3. Mrežno programiranje

Univerzitet u Novom Sadu
Fakultet tehničkih nauka
Web programiranje

Mrežno programiranje? Čemu i kako?

- Komunikacija preko mreže (očigledno)
- Klijent-server
- Protokoli: serijalizovani bajtovi na mreži, tekstualni ili binarni sadržaj u programu
- Komunikacija preko **socketa**

IP adresa i...

- IP adresa == uređaj na mreži
- 1 računar, 100 programa
- Kome je poruka namenjena?

… port

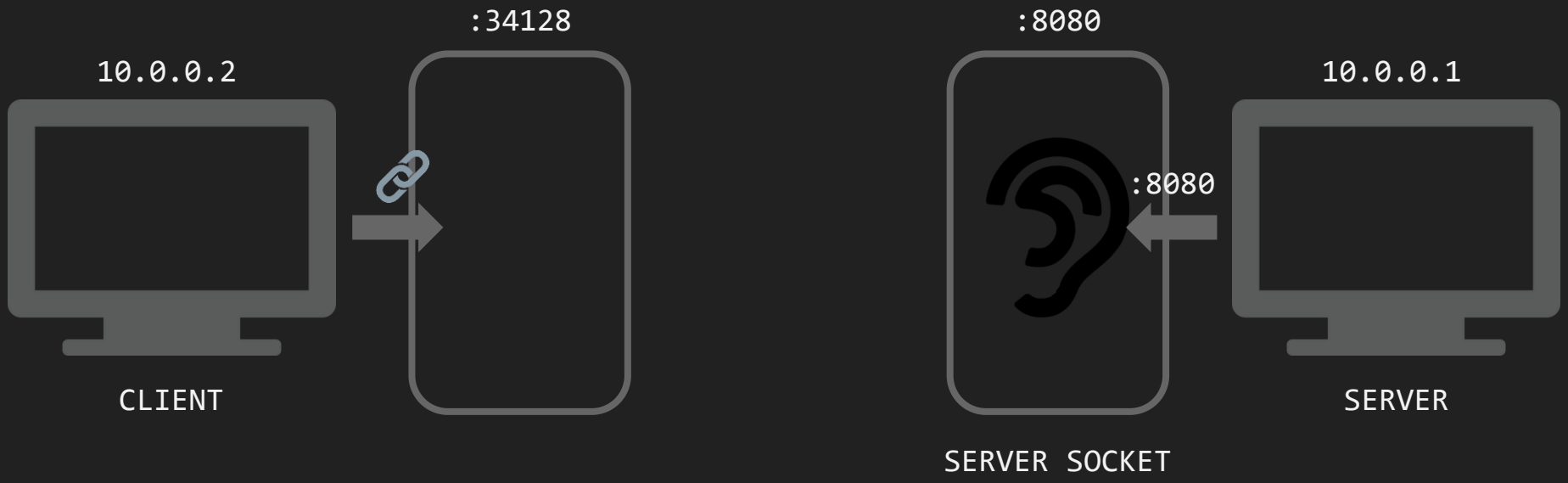
- Svaki program na mreži ima dodeljen broj
- Integer (0-65535)
- IP adresa su ulica i broj, port je broj stana

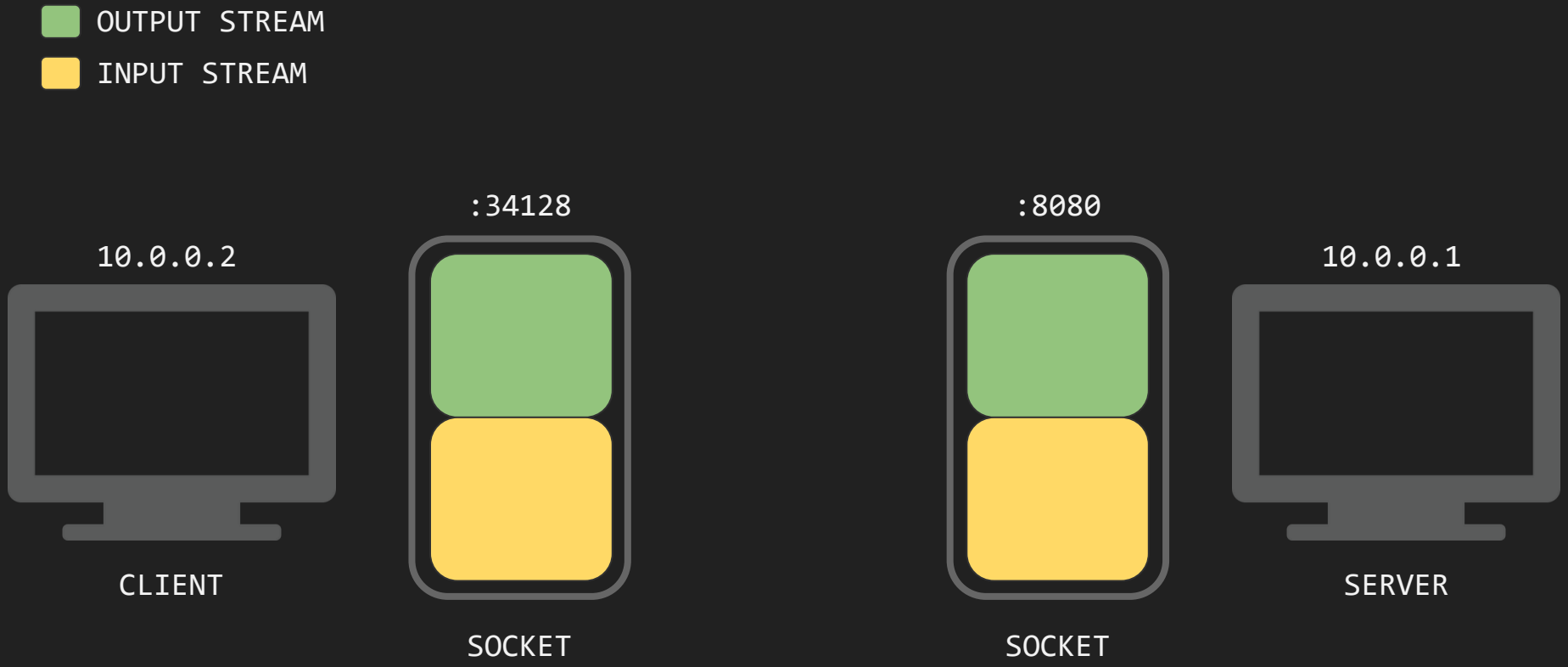
10.0.0.2

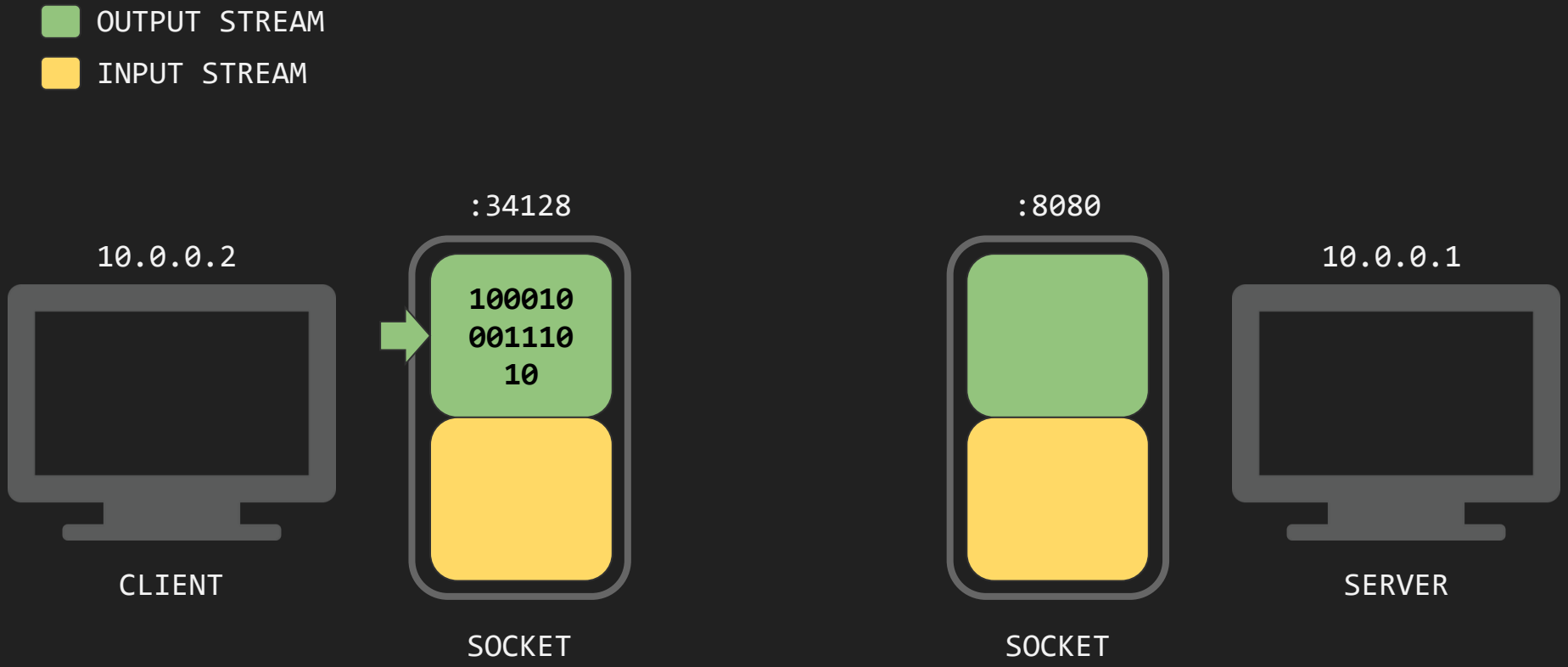
10.0.0.1

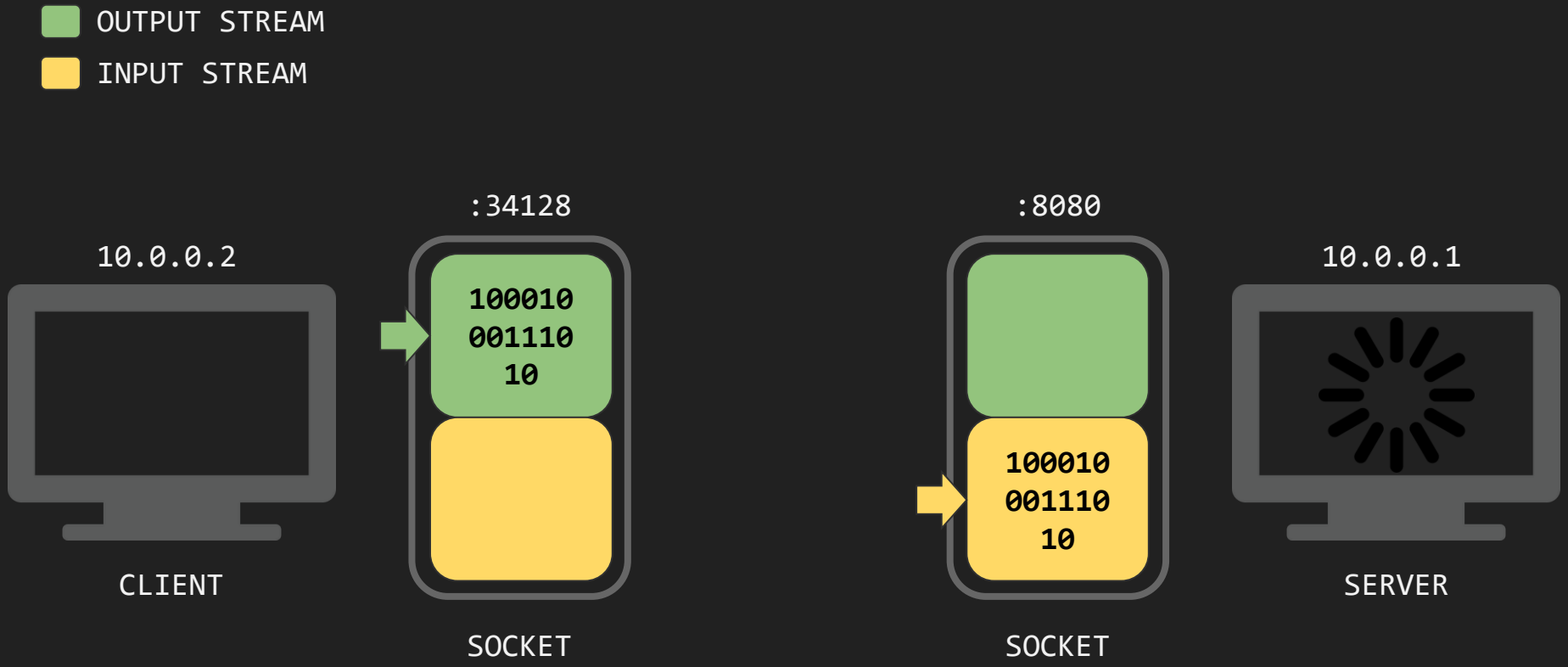
CLIENT

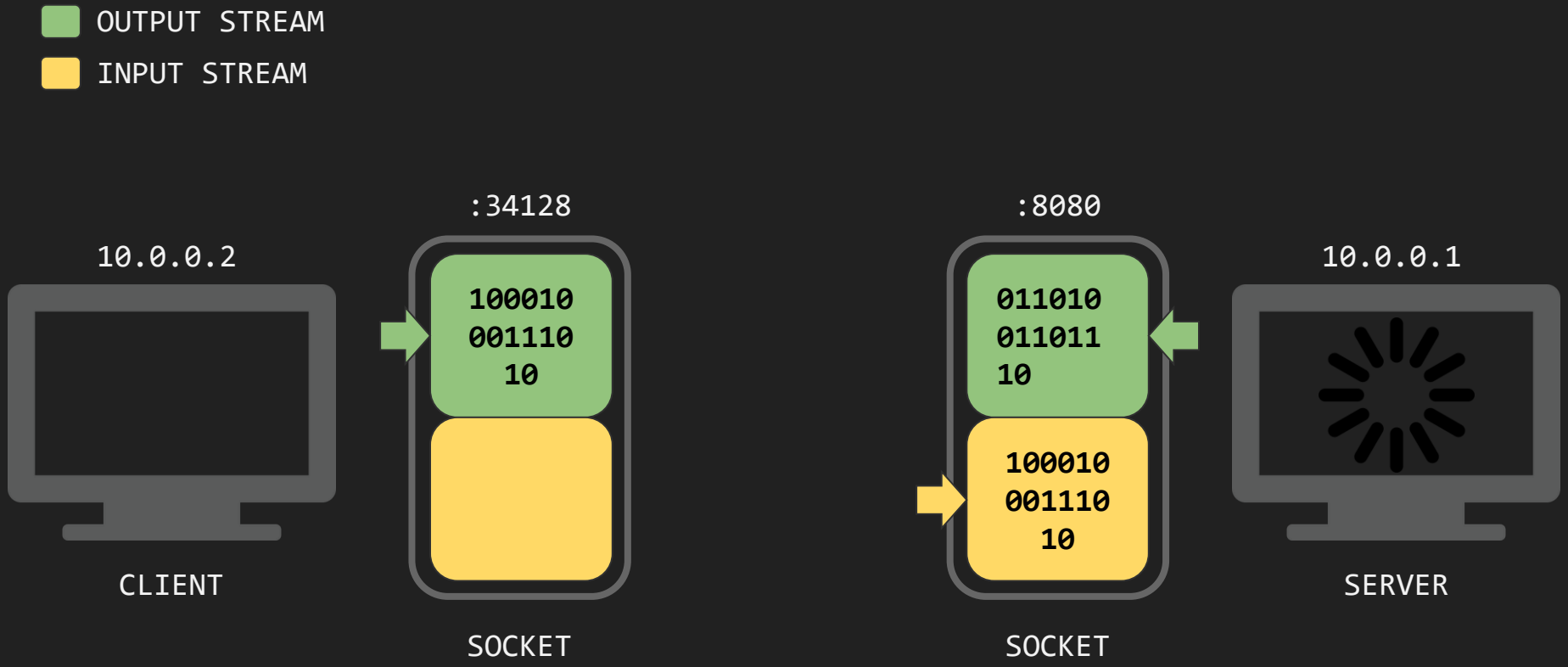
SERVER

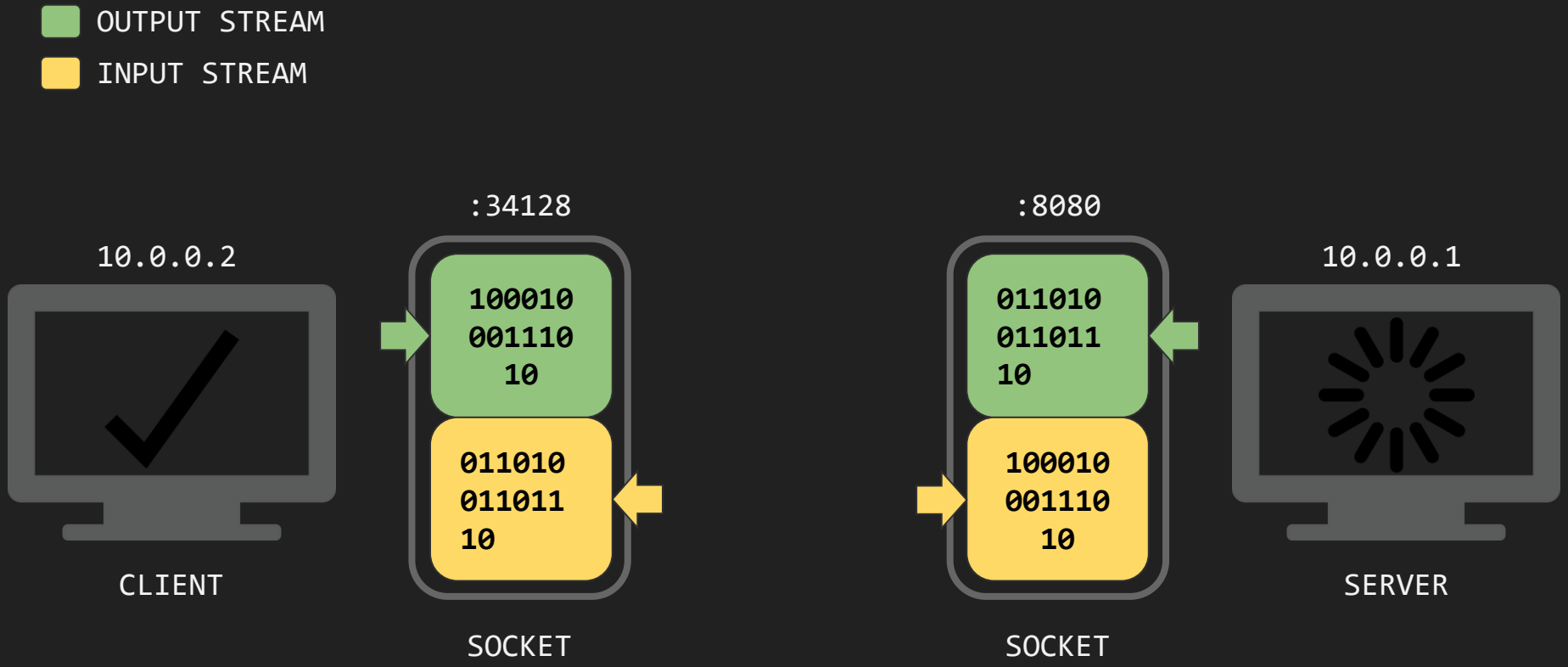

:34128
10.0.0.2
:8080
10.0.0.1
:8080
CLIENT
SERVER
SERVER SOCKET


OUTPUT STREAM
INPUT STREAM
:34128
10.0.0.2
:8080
10.0.0.1
CLIENT
SOCKET
SOCKET
SERVER


OUTPUT STREAM
INPUT STREAM
:34128
:8080
10.0.0.2
10.0.0.1
100010
001110
10
CLIENT
SOCKET
SOCKET
SERVER


OUTPUT STREAM
INPUT STREAM
:34128
10.0.0.2
100010
001110
10
:8080
10.0.0.1
100010
001110
10
CLIENT
SOCKET
SOCKET
SERVER


OUTPUT STREAM
INPUT STREAM
:34128
10.0.0.2
100010
001110
10
:8080
011010
011011
10
10.0.0.1
100010
001110
10
CLIENT
SOCKET
SOCKET
SERVER


OUTPUT STREAM
INPUT STREAM
:34128
:8080
10.0.0.2
10.0.0.1
100010
001110
10
011010
011011
10
011010
011011
10
100010
001110
10
CLIENT
SERVER
SOCKET
SOCKET

Mrežno programiranje u Javi

- Klasa **Socket**: komunikacija preko mreže
- Klasa **ServerSocket**: server
  - ServerSocket.accept()
- I klijenti i serveri koriste sockete
- 2 socketa == 1 komunikacija

Uspostavljanje konekcije i čitanje zahteva

- Client

```
InetAddress addr =
InetAddress.getByName("127.0.0.1");

// otvori socket prema drugom racunaru
Socket sock = new Socket(addr, TCP_PORT);

PrintWriter out = new PrintWriter(new
BufferedWriter(
new
OutputStreamWriter(socket.getOutputStream())),
true);
out.println(message);
```

- Server

```
ServerSocket ss = new ServerSocket(TCP_PORT);

while (true) {
        Socket socket = ss.accept();
        BufferedReader in = new
        BufferedReader(
        new InputStreamReader(socket.getInputStream()));

        // procitaj zahtev
        String request = in.readLine();
}
```

Slanje i čitanje odgovora

- Client

```
BufferedReader in = new
BufferedReader(new InputStreamReader(
socket.getInputStream()));
String response = in.readLine();
```

- Server

```
PrintWriter out = new PrintWriter(new
BufferedWriter(new
OutputStreamWriter(
socket.getOutputStream())), true);

out.println(message);
```

Čekanje...

- Blokirajuće vs neblokirajuće operacije
- Spori klijenti, duge operacije
- Neefikasno iskorišćenje resursa računara

Konkurentno programiranje

- **Niti (Threads)**
- Uvodimo nit po klijentu => da bismo izbegli čekanje!
- U Javi klasa **Thread** ili **Runnable** interfejs
- Programski kod niti se implementira **run** metodi, a nit se pokreće pozivom **start** metode

Niti niti neće dovoljne biti

- Niti nisu besplatne (1MB Linux)
- Puno niti == puno context switching-a
- Puno posla a ništa urađeno
- Puno otvorenih konekcija: thundering herd problem
- Thread pooling, reactor, proactor