Inter-Process Communication: Network Programming using Sockets

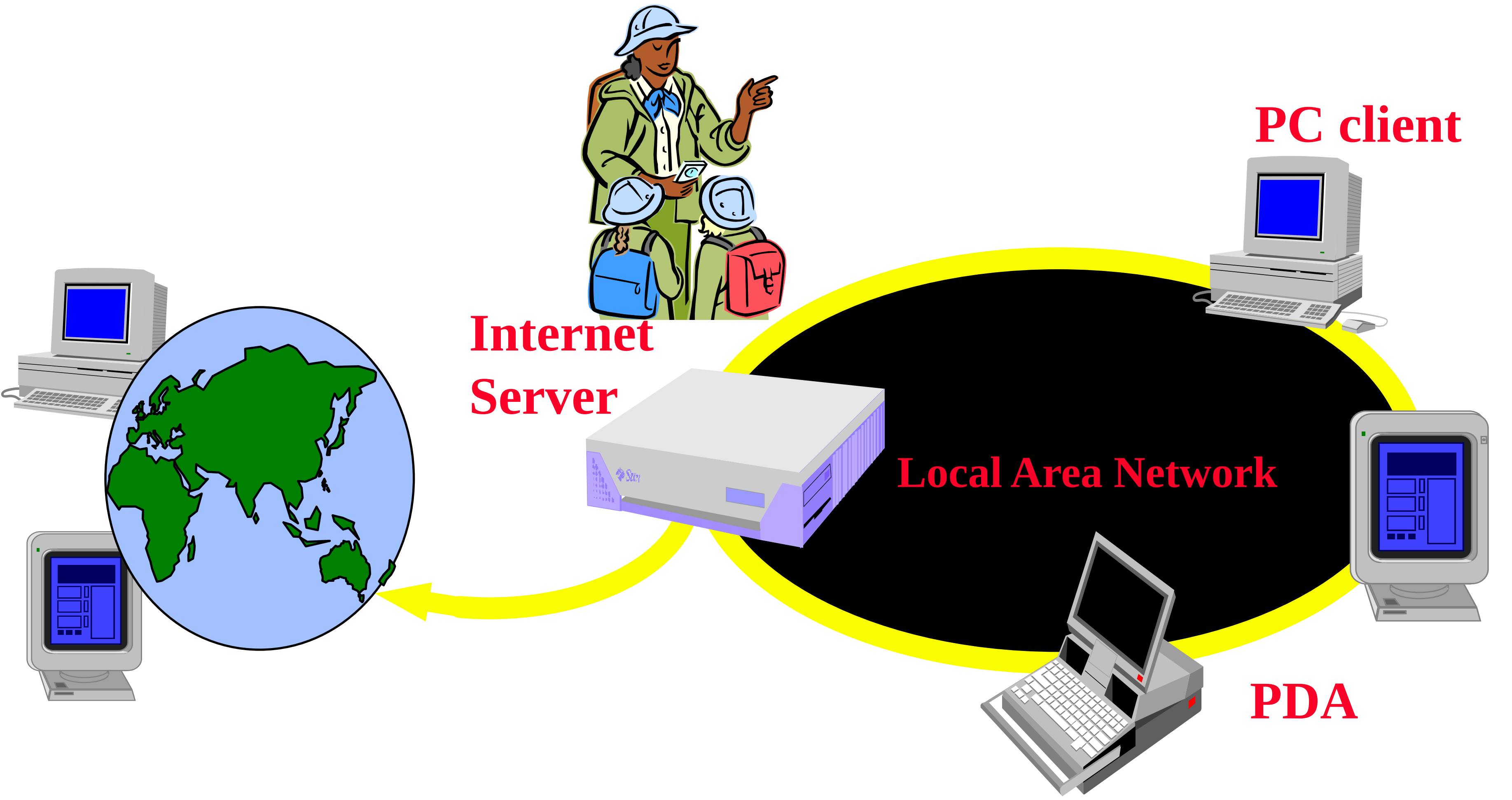
Agenda

- Introduction
- Networking Basics
- Understanding Ports and Sockets
- Java Sockets
    - Implementing a Server
    - Implementing a Client
- Sample Examples
- Conclusions

Introduction

- Internet and WWW have emerged as global ubiquitous media for communication and are changing the way we conduct science, engineering, and commerce
- They are also changing the way we learn, live, enjoy, communicate, interact, engage, etc. It appears like the modern life activities are getting completely centered around the Internet

Internet Applications Serving Local and Remote Users

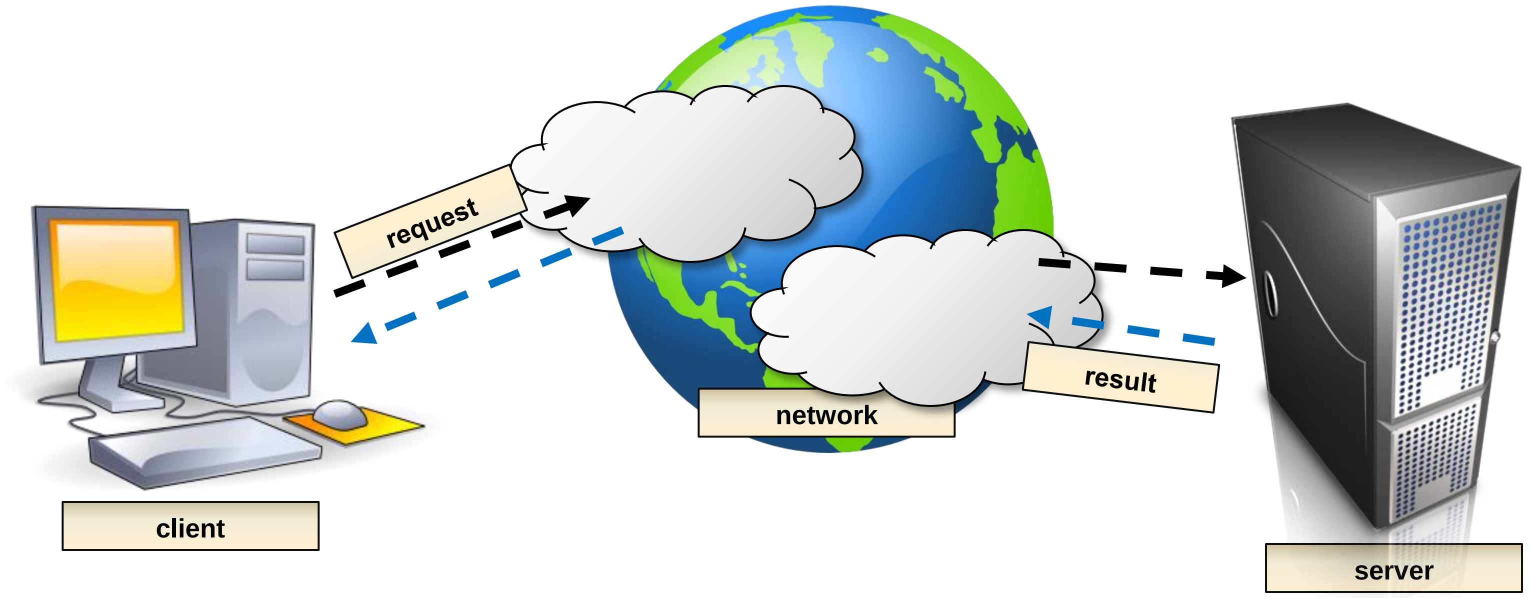


Increasing Demand for Internet Applications

- To take advantage of opportunities presented by the Internet, businesses are continuously seeking new and innovative ways and means for offering their services via the Internet
- This created a huge demand for software designers with skills to create new Internet-enabled applications or migrate existing/legacy applications to the Internet platform
- Object-oriented Java technologies—Sockets, threads, RMI, clustering, Web services—have emerged as leading solutions for creating portable, efficient, and maintainable large and complex Internet applications

Elements of Client-Server Computing/Communication

a client, a server, and network

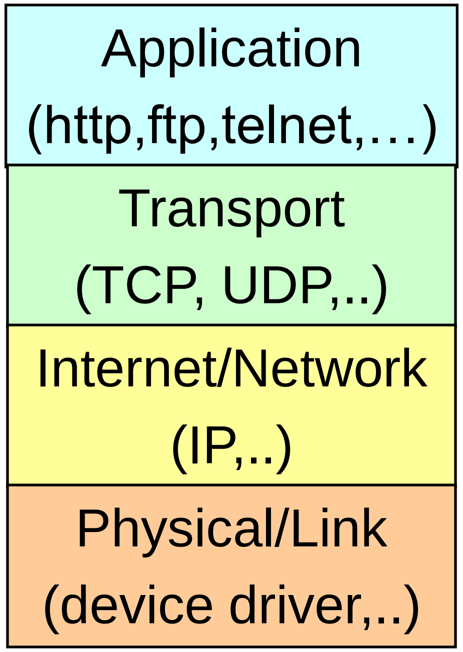


- Processes follow protocols that define a set of rules that must be observed by participants:
  - How the data exchange is encoded?
  - How events (sending, receiving) are synchronized (ordered) so that participants can send and receive data in a coordinated manner?
- In face-to-face communication, humans beings follow unspoken protocols based on eye contact, body language, gesture.

Networking Basics

- Physical/Link Layer
  - Functionalities for transmission of signals representing a stream of data from one computer to another
- Internet/Network Layer
  - IP (Internet Protocols) – a packet of data to be addressed to a remote computer and delivered
- Transport Layer
  - Functionalities for delivering data packets to a specific process on a remote computer
  - TCP (Transmission Control Protocol)
  - UDP (User Datagram Protocol)
  - Programming Interface:
    - Sockets
- Applications Layer
  - Message exchange between standard or user applications:
    - HTTP, FTP, Telnet, **Skype,…**

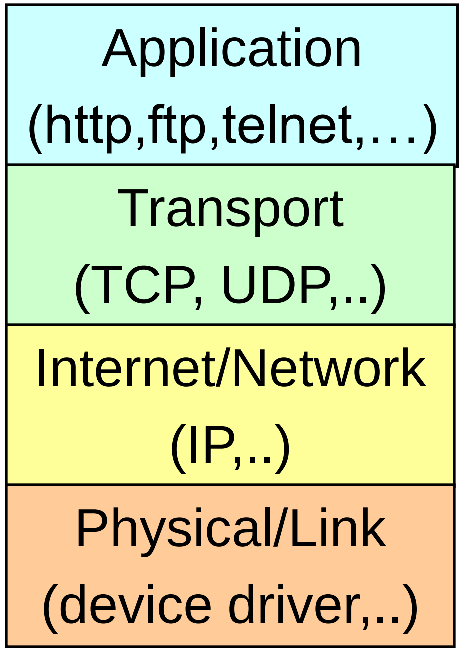
- TCP/IP Stack



Networking Basics

- TCP (Transmission Control Protocol) is a connection-oriented communication protocol that provides a reliable flow of data between two computers.
- Analogy: Speaking on Phone
- Example applications:
  - HTTP, FTP, Telnet
  - **Skype** uses **TCP** for call signalling, and both **UDP** and **TCP** for transporting media traffic.

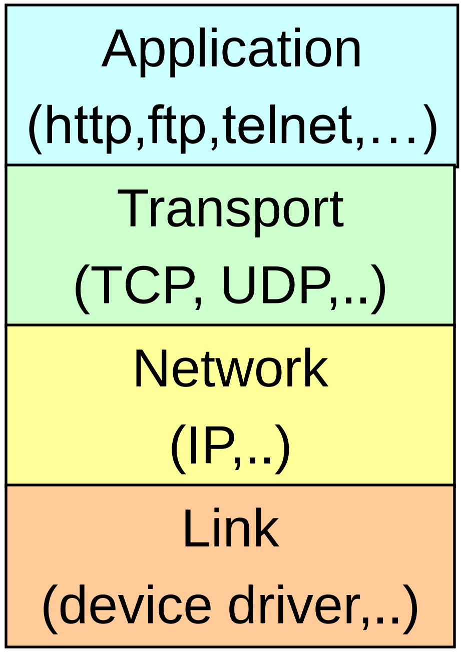
- TCP/IP Stack


Application (http,ftp,telnet,…)
Transport (TCP, UDP,..)
Internet/Network (IP,..)
Physical/Link (device driver,..)

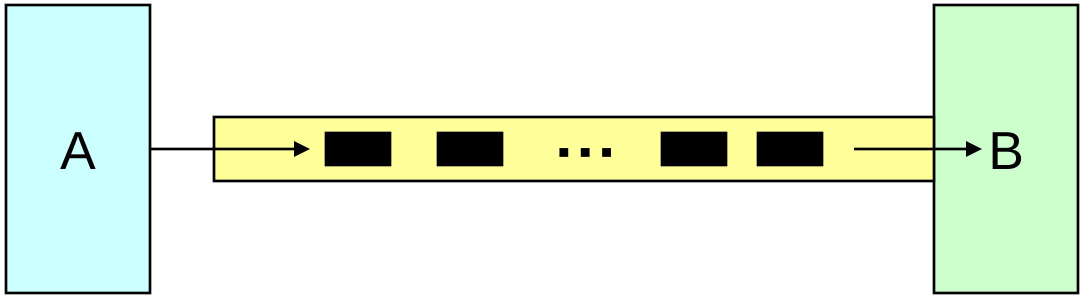
Networking Basics

- UDP (User Datagram Protocol) is a connectionless communication protocol that sends independent packets of data, called *datagrams*, from one computer to another with no guarantees about arrival or order of arrival
- Similar to sending multiple emails/letters to friends, each containing part of a message.
- Example applications:
  - Clock server
  - Ping
  - Live streaming (event/sports broadcasting)

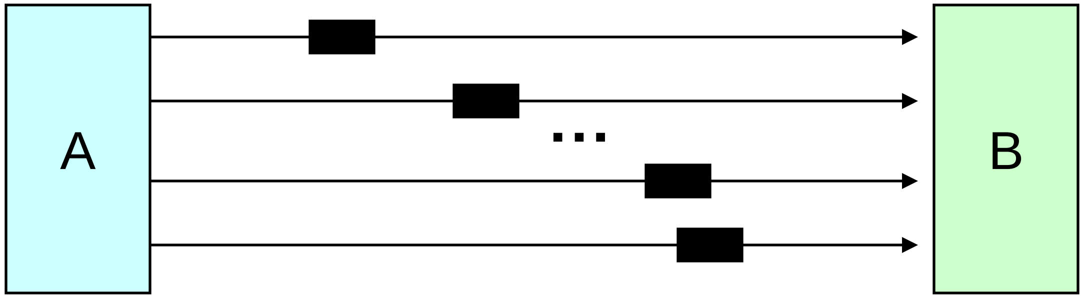
- TCP/IP Stack



TCP Vs UDP Communication



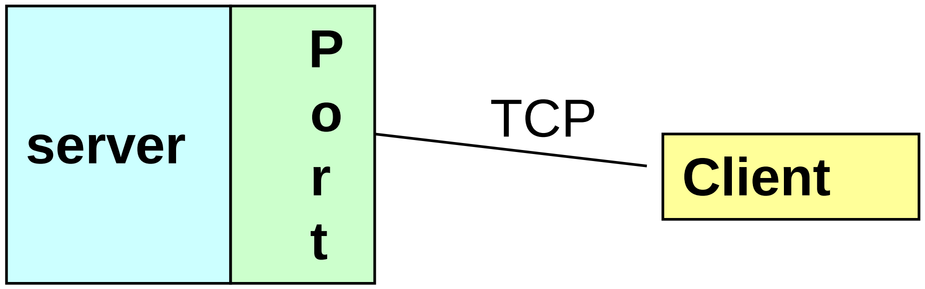
- Connection-Oriented Communication

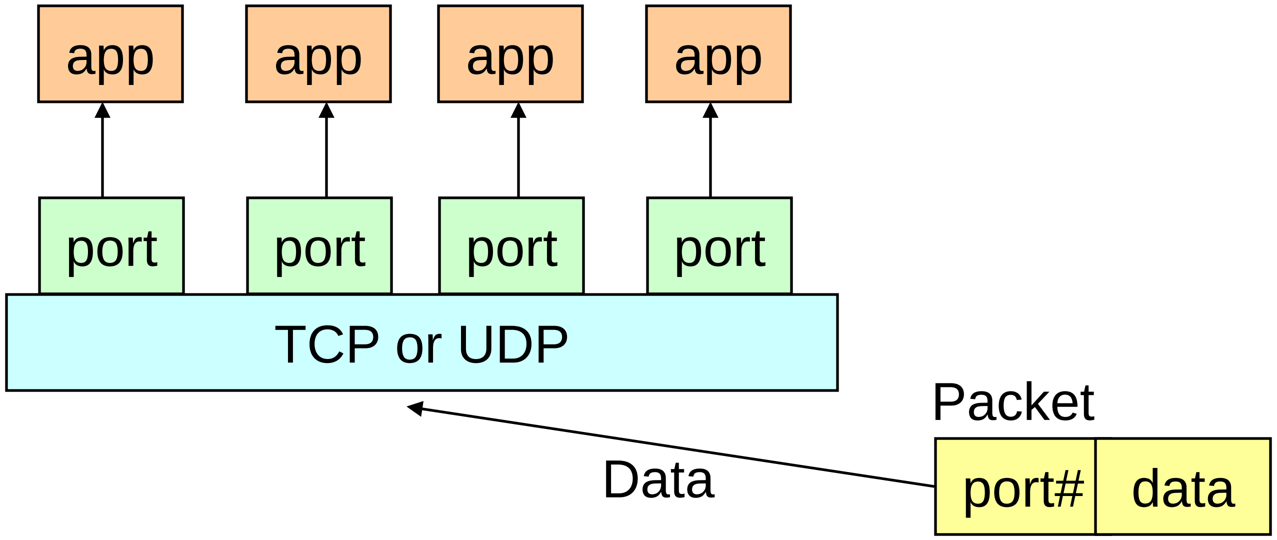


- Connectionless Communication

Understanding Ports

- The TCP and UDP protocols use *ports* to map incoming data to a particular *process* running on a computer.





Understanding Ports

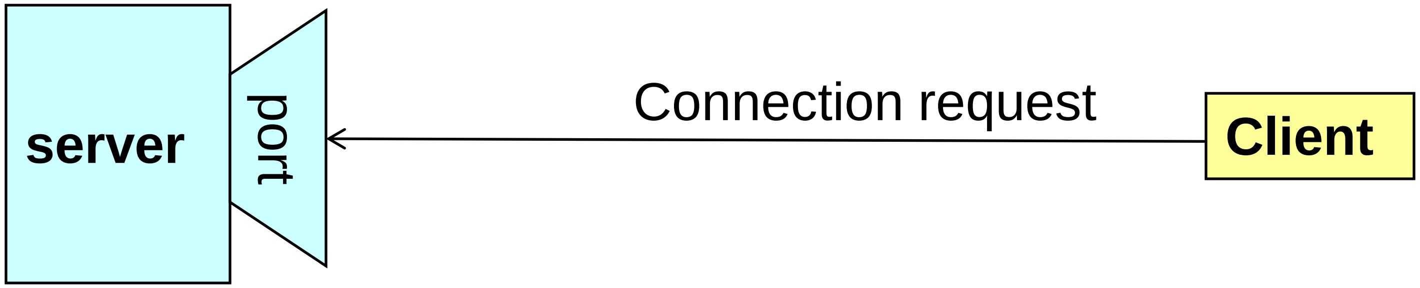
- Port is represented by a positive (16-bit) integer value
- Some ports have been reserved to support common/well known services:
  - ftp   21/tcp
  - telnet 23/tcp
  - smtp 25/tcp
  - login 513/tcp
- User-level processes/services generally use port number value >= 1024

Sockets

- Sockets provide an interface for programming networks at the transport layer
- Network communication using Sockets is very much similar to performing file I/O
  - In fact, socket handle is treated like file handle.
  - The streams used in file I/O operation are also applicable to socket-based I/O
- Socket-based communication is programming language independent.
  - That means, a socket program written in Java language can also communicate to a program written in Java or non-Java socket program

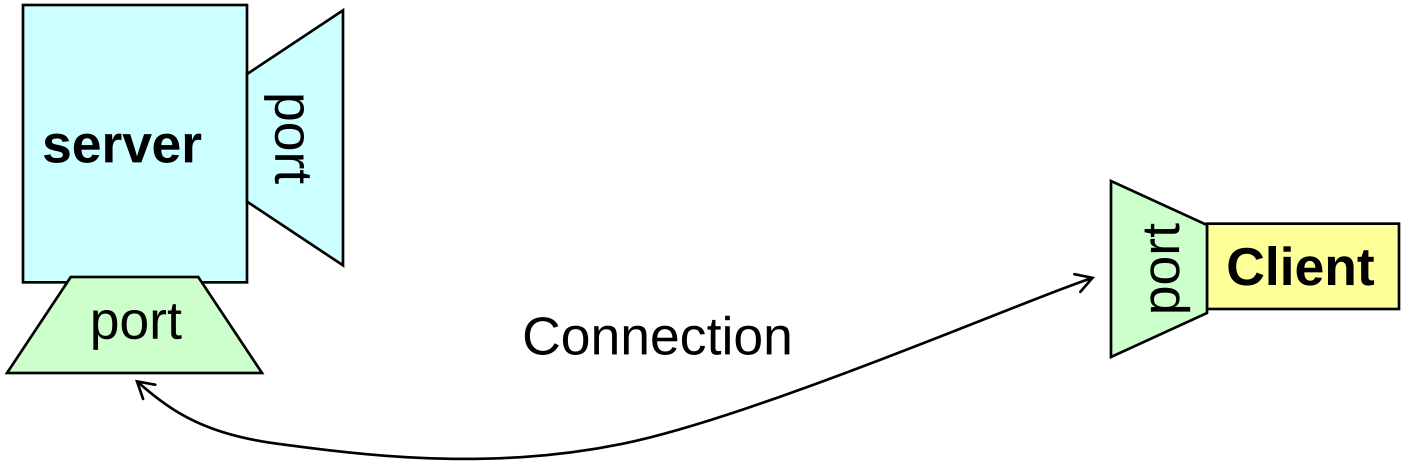
Socket Communication

- A server (program) runs on a specific computer and has a socket that is bound to a specific port. The server waits and listens to the socket for a client to make a connection request.



Socket Communication

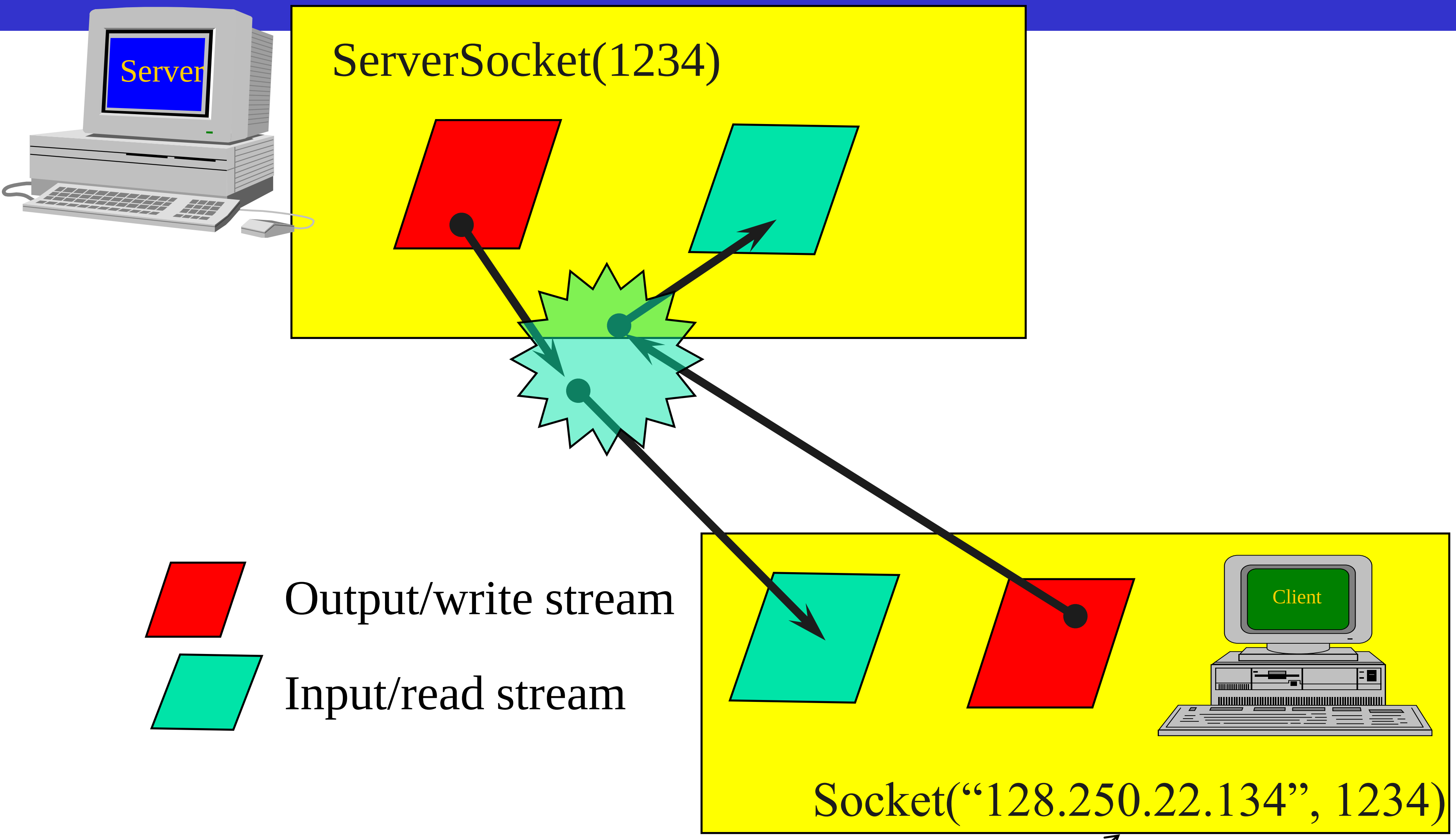
- If everything goes well, the server accepts the connection. Upon acceptance, the server gets a new socket bounds to a different port. It needs a new socket (consequently a different port number) so that it can continue to listen to the original socket for connection requests while serving the connected client.



Sockets and Java Socket Classes

- A socket is an endpoint of a two-way communication link between two programs running on the network.
- A socket is bound to a port number so that the TCP layer can identify the application that data destined to be sent.
- Java’s .net package provides two classes:
  - Socket – for implementing a client
  - ServerSocket – for implementing a server

Java Sockets



🟥 Output/write stream

🟩 Input/read stream

It can be host_name like "jarrett.cis.unimelb.edu.au"

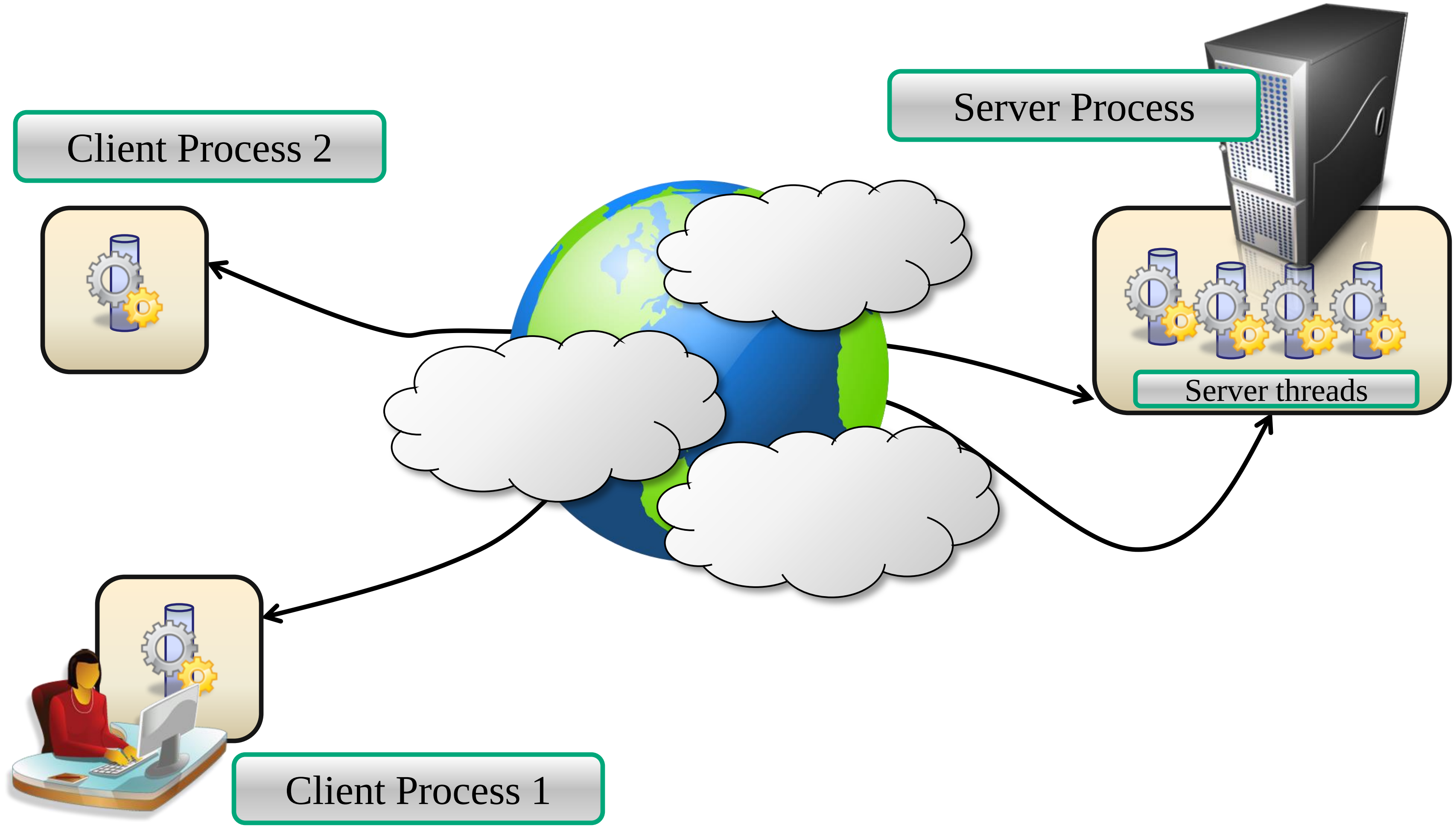
Java API for UDP Programming

- Java API provides datagram communication by means of two classes
  - DatagramPacket

    - | Msg | length | Host | serverPort |

  - DatagramSocket

Multithreaded Server: For Serving Multiple Clients Concurrently



Summary

- Programming client/server applications in Java is fun and challenging
- Programming socket programming in Java is much easier than doing it in other languages such as C
- TCP for Connection-oriented communication, more reliable, flow control
- UDP for connection-less communication
- Keywords:
  - Clients, servers, TCP/IP, port number, sockets, Java sockets