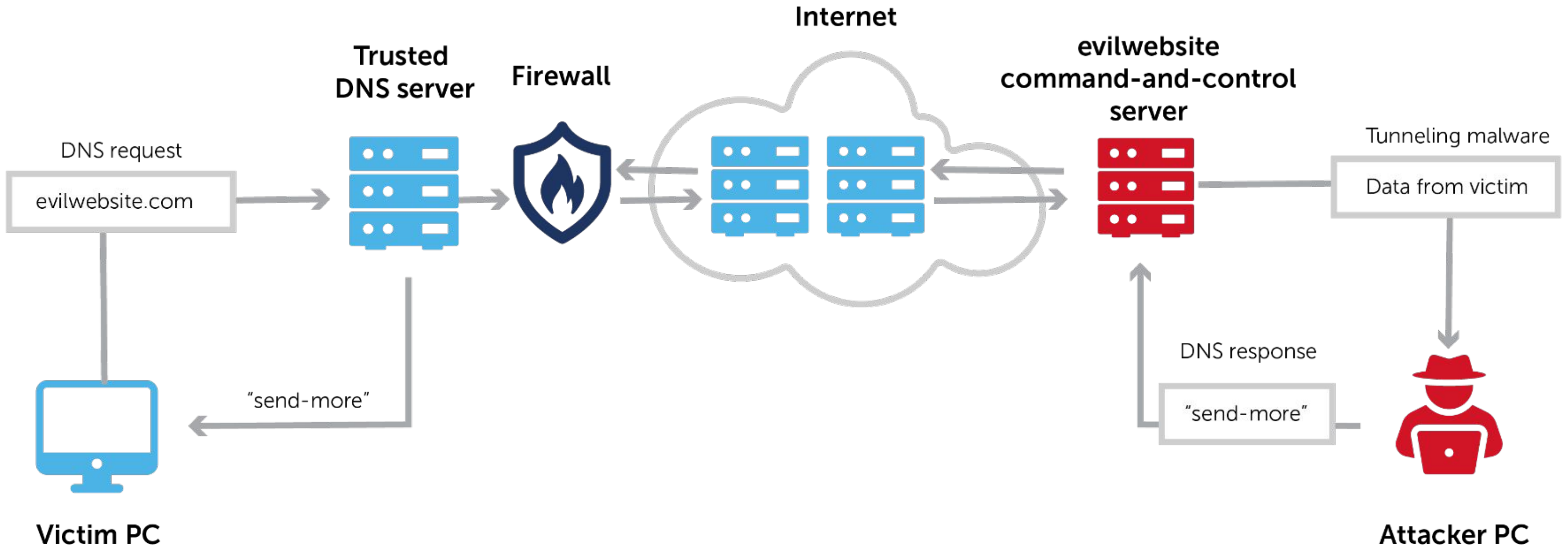
INFO0010 - Introduction to Computer Networking

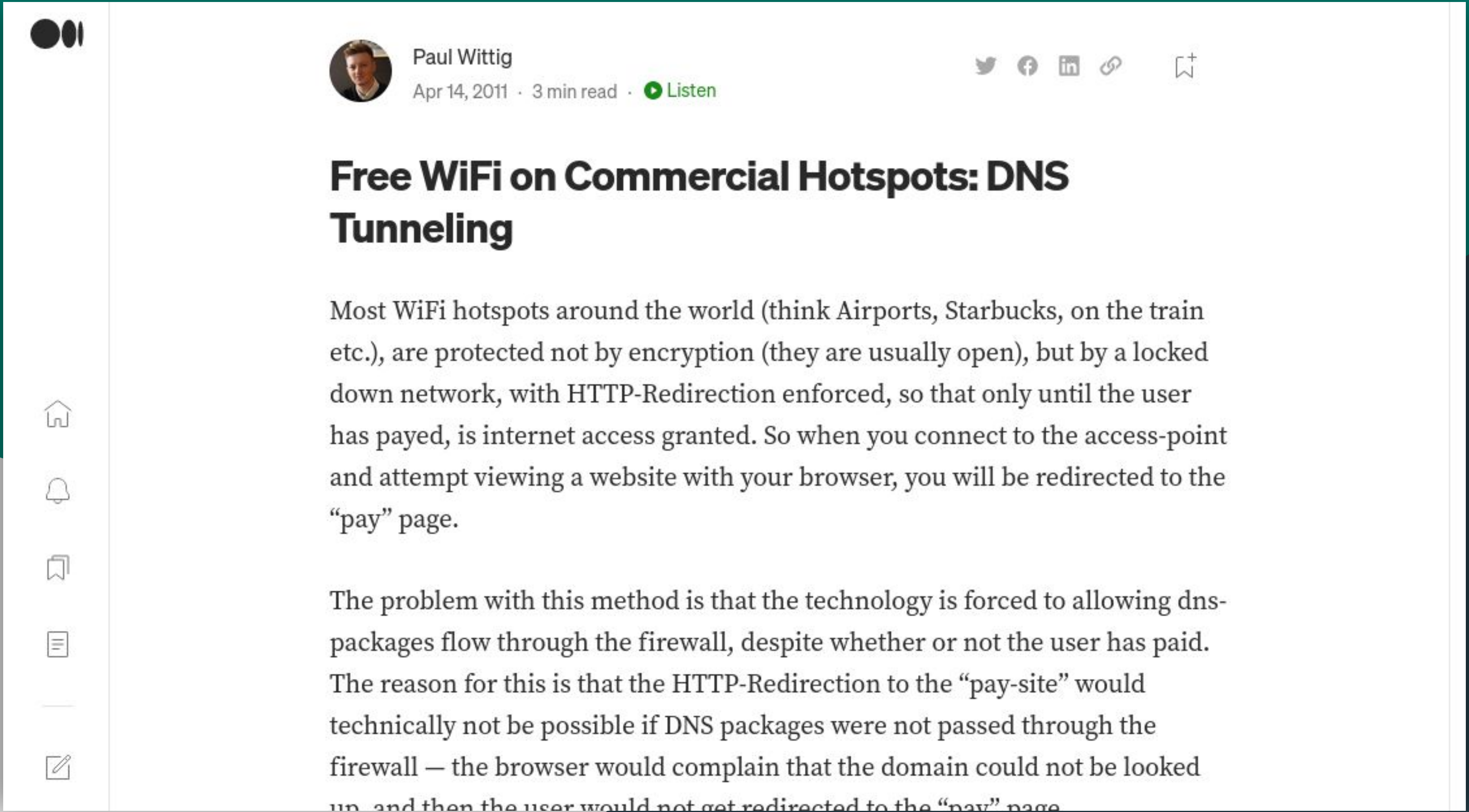
DNS Tunneling

Maxime Aerts, Emeline Maréchal, Guy Leduc
Year 2022 - 2023

Project in a nutshell

DNS tunneling




Paul Wittig

Apr 14, 2011 · 3 min read · Listen

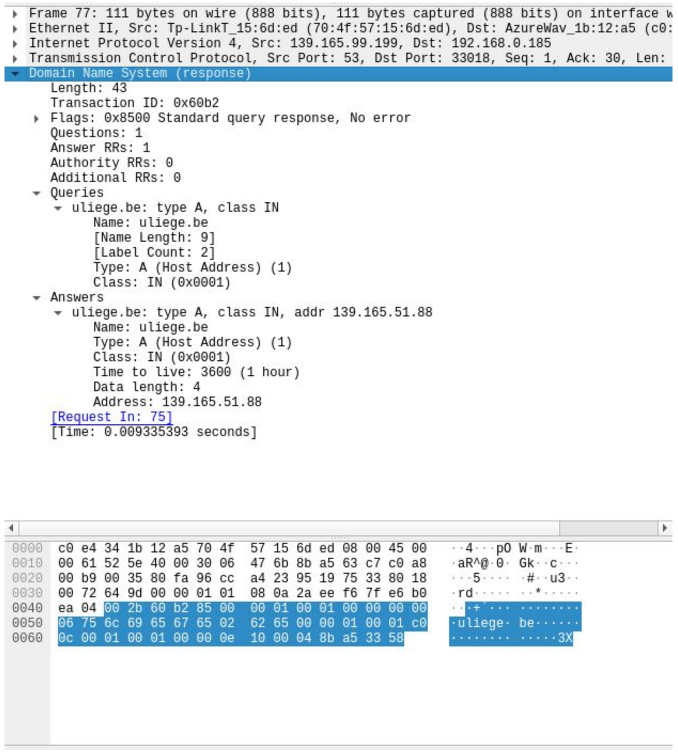
Free WiFi on Commercial Hotspots: DNS Tunneling

Most WiFi hotspots around the world (think Airports, Starbucks, on the train etc.), are protected not by encryption (they are usually open), but by a locked down network, with HTTP-Redirection enforced, so that only until the user has payed, is internet access granted. So when you connect to the access-point and attempt viewing a website with your browser, you will be redirected to the “pay” page.

The problem with this method is that the technology is forced to allowing dns-packages flow through the firewall, despite whether or not the user has paid. The reason for this is that the HTTP-Redirection to the “pay-site” would technically not be possible if DNS packages were not passed through the firewall — the browser would complain that the domain could not be looked up, and then the user would not get redirected to the “pay” page.

Project part 1 - DNS Client

- Your client will be able to:
  - Reach different DNS servers.
  - Send DNS queries to resolve domain names.
  - Parse responses to extract useful information.
- Important points:
  - Queries are sent through **<u>TCP</u>** on port 53.
  - Network is handled with “java.net.Socket”.
  - Only console Input/Output.
  - You will manipulate binary data at application layer.
  - Project 1 must be realised alone.



```
▸ Frame 77: 111 bytes on wire (888 bits), 111 bytes captured (888 bits) on interface w
▸ Ethernet II, Src: Tp-LinkT_15:6d:ed (70:4f:57:15:6d:ed), Dst: AzureWav_1b:12:a5 (c0:
▸ Internet Protocol Version 4, Src: 139.165.99.199, Dst: 192.168.0.185
▸ Transmission Control Protocol, Src Port: 53, Dst Port: 33018, Seq: 1, Ack: 30, Len:
▾ Domain Name System (response)
    Length: 43
    Transaction ID: 0x60b2
  ▸ Flags: 0x8500 Standard query response, No error
    Questions: 1
    Answer RRs: 1
    Authority RRs: 0
    Additional RRs: 0
  ▾ Queries
    ▾ uliege.be: type A, class IN
        Name: uliege.be
        [Name Length: 9]
        [Label Count: 2]
        Type: A (Host Address) (1)
        Class: IN (0x0001)
  ▾ Answers
    ▾ uliege.be: type A, class IN, addr 139.165.51.88
        Name: uliege.be
        Type: A (Host Address) (1)
        Class: IN (0x0001)
        Time to live: 3600 (1 hour)
        Data length: 4
        Address: 139.165.51.88
    [Request In: 75]
    [Time: 0.009335393 seconds]
```

```
0000  c0 e4 34 1b 12 a5 70 4f  57 15 6d ed 08 00 45 00   ··4···pO W·m···E·
0010  00 61 52 5e 40 00 30 06  47 6b 8b a5 63 c7 c0 a8   ·aR^@·0· Gk··c···
0020  00 b9 00 35 80 fa 96 cc  a4 23 95 19 75 33 80 18   ···5···· ·#··u3··
0030  00 72 64 9d 00 00 01 01  08 0a 2a ee f6 7f e6 b0   ·rd····· ··*·····
0040  ea 04 00 2b 60 b2 85 00  00 01 00 01 00 00 00 00   ···+`··· ········
0050  06 75 6c 69 65 67 65 02  62 65 00 00 01 00 01 c0   ·uliege· be······
0060  0c 00 01 00 01 00 00 0e  10 00 04 8b a5 33 58      ········ ·····3X
```

Technical points

- Program must be compatible with Java 8 to run on ms8xx.montefiore.ulg.ac.be.
- We will set up a FAQ on eCampus. If possible, use the platform to ask your questions.
- Full assignment will be delivered ASAP.
- You program will be able to run with the following command: “java Client 139.165.99.199 uliege.be”

Hard deadline: 2nd of November 2022

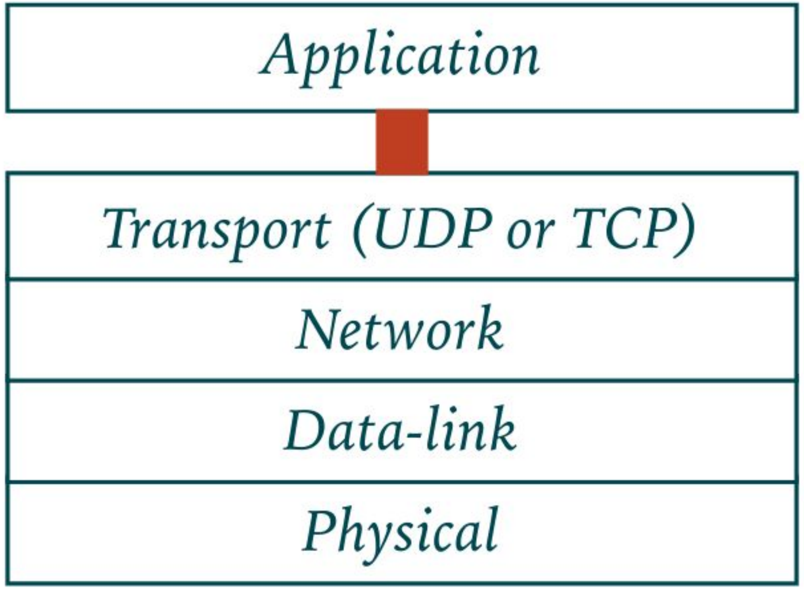
Socket programming

What is a socket?

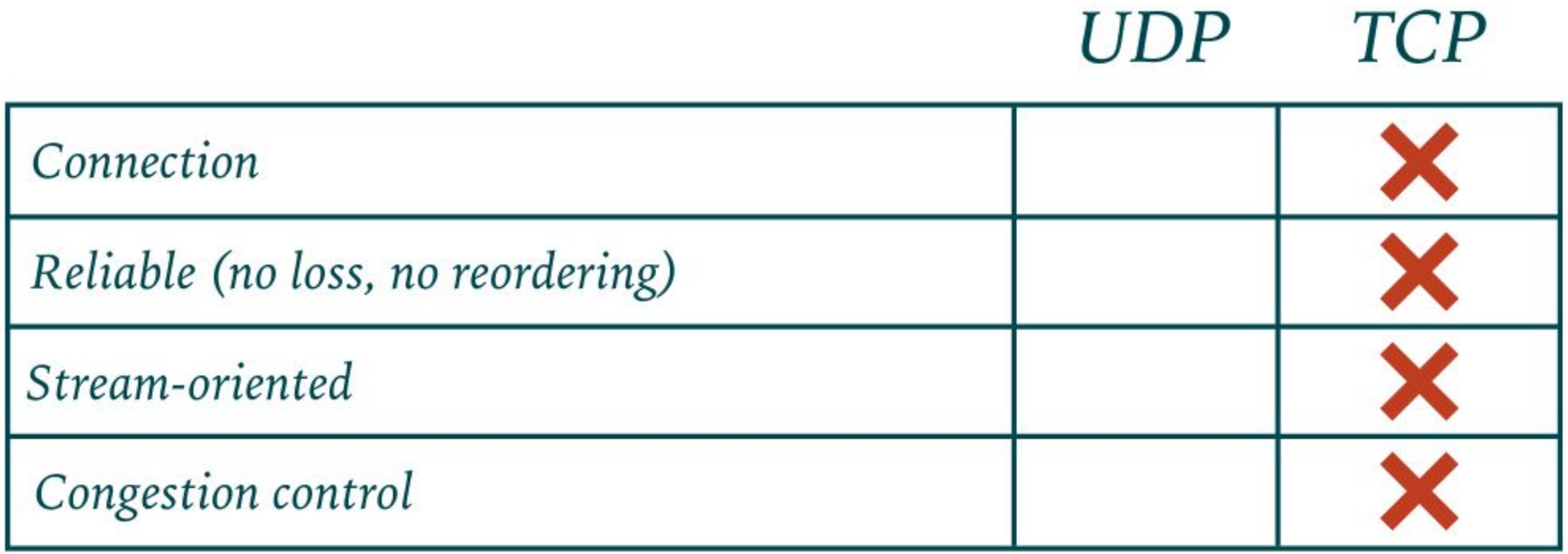
Interface to network protocol stack (typically the transport layer).

Allows communication between processes (same or remote machines).

Like a door through which the application pushes bytes to the transport layer.



TCP vs UDP



	UDP	TCP
Connection		✗
Reliable (no loss, no reordering)		✗
Stream-oriented		✗
Congestion control		✗

TCP - Stream oriented

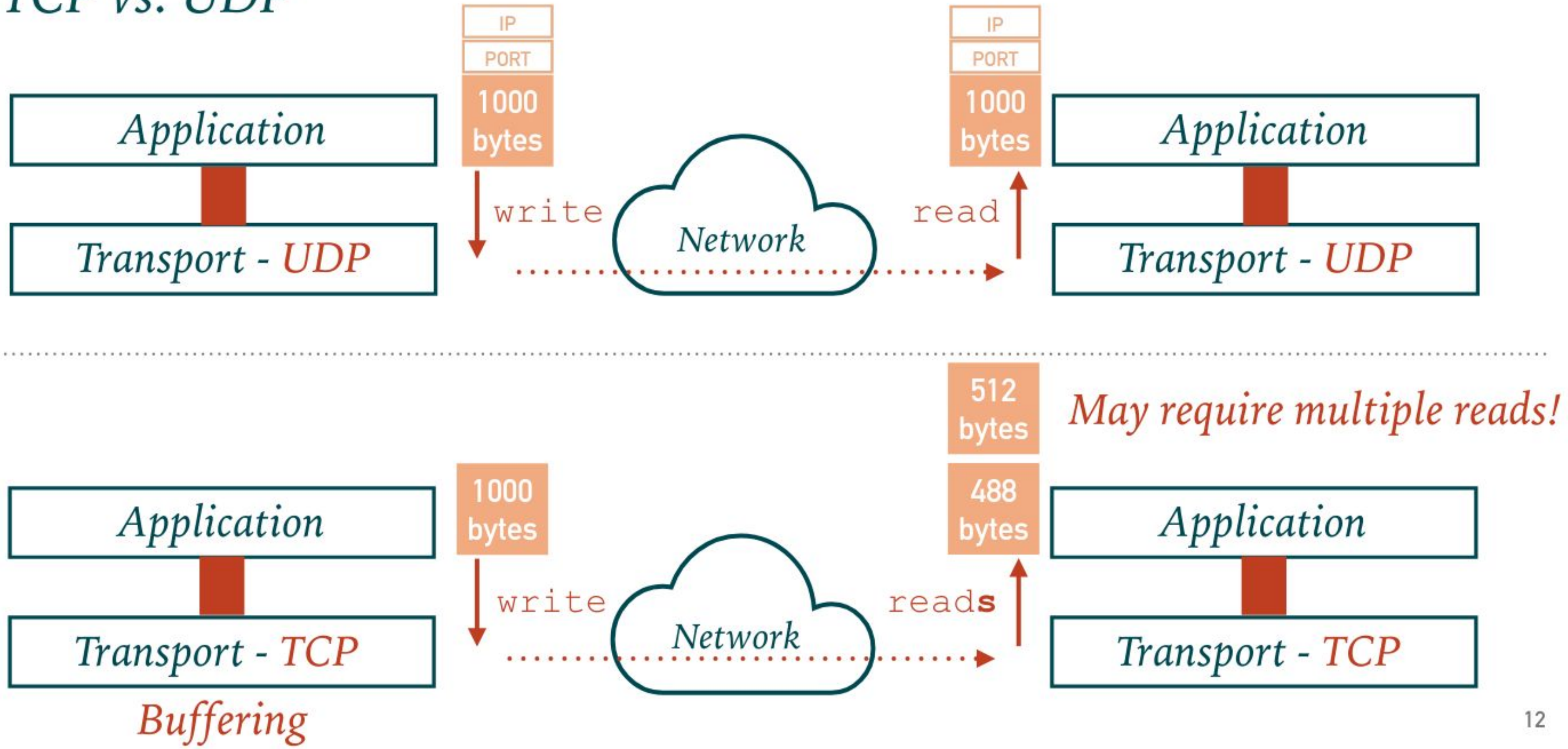
The application sends messages through the TCP socket, but TCP just sees a stream of bytes.

On the two sides of the TCP connection, the application processes may well write/read byte chunks of different sizes.

Streams are identical nevertheless.

Your job is to send and recover application messages to and from the stream.

➤ TCP vs. UDP


IP
PORT
1000 bytes
Application
Transport - UDP
write
Network
read
IP
PORT
1000 bytes
Application
Transport - UDP
512 bytes
May require multiple reads!
Application
Transport - TCP
Buffering
1000 bytes
write
Network
reads
488 bytes
Application
Transport - TCP

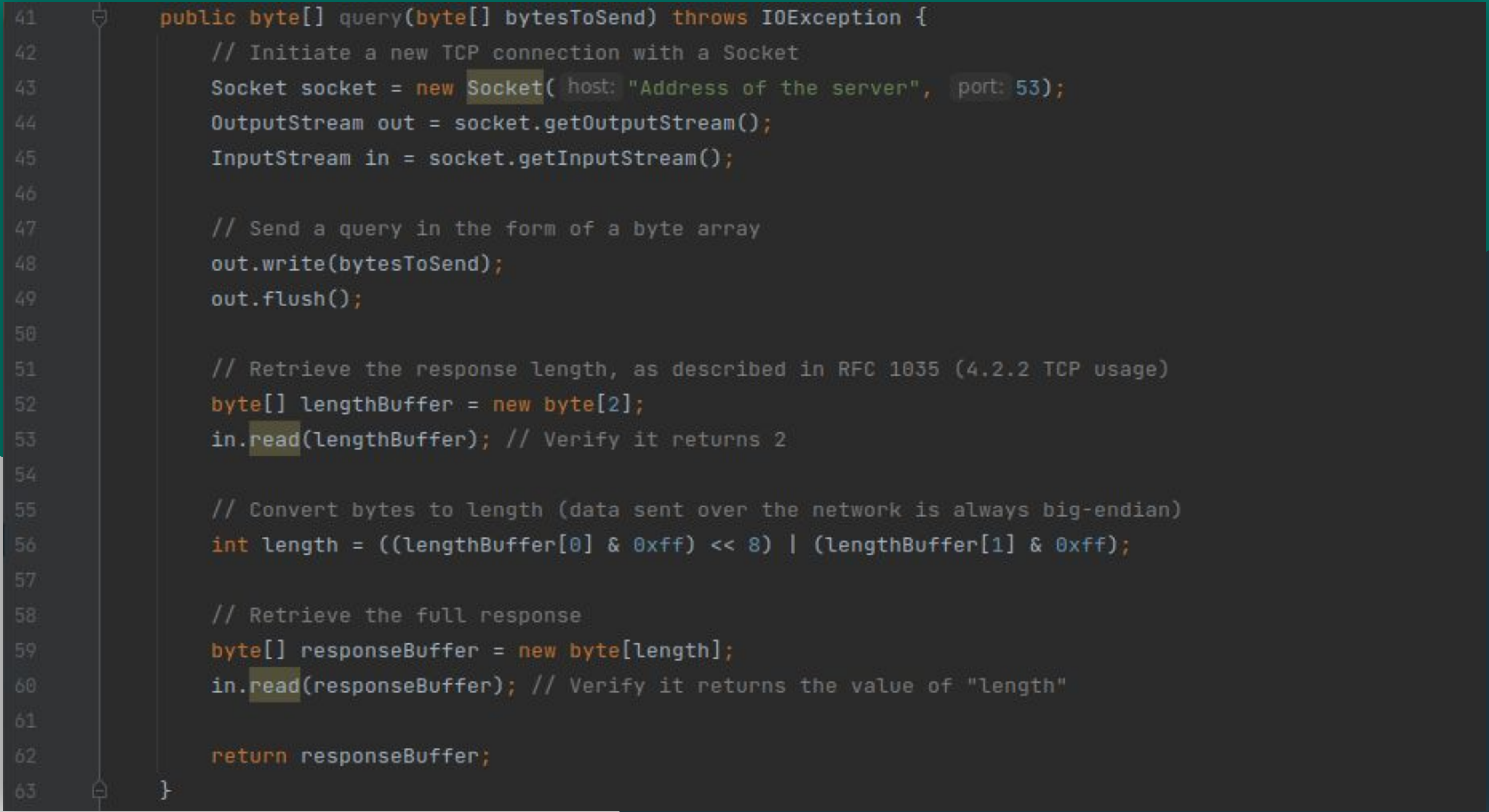
TCP - Congestion control

What has been sent through the socket may not have left the machine after calling write().

Can use flush() to force the writing to the socket.

Even after flushing the data to the socket, TCP can still buffer the data with the Nagle’s algorithm :

- Nagle’s algorithm aims at improving TCP efficiency by reducing the number of small packets sent over the network.
- Not appropriate for highly interactive applications with small data transfer.
- Can use socket.setTcpNoDelay(true) to disable Nagle’s algorithm.



```java
public byte[] query(byte[] bytesToSend) throws IOException {
    // Initiate a new TCP connection with a Socket
    Socket socket = new Socket( host: "Address of the server",  port: 53);
    OutputStream out = socket.getOutputStream();
    InputStream in = socket.getInputStream();

    // Send a query in the form of a byte array
    out.write(bytesToSend);
    out.flush();

    // Retrieve the response length, as described in RFC 1035 (4.2.2 TCP usage)
    byte[] lengthBuffer = new byte[2];
    in.read(lengthBuffer); // Verify it returns 2

    // Convert bytes to length (data sent over the network is always big-endian)
    int length = ((lengthBuffer[0] & 0xff) << 8) | (lengthBuffer[1] & 0xff);

    // Retrieve the full response
    byte[] responseBuffer = new byte[length];
    in.read(responseBuffer); // Verify it returns the value of "length"

    return responseBuffer;
}
```

Tips

Your new best friends

- RFC 1035 - Section 4. MESSAGES: describes precisely the format of DNS messages
- java.net.Socket
- java.nio.ByteBuffer
- Bitwise AND (&): useful to apply a mask
- Shift operators (<<, >>): useful to move byte patterns at their final position
- Bitwise OR (|): useful to concatenate multiple parts in the same byte or integer
- Binary literals (**0b**0101): useful to complete static flags
- Typecasting: Java will often let you end up with integers, even when you want to have bytes or shorts

Also

- You can use Wireshark to verify the packets sent and received.
- A good IDE can save you time. These ones are free: IntelliJ, VSCode (with plugins), Eclipse, …
- If possible use eCampus to ask questions, otherwise you can contact me at maerts@uliege.be.

GL

HF