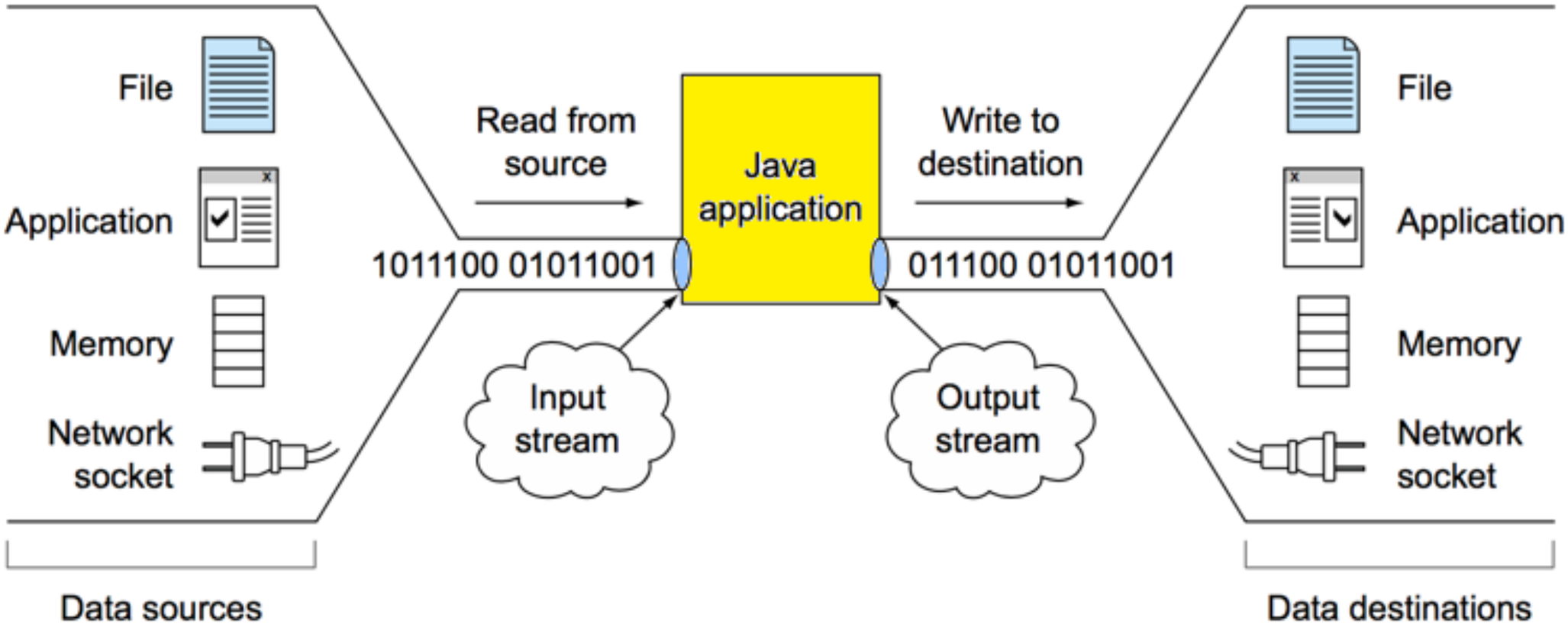
Java Network Sockets

Basic Java Network Programming

From I/O Streams to Sockets

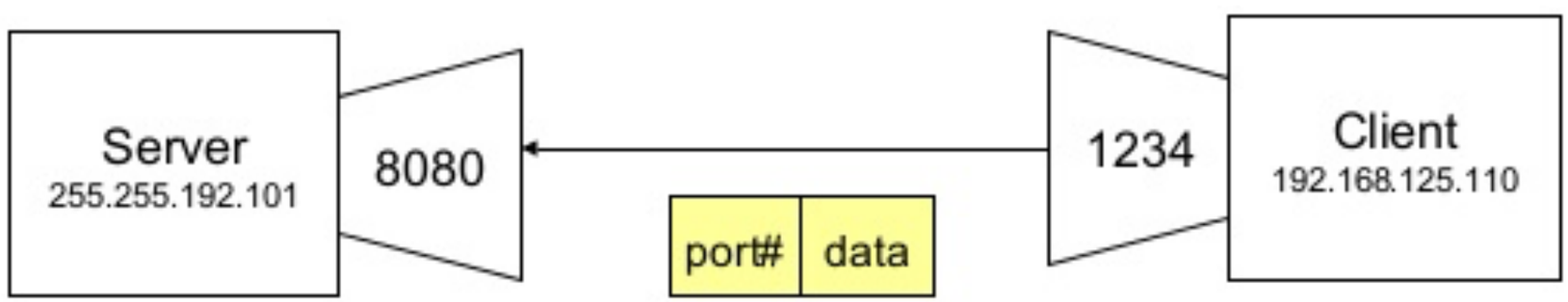
- A *stream* is a sequence of data.
- The stream can be either:
  - Input Stream: used for reading data from a source.
  - Output Stream: used for writing data to a destination.
- Java I/O stream abstracts a data source or data destination.

I/O Streams & Sockets



Network Sockets

- A socket is one end-point of a two-way network connection.
- A socket = IP + port
- Based on the mostly used Internet protocols, there are:
  - TCP/IP sockets
  - UDP sockets



Client Socket

- From a client perspective, connecting to a server starts with a `java.net.Socket` instance.

```
Socket socket = new Socket("192.168.1.17", 8080);
```

- The new Java I/O api (also known as Java NIO) offers also a `SocketChannel` alternative.

- Reading from or writing to a socket is as easy as working with input and output streams.

Client Socket (cont'd)

- Reading from a socket is done with the help of `InputStream`.

```
Socket socket = new Socket("192.168.1.17", 8080);
InputStream in = socket.getInputStream();

int data = in.read();
//... read more data...
in.close();
socket.close();
```

- As a client, you can close the connection if the server does not close it automatically.

Client Socket (cont'd)

- Writing to a socket is done with the help of `OutputStream`.

```
Socket socket = new Socket("192.168.1.17", 8080);
OuputStream out = socket.getOutputStream();

out.write("some data".getBytes());
out.flush();
out.close();
socket.close();
```

- As a client, you can close the connection if the server does not close it automatically.

Server Socket

- From a server perspective, listening to a socket for incoming client connections starts with a `java.net.ServerSocket` instance.

```
ServerSocket serverSocket = new ServerSocket(8080);
```

- The new Java I/O api (also known as Java NIO) offers also a `ServerSocketChannel` alternative.

- `serverSocket.accept()` accepts a new incoming connection and returns a `Socket` that can further be used for talking on that connection.

Server Socket (cont'd)

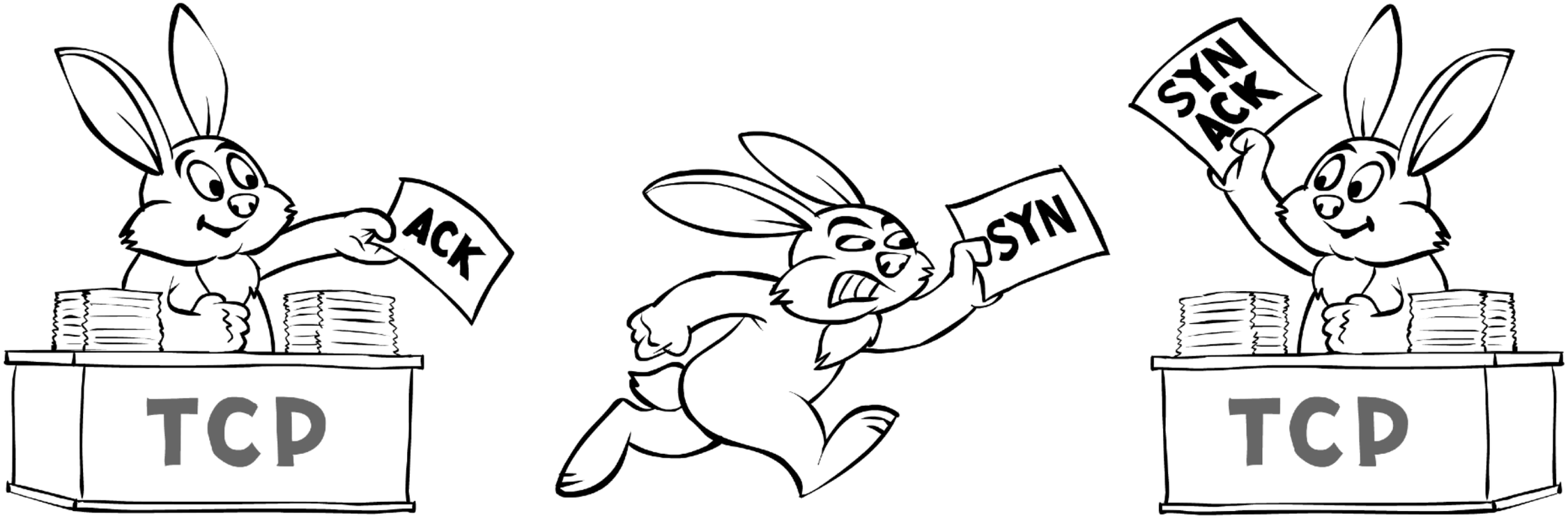
- Reading from a socket is done with the help of `InputStream`.

```
ServerSocket serverSocket = new ServerSocket(8080);
while (isRunning) {
    Socket clientSocket = serverSocket.accept();
    ...
    clientSocket.close();
}
```

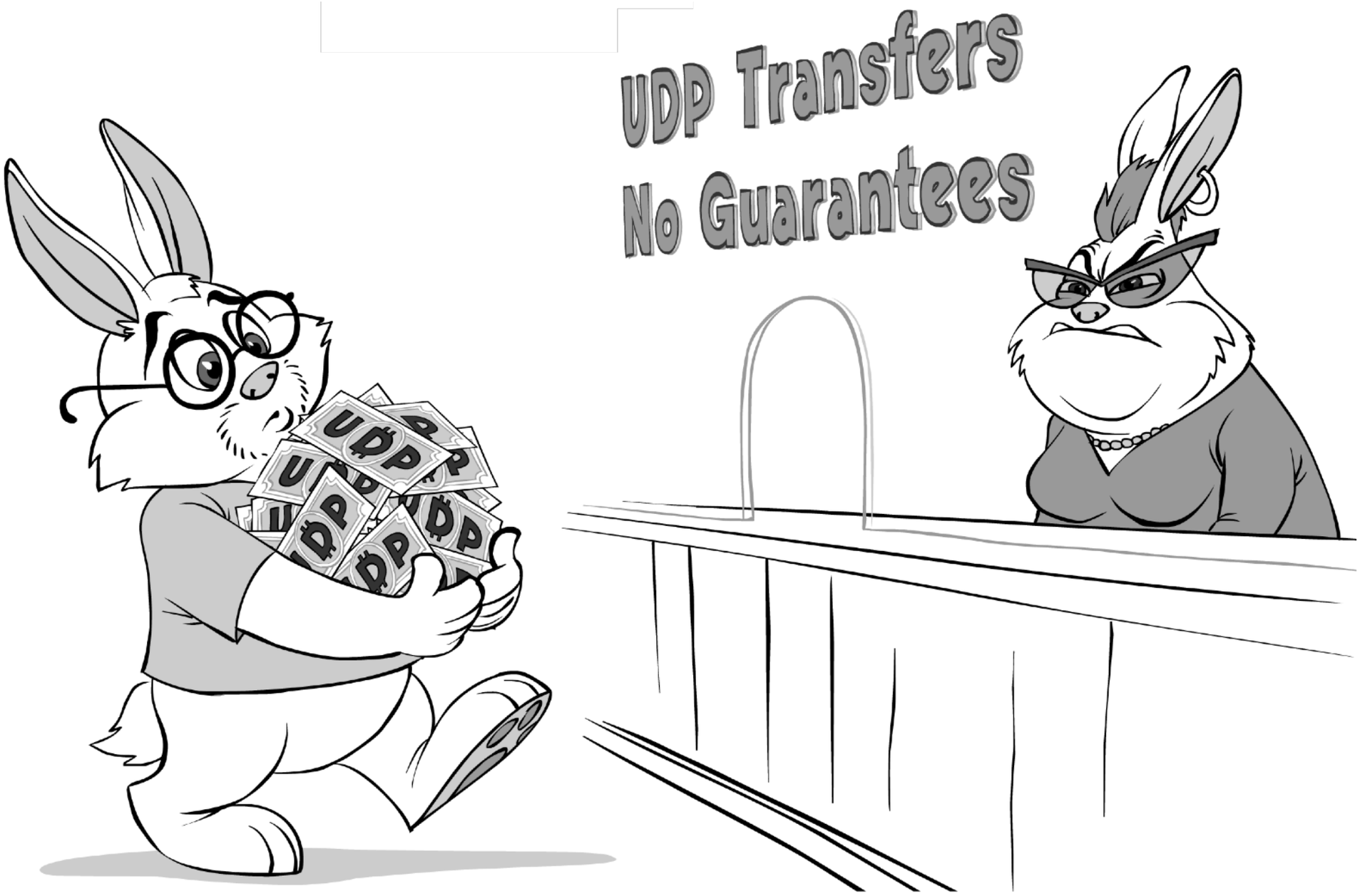
- As a server, you can close the connection if the client does not close it.

TCP Handshake

- For troubleshooting network issues, it is useful to understand the TCP three-way handshake session that establishes the connection.



UDP, no guarantee ?!



UDP notes

- Previous server and client slides are referring to TCP based sockets.
- UPD protocol has its own purpose in network communication.
- Besides being unreliable - some data may be missed, the order is not assured - it is a good fit for broadcasting and streaming data.
- Java API provides a:
  - `DatagramPacket` as a data container, including destination details.
  - `DatagramSocket` for sending or receiving UDP packets.

Further Reading

- https://docs.oracle.com/javase/tutorial/networking/sockets
- http://tutorials.jenkov.com/java-networking/index.html
- Java Network Programming (2013, 4th Edition, O'Reilly)