Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Interprocess Communication (IPC)

Unix pipes (|)

Sockets

Hes·so
Haute Ecole Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

What are sockets (1/2)

- Is an application programming interface (API) for Internet and Unix domain socket
- Proposed by the uni. of California in Berkeley
- Known as Berkeley sockets
- Used for IPC
- Libraries exists for using sockets in different programming languages

What are sockets (2/2)

- It's an **abstract representation** (handle) for the **local endpoint** of a (network) communication path
  - Often represented as a file descriptor (file handler) in the Unix philosophy
- Provides a common interface for input and output **streams** of data
  - De facto standard: **POSIX sockets**

Types of sockets

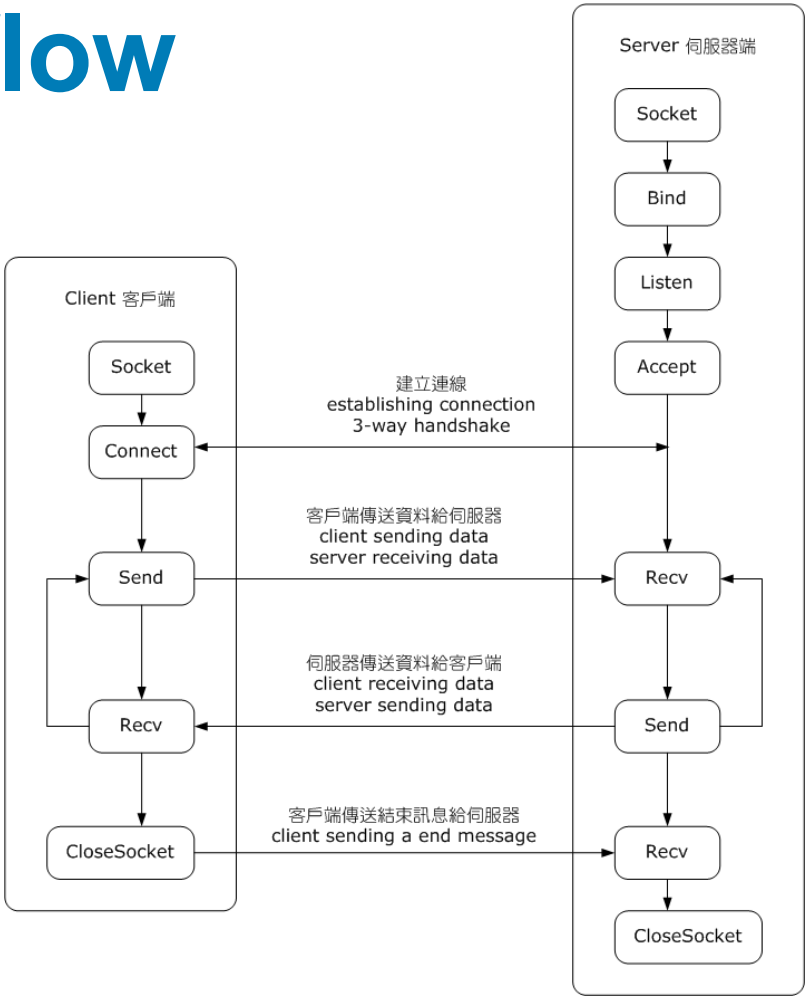
1. TCP sockets (most common)
   - Reliable ✅
   - Ordered ✅
   - Error-checked ✅
- Used for:
   - websites (HTTP/HTTPS)
   - APIs
   - databases

2. UDP sockets
   - Faster ⚡
   - No guarantee of delivery ❌
   - No ordering ❌
- Used for:
   - video streaming
   - online games
   - VoIP

(TCP) Socket flow diagram

Server
- waits for incoming connections
- listens on a specific port
- responds to requests

Client
- initiates the connection
- sends requests
- receives responses



https://en.wikipedia.org/wiki/Berkeley_sockets

Example – Server socket

```
// socket creation
int PORT = 8080;
ServerSocket serverSocket = new ServerSocket(PORT);
//Server launched, listening on port: <PORT>

// repeatedly wait for connections, and process
while (true) {
  // blocks until a client tries to connect
  Socket clientSocket = serverSocket.accept();
  System.err.println("New client connected");

  // open conversation flow
  BufferedReader in = new BufferedReader(new
    InputStreamReader(clientSocket.getInputStream()));
  BufferedWriter out = new BufferedWriter(new
    OutputStreamWriter(clientSocket.getOutputStream()));
```

```
    while ((s = in.readLine()) != null) {
      System.out.println(s);
      if (s.isEmpty()) // end detected
        break;
    }


    // now send on the output socket (out) the response
    out.write("HTTP/1.1 200 OK\r\n");
    ....
    out.write("<P>This is my test server page</P>");

    // closing of the sockets
    // Connection finished, closing sockets
    out.close();
    in.close();
    clientSocket.close();
  }
}
```

Example – Client socket

```
public static void main(String[] args) {
  String server = args[0];
  String path = args[1];
  int PORT = 80;

  try {
    // Connect to the server
    Socket socket = new Socket(server, PORT);

    // Create input and output streams to read from and
       write to the server
    PrintStream out = new
      PrintStream(socket.getOutputStream());
    BufferedReader in = new BufferedReader(new
      InputStreamReader(socket.getInputStream()));

    // Follow the HTTP protocol of GET <path> HTTP/1.0
       followed by an empty line
    out.println("GET " + path + " HTTP/1.0");
    out.println();

    // Read data until we finish reading the document
    String line = in.readLine();
    while (line != null) {
      System.out.println(line);
      line = in.readLine();
    }

    // Close our streams
    in.close();
    out.close();
    socket.close();
  } catch (Exception e) {
    e.printStackTrace();
  }
}
```

Key take-away messages

- POSIX sockets are used to exchange messages between processes
  - IPC
- Interface for input and output **streams** of data
- API / Library for sockets for easy programming
  - Assist for creating the socket flow
- Types of sockets: TCP and UDP

Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Annex
Exercises
Python mini-HTTP server

Hes·so
Haute École Spécialisée de Suisse occidentale
Fachhochschule Westschweiz

Exercise – Concurrent server

Exercise 24

- The shown HTTP server can only serve one client at time (one endless while-true loop doing one `accept()`)
- Change the code in such a manner that the server accepts multiple clients at the same time, using Threads serving the different clients

Python `http.server`

- In Python a minimalistic, out-of-the-box http server exists in the form of the `http.server` library
  - https://docs.python.org/3/library/http.server.html

- E.g., for testing purpose, `http.server` can be used to serve an existing directory as an http resource:

  - `python -m http.server --directory /tmp/ 8000`