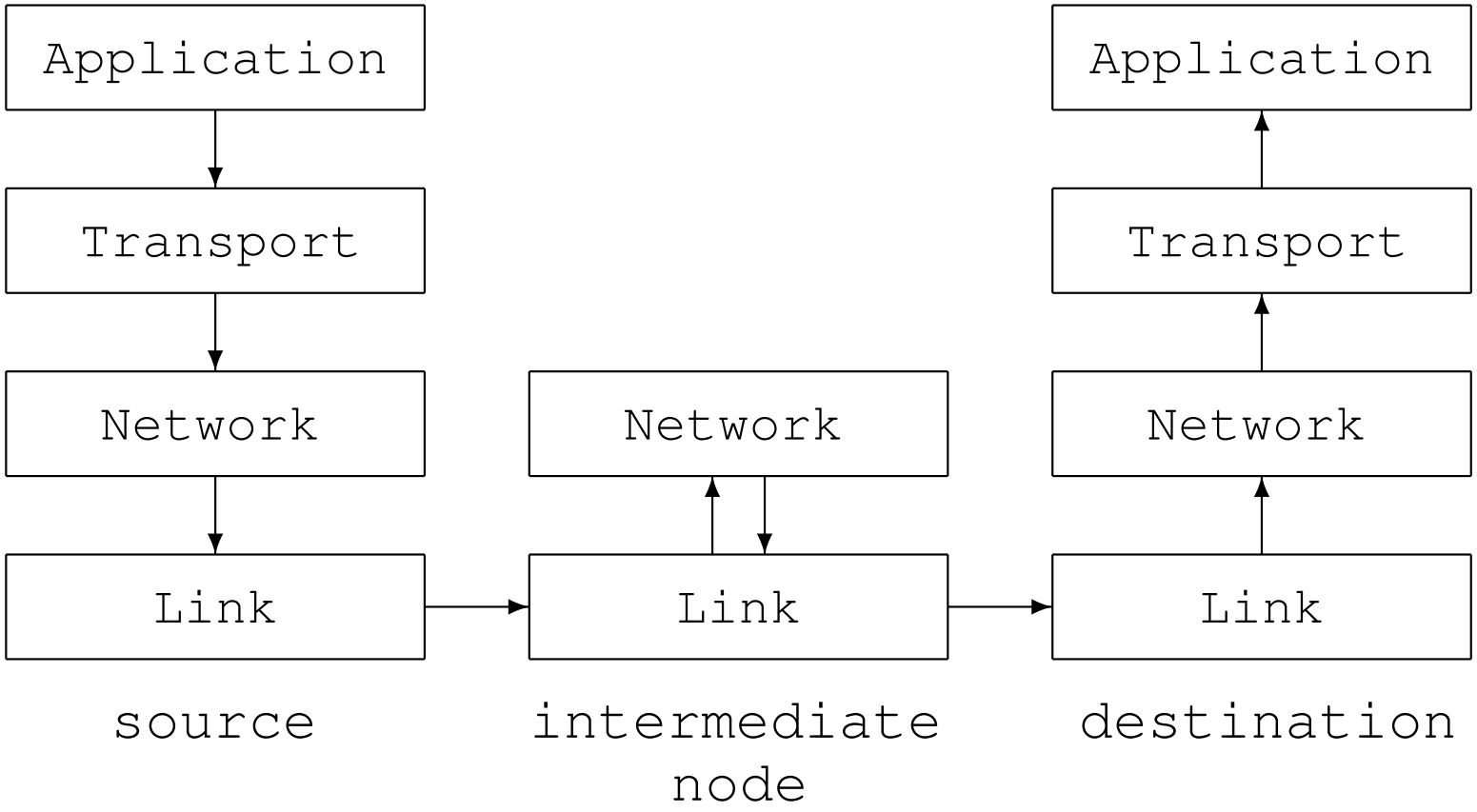
Four Layers in Internet Communication

application, transport, network, and link

1. The ***application layer*** consists of client and server software. The application prepares the message and defines the destination address.
2. The ***transport layer*** formats the message by chopping it into packets attaching a sequence number and destination address to each packet. When receiving, packets are collected and reassembled into a message.
3. The ***network layer*** determines the (intermediate) address for each packet. This layer detects when a packet has reached its final destination.
4. The ***link layer*** is responsible for the actual transmission of packets.

Following a Message

through all four layers



Network Protocols

TCP and UDP

Network protocols are rules for network communication.

We consider two types of protocols:

TCP Transmission Control Protocol
First a message is sent that data is coming.
Only after the receiver acknowledges this message will the sender send the data.
All successful transmissions are confirmed, and retransmissions are acknowledged.

UDP User Datagram Protocol
Unlike TCP, no connection is established prior to sending data. The sender just carries on after sending a message.

TCP is connection oriented, UDP is connectionless.
TCP is more reliable whereas UDP is more streamlined.

Sockets

Sockets are objects programs use to connect.

Sockets were introduced in 1981 in BSD Unix.
Originally used for communication between processes,
i.e.: between two programs on same computer.

Sockets support communication across platforms,
independent of the operating system.

IP stands for Internet Protocol.

In addition to the IP address of the computer,
both server and client must use the same *port*.
A port is a 16-bit integer, some are reserved for particular protocols.
Any port between 1,024 and 65,535 is free.

Sockets support both TCP and UDP.

Analogy with Telephone Exchange

client/server communication with sockets

Analogy between a telephone exchange and sockets:

1. dial company on 1-312-666-9000
   connect to IP address 127.0.0.1
2. call answered by reception
   connection established to remote host
3. ask for computer center
   route using specified port (8732)
4. call answered by computer center
   server handles request from client
5. hang up the phone
   close sockets

Network Programming

1. Computer Networks
   - transmission media and network topologies
   - client/server architecture
   - layers, protocols, and sockets

2. Network Programming
   - a simple client/server interaction
   - the module socket in Python
   - a game: player/dealer communication

a simple client/server interaction

Running `tcp_server.py`:

```
asterix:Lec39 jan$ python tcp_server.py
server waits for connection
```

The client `tcp_client.py` runs in another terminal.

connecting client and server

Running `tcp_client.py` in another terminal:

Lec39 — Python — 63×7

```
asterix:Lec39 jan$ python tcp_client.py
client is connected
Give message :
```

Running `tcp_server.py` in one terminal:

Lec39 — Python — 62×7

```
asterix:Lec39 jan$ python tcp_server.py
server waits for connection
server accepted connection request from  ('127.0.0.1', 49154)
server waits for data
```

passing data from client to server

Running `tcp_client.py` in another terminal:

```
asterix:Lec39 jan$ python tcp_client.py
client is connected
Give message : Hello there!
asterix:Lec39 jan$
```

Running `tcp_server.py` in one terminal:

```
asterix:Lec39 jan$ python tcp_server.py
server waits for connection
server accepted connection request from  ('127.0.0.1', 49154)
server waits for data
server received "Hello there!"
asterix:Lec39 jan$
```

Network Programming

1. Computer Networks
   - transmission media and network topologies
   - client/server architecture
   - layers, protocols, and sockets
2. Network Programming
   - a simple client/server interaction
   - the module socket in Python
   - a game: player/dealer communication

sockets in Python: the socket module

The socket module exports the class `socket()` which returns an object representing a socket.

The first argument of `socket` is the Address Family (AF):

- `AF_UNIX` : for UNIX sockets;
- `AF_INET` : most commonly used for internet

`AF_INET` supports both TCP and UDP, given respectively by `SOCK_STREAM` and `SOCK_DGRAM` as second argument of `socket()`.

For example:

```
from socket import socket as Socket
from socket import AF_INET, SOCK_STREAM
sock = Socket(AF_INET, SOCK_STREAM)
```

methods on socket objects

Most commonly used methods:

method	description
`accept()`	accepts connection and returns new socket for passing data
`bind()`	binds a socket to an address
`close()`	closes the socket
`connect(a)`	connects to address `a`
`listen(c)`	listen for `c` connections
`shutdown(flag)`	shut down for reading, writing, or both
`recv(b)`	receives data in buffer of size `b`
`send(d)`	sends data in `d`

Network Programming

1. Computer Networks
   - transmission media and network topologies
   - client/server architecture
   - layers, protocols, and sockets

2. Network Programming
   - a simple client/server interaction
   - the module socket in Python
   - a game: player/dealer communication

A Little Game

player/dealer communication

The dealer generates a secret number.

The player must guess the number.

After each guess, the dealer sends feedback:
too low, too high, or found the secret

The game terminates when the secret is found.

The player is the client, the dealer is the server

Exercises

1. Extend the client/server interaction to simulate a password dialogue. After receiving data from a client, the server returns `access granted` or `access denied` depending on whether the received data matches the password.
2. Describe how you would implement the game of rock, paper, and scissors in a client/server manner.
3. Redefine the quiz of Project Three as a client/server interaction. The server generates, asks the questions, and gives feedback on the answers submitted by the client.
4. Write an implementation of the server in the previous exercise.
5. Write an implementation of the client in the previous exercise.