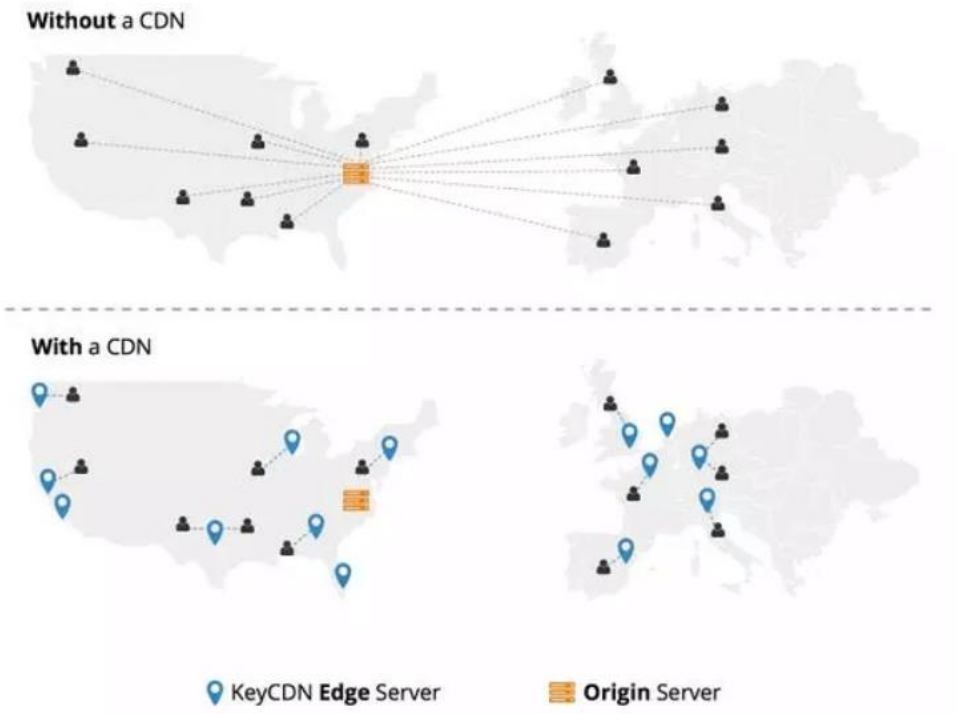
# CS 305 Lab Tutorial
# Lab 6 CDN & WebSocket

Dept. Computer Science and Engineering
Southern University of Science and Technology

SUSTech
Southern University
of Science and Technology

# Part 1. CDN

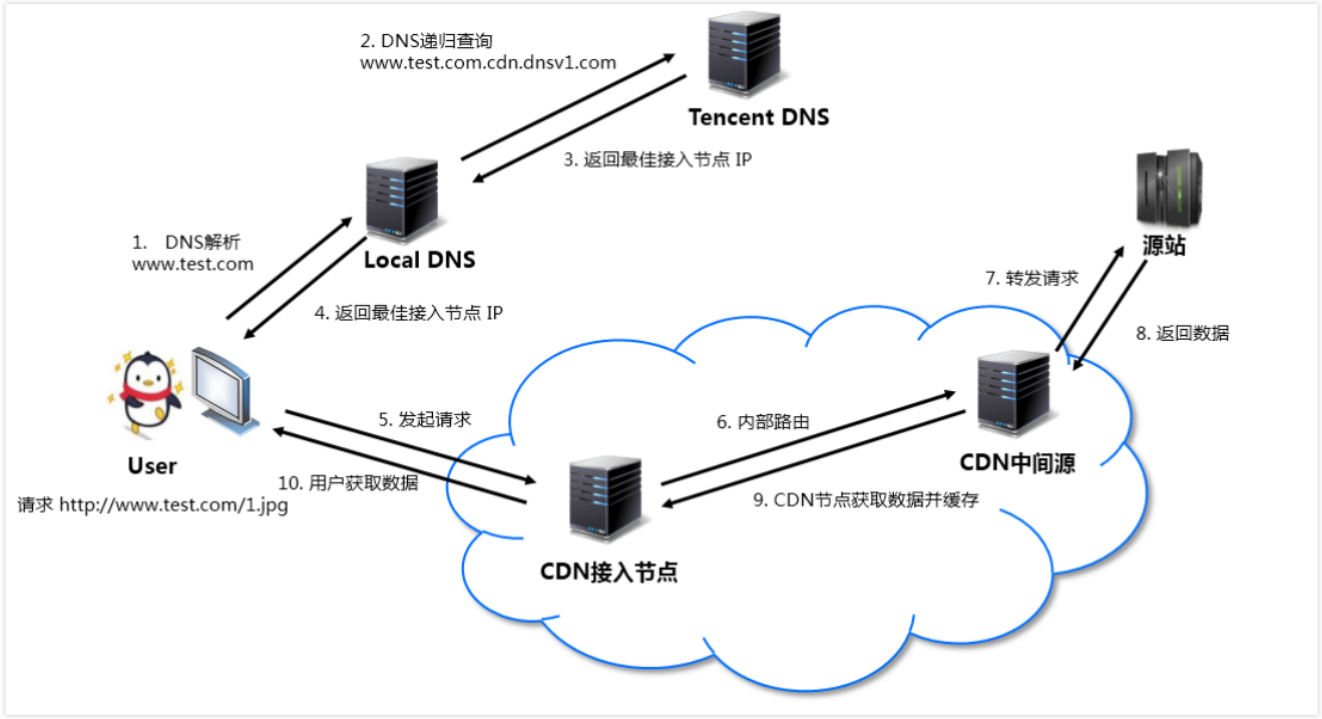



CDN is to cache content on a node closer to the user to improve the user experience;

What are the scenarios of CDN?

- **Big flow** website, such as: online video, games, pictures, audio, social, e-commerce, download stations, etc.
- CDN is suitable for a certain level of **static resource** access, including html, js, css, apk, mp3, flv, jpg, gif, mp4, flv and all other static resources.

# Part 1. CDN




If: X-Cache(HIT ): TCP_MEM_HIT means hit cache.
If: X-Cache(MISS): TCP_MISS means Miss cache.

SUSTech
Southern University
of Science and Technology

https://cloud.tencent.com/document/product/228/2939

# Part 1. CDN

Q: How long is the cache time of file on the CDN server ?

A:

- The caching time of files refers to the cache time cycle of files in browsers.
- The CDN cache server strictly adheres to the standard HTTP protocol, and the cache time is controlled by the Cache-Control and Expires response headers in the HTTP response header
- Html file cache time viewing: Look at the Cache-Control in the HTTP header, such as "Cache-Control max-age = 2592000 (seconds)" to indicate that the file will be cached for 30 days. At this point, unless you use manual refresh, the newly opened browser page will not go back to the source to retrieve the file during the file cache cycle.

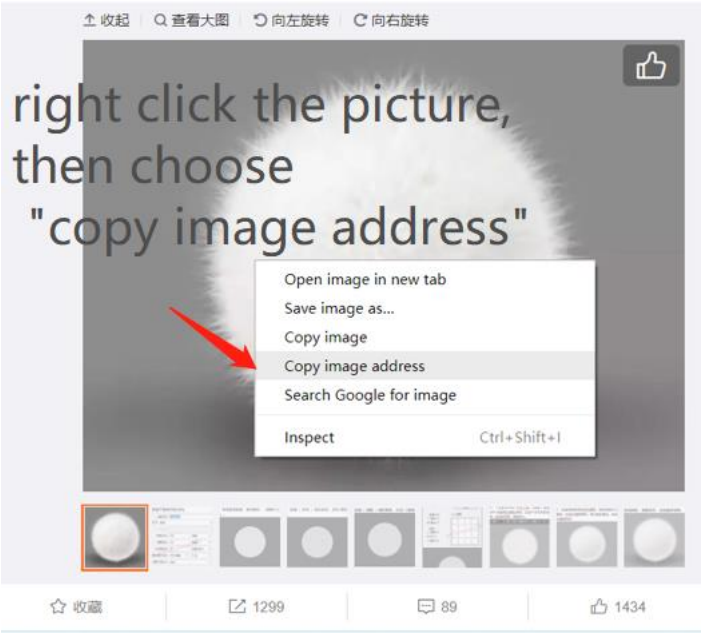
# Demo 1

1. Find a web sit which may use CDN, find a static resource on it

2. To get the URL of this static resource(such as a picture)

3. Using command “curl” to get the content of this static resource

4. Check if this is on the CDN node or not based on the command output



# The result of curl

```
C:\Users\Administrator>curl https://d4.sina.com.cn/202110/15/1581703.jpg --head
HTTP/1.1 200 OK
Server: nginx
Date: Mon, 18 Oct 2021 01:25:27 GMT
Content-Type: image/jpeg
Content-Length: 86080
Connection: keep-alive
X-RequestId: 0cbc680a-2110-1809-0045-0894eff93828
X-Requester: GRPS000000ANONYMOUSE
Last-Modified: Fri, 15 Oct 2021 07:16:45 GMT
X-Filesize: 86080
ETag: "96292990b34d783193938a31187d892c"
x-amz-meta-crc32: 12D0DBA9
x-amz-meta-uploadlocation: /ad4
Cache-Control: max-age=604800
Access-Control-Allow-Headers: Origin, Content-Type, Accept, Range, Content-Length
Access-Control-Allow-Methods: GET, PUT, POST, DELETE, OPTIONS, HEAD
Access-Control-Max-Age: 31536000
Access-Control-Allow-Origin: *
Expires: Mon, 25 Oct 2021 01:00:45 GMT
Edge-Copy-Time: 1634518845207
Age: 1483
Via: https/1.1 dfwx.guangdong.union.163 (ApacheTrafficServer/6.2.1 [cRs f ])
X-Cache: HIT.163
X-Via-CDN: f=edge,s=dfwx.guangdong.union.163.nb.sinaedge.com,c=10.245.100.15;f=Edge,s=dfwx.guangdong.union.163,c=103.116
.123.163
X-Via-Edge: 16345203279890f64f50aa37b74676d6ca013
```

A static resource which is cached on a CDN node

# Part 2. WebSocket

- The WebSocket Protocol is designed to supersede existing bidirectional communication technologies that use HTTP as a transport layer to benefit from existing infrastructure (proxies, filtering, authentication).
- The WebSocket Protocol attempts to address the goals of existing bidirectional HTTP technologies in the context of the existing HTTP infrastructure.
  - support HTTP proxies and intermediaries
  - does not limit WebSocket to HTTP, and future implementations could use a simpler handshake over a dedicated port without reinventing the entire protocol.

https://www.rfc-editor.org/rfc/rfc6455.txt

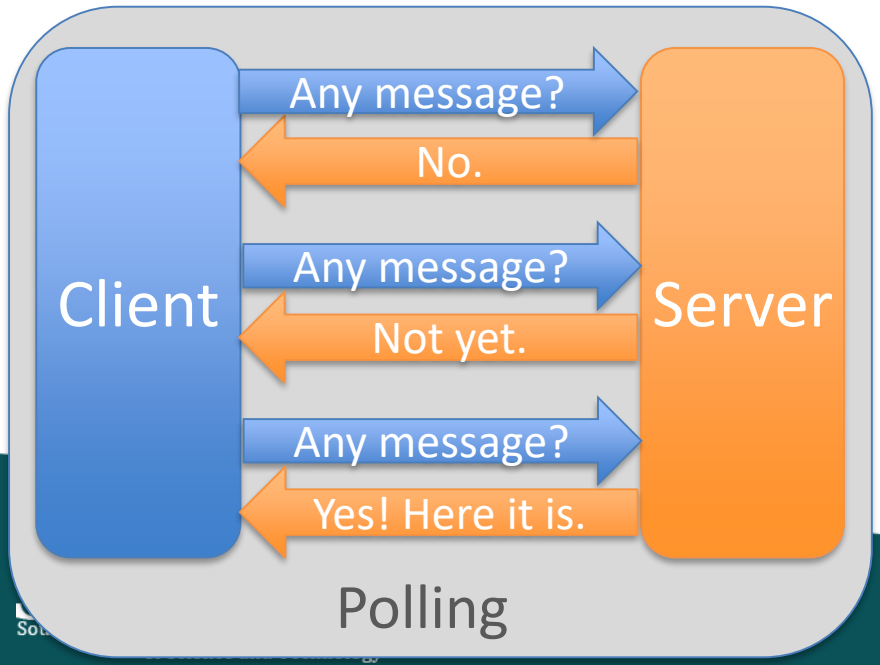
# Main features of WebSocket

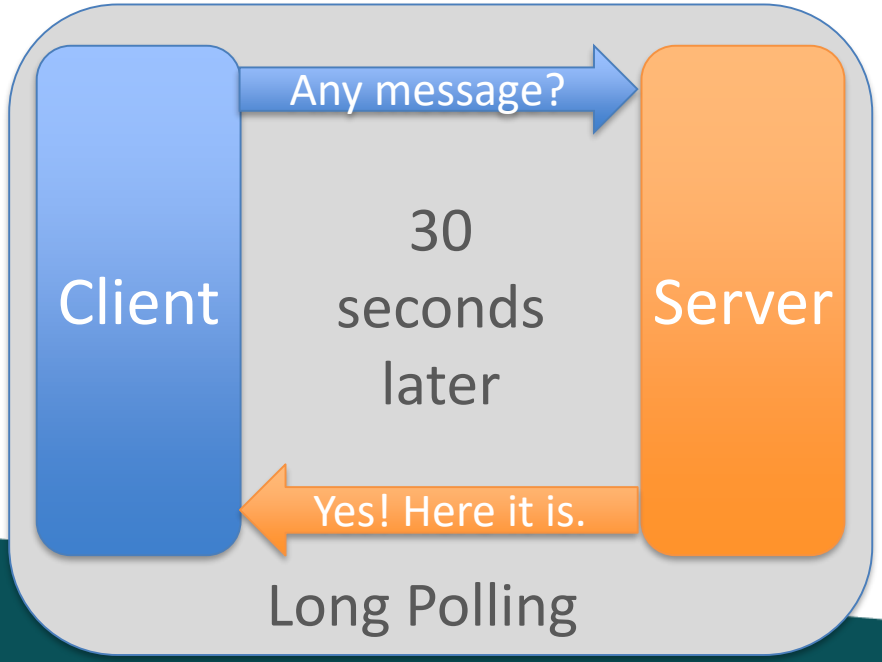
- Application layer protocol
- Established on TCP protocol
- Use the port 80
- Full duplex
- ws-URL= "ws:" "//" host [ ":" port ] path [ "?" query ]
  - ws://example.com/chat
- wss-URL= "wss:" "//" host [ ":" port ] path [ "?" query ]

https://www.rfc-editor.org/rfc/rfc6455.txt

# HTTP polls for new messages

- Historically, creating web applications that need bidirectional communication between a client and a server (e.g., instant messaging and gaming applications) has required an abuse of HTTP to poll the server for updates while sending upstream notifications as distinct HTTP calls .



# WebSocket protocol overview

- Two parts
  - Handshakes: Opening Handshake & Closing Handshake
  - Data transfer
- The WebSocket Protocol is an independent TCP-based protocol. Its only relationship to HTTP is that its handshake is interpreted by HTTP servers as an Upgrade request.
  - Uses port 80 for regular WebSocket connections
  - Uses port 443 for WebSocket connections tunneled over TLS
  - Can not establish a connection with servers of pre-existing protocols like SMTP and HTTP

# Opening handshake

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

client

example.com

"dGhlIHNhbXBsZSBub25jZQ=="

Step1. concatenate "258EAFA5-E914-47DA-95CA-C5AB0DC85B11"

Step2. take the SHA-1 hash

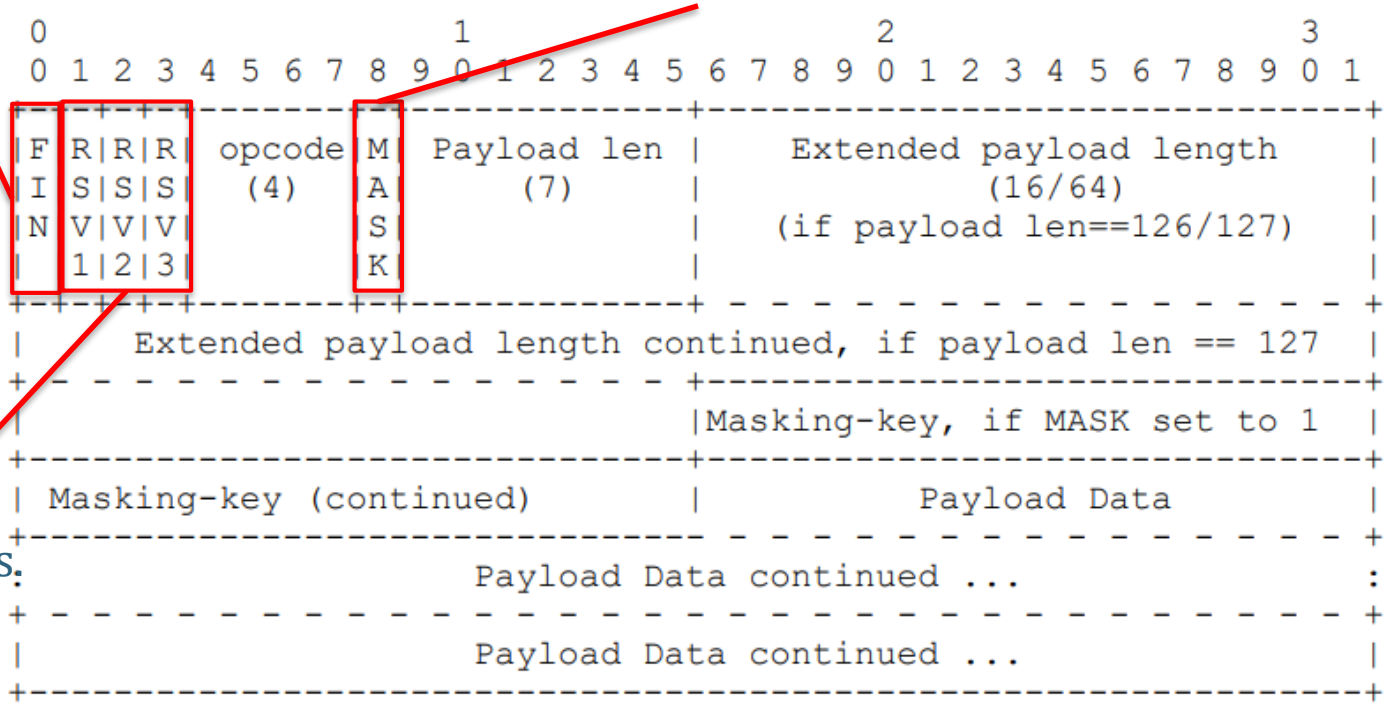
Step3. base64-encoded the hash value

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept:
s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

# Data framing

Indicates that this is the final fragment in a message.

Defines whether the "Payload data" is masked.




MUST be 0 unless an extension is negotiated that defines meanings for non-zero values.

SUSTech
Southern University of Science and Technology

# Data framing (continued)

- Opcode (4 bits)
  - %x0(0000) denotes a continuation frame
  - %x1(0001) denotes a text frame
  - %x2(0010) denotes a binary frame
  - %x3-7(0011-0111) are reserved for further non-control frames
  - %x8(1000) denotes a connection close
  - %x9(1001) denotes a ping
  - %xA(1010) denotes a pong
  - %xB-F(1011-1111) are reserved for further control frames

# Data framing (continued)

- Payload len
  - 7 bits: if 0~125 bytes
  - 7 + 16 bit: if the 7 bits equals to 126
  - 7 + 64 bits: if the 7 bits equals to 127
  - payload length = the length of the "Extension data" + the length of the "Application data"
- Masking key
  - 0 bits: if the mask bit is set to 1
  - 32 bits: if the mask bit is set to 0
  - The masking key is a 32-bit value chosen at random by the client.

# Data framing examples

- A single-frame unmasked text message
  - 0x81 0x05 0x48 0x65 0x6c 0x6c 0x6f (contains "Hello")
- A single-frame masked text message
  - 0x81 0x85 0x37 0xfa 0x21 0x3d 0x7f 0x9f 0x4d 0x51 0x58 (contains "Hello")
- A fragmented unmasked text message
  - 0x01 0x03 0x48 0x65 0x6c (contains "Hel")
  - 0x80 0x02 0x6c 0x6f (contains "lo")
- Unmasked Ping request and masked Ping response
  - 0x89 0x05 0x48 0x65 0x6c 0x6c 0x6f (contains a body of "Hello", but the contents of the body are arbitrary)
  - 0x8a 0x85 0x37 0xfa 0x21 0x3d 0x7f 0x9f 0x4d 0x51 0x58 (contains a body of "Hello", matching the body of the ping)
- 256 bytes binary message in a single unmasked frame
  - 0x82 0x7E 0x0100 [256 bytes of binary data]

# Closing the connection

- An endpoint MUST send a Close control frame(opcode = 1000)
- Connection Close Code
  - 1000: a normal closure.
  - 1001: an endpoint is "going away".
  - 1002: a protocol error occurs.
  - 1003: an endpoint has received a type of data it cannot accept.
  - 1004, 1005, 1006, 1015: Reserved.
  - 1007: an endpoint has received an inconsistent type of data.
  - 1008: an endpoint has received a message violates its policy.
  - 1009: an endpoint has received a message that is too big to process.
  - 1010: the client has expected the server to negotiate one or more extensions but received no response about that.
  - 1011: the server encountered an unexpected condition.

# Example 1: Mimic a WebSocket Server

```
import asyncio
import websockets

async def echo(websocket, path):
    async for message in websocket:
        message = "I got your message: {}".format(message)
        await websocket.send(message)

asyncio.get_event_loop().run_until_complete(
    websockets.serve(echo, '127.0.0.1', 8766))
asyncio.get_event_loop().run_forever()
```

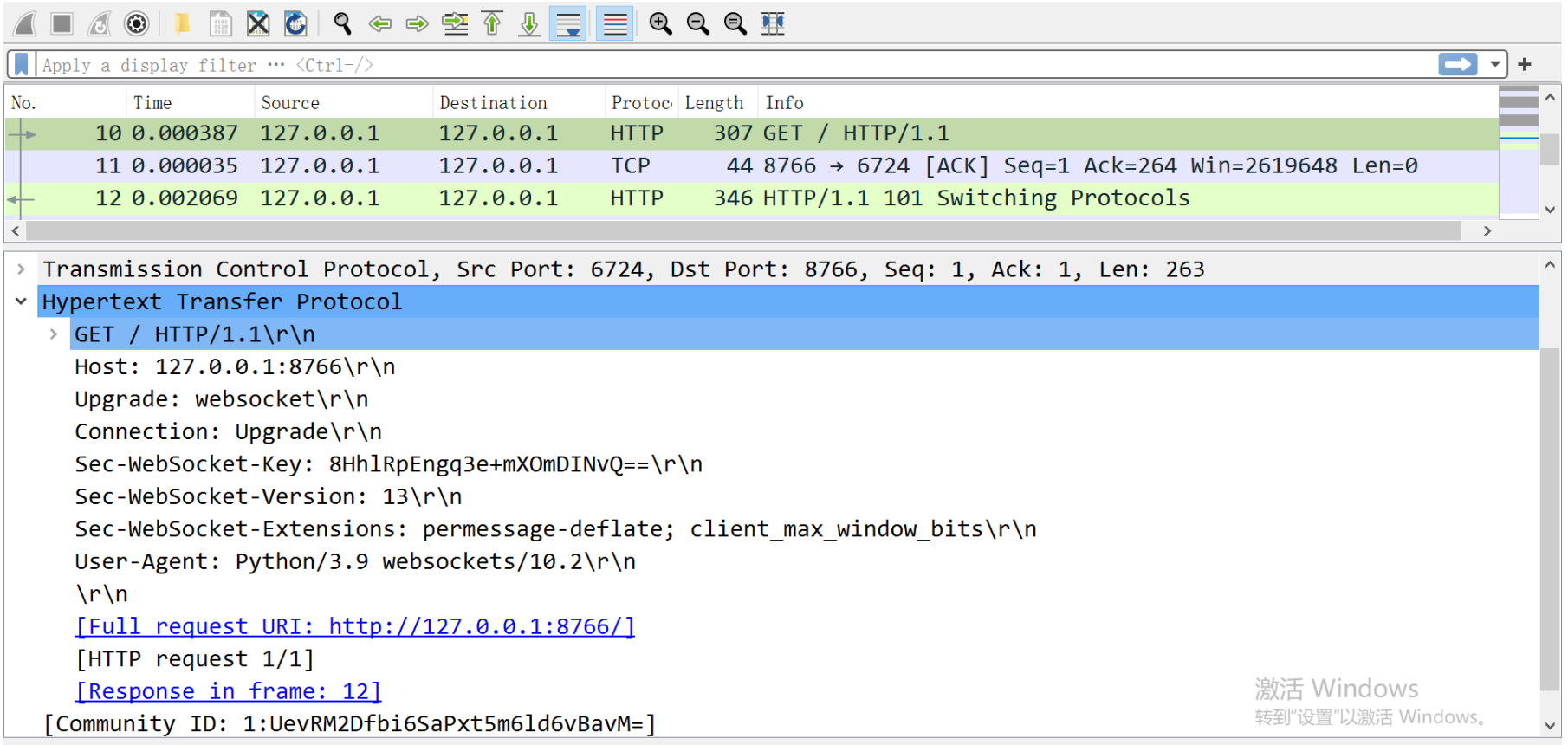
# Example 2: Mimic a WebSocket Client

```
import asyncio
import websockets

async def echo(uri):
    async with websockets.connect(uri) as websocket:
        while True:
            message = input("Write down your message:")
            await websocket.send(message)
            print("<", message)
            recv_text = await websocket.recv()
            print("> {}".format(recv_text))

asyncio.get_event_loop().run_until_complete(
    echo('ws://127.0.0.1:8766'))
```

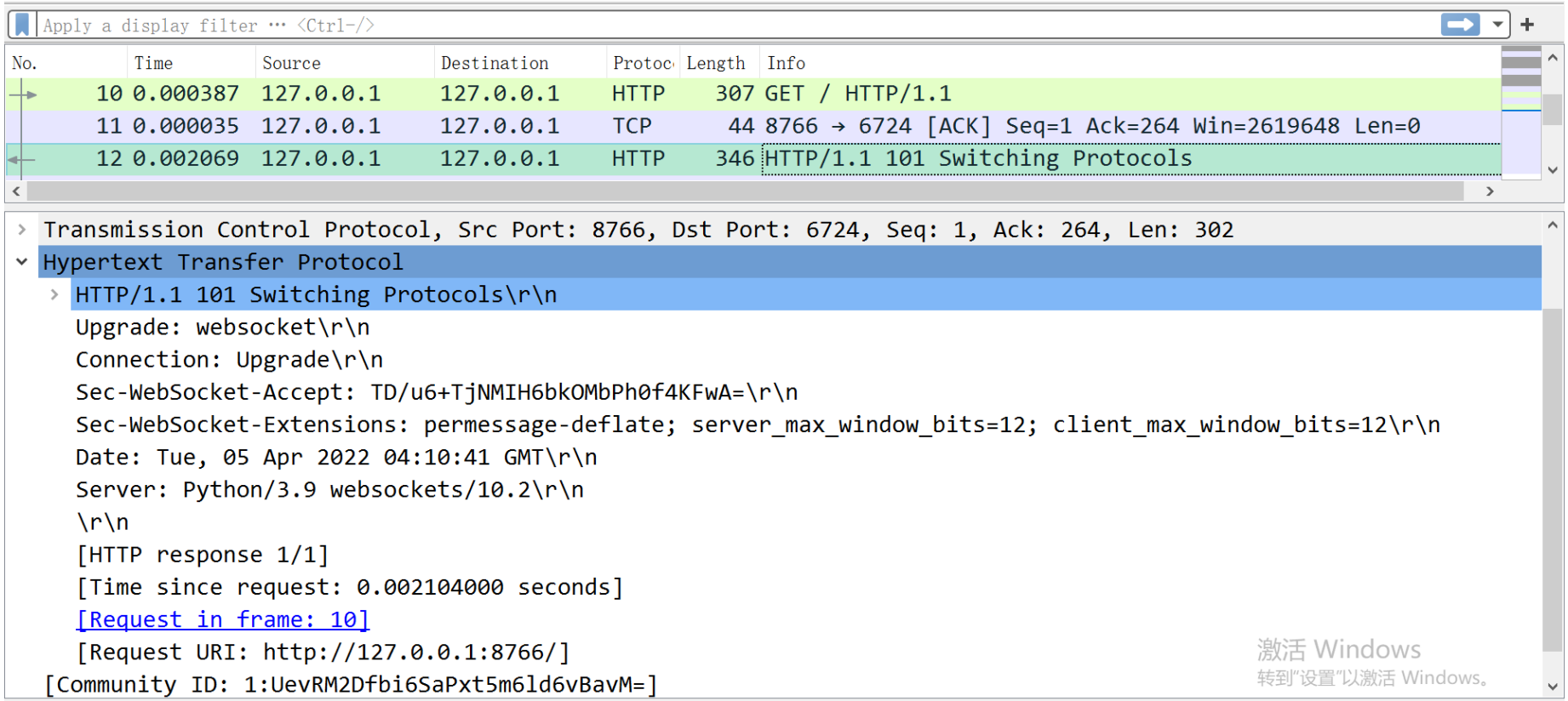
# Example 3: Use curl

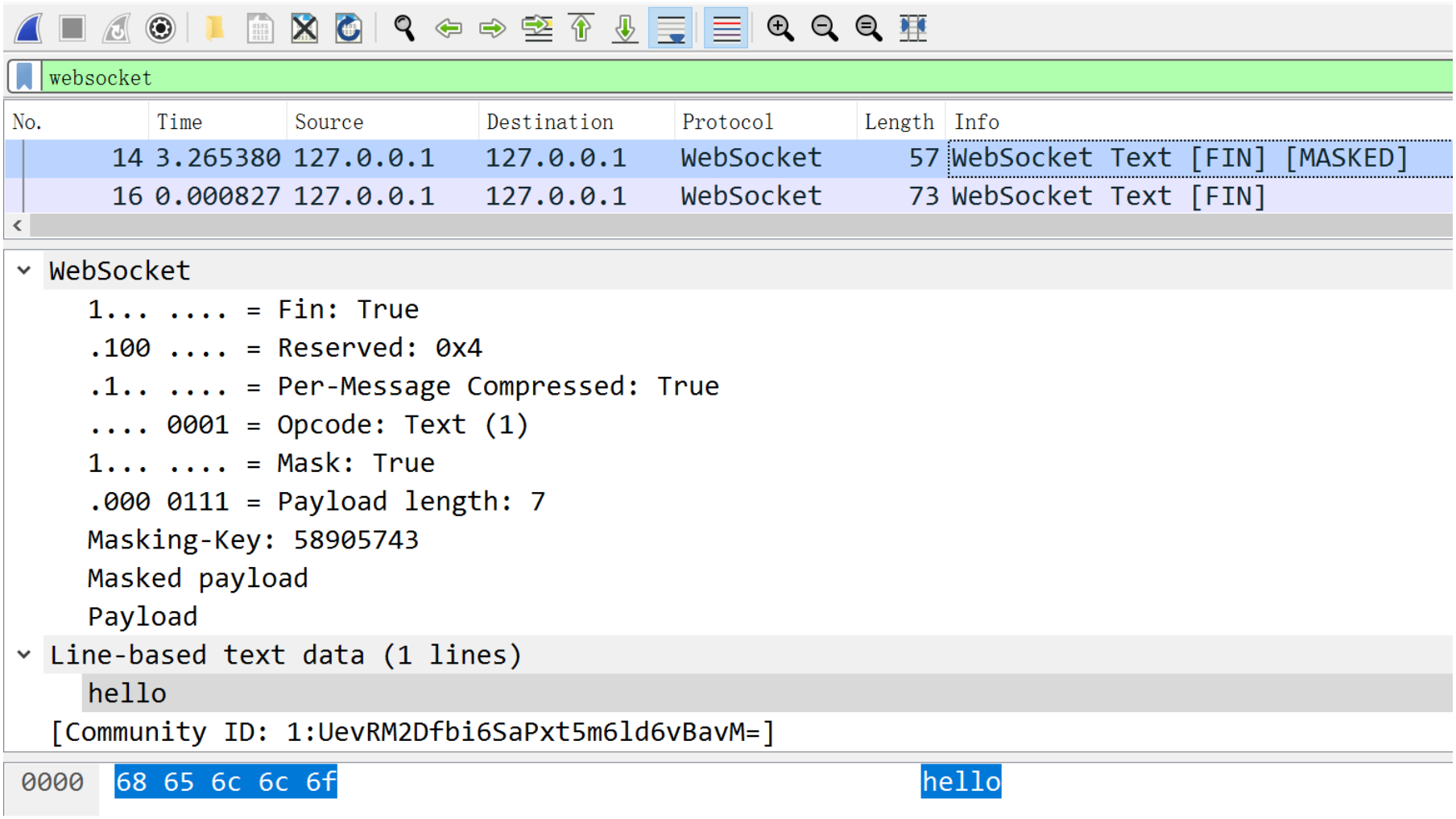
> curl --include --no-buffer --header "Connection: Upgrade" --header "Upgrade: websocket" --header "Host: example.com:80" --header "Origin: http://example.com:80"  --header "Sec-WebSocket-Key: GVsbG8sIHdvcmxkIQ==" --header "Sec-WebSocket-Version: 13" http://example.com:80/

```
C:\Users\wq>curl --include --no-buffer --header "Connection: Upgrade" --header "Upgrade: w
ebsocket" --header "Host: example.com:80" --header "Origin: http://example.com:80"  --head
er "Sec-WebSocket-Key: GVsbG8sIHdvcmxkIQ==" --header "Sec-WebSocket-Version: 13" http://ex
ample.com:80/
HTTP/1.1 200 OK
Accept-Ranges: bytes
Cache-Control: max-age=604800
Content-Type: text/html; charset=UTF-8
Date: Tue, 05 Apr 2022 03:56:49 GMT
Etag: "3147526947"
Expires: Tue, 12 Apr 2022 03:56:49 GMT
Last-Modified: Thu, 17 Oct 2019 07:18:26 GMT
Server: EOS (vny/0454)
Content-Length: 1256
Connection: close

<!doctype html>
<html>
<head>
```

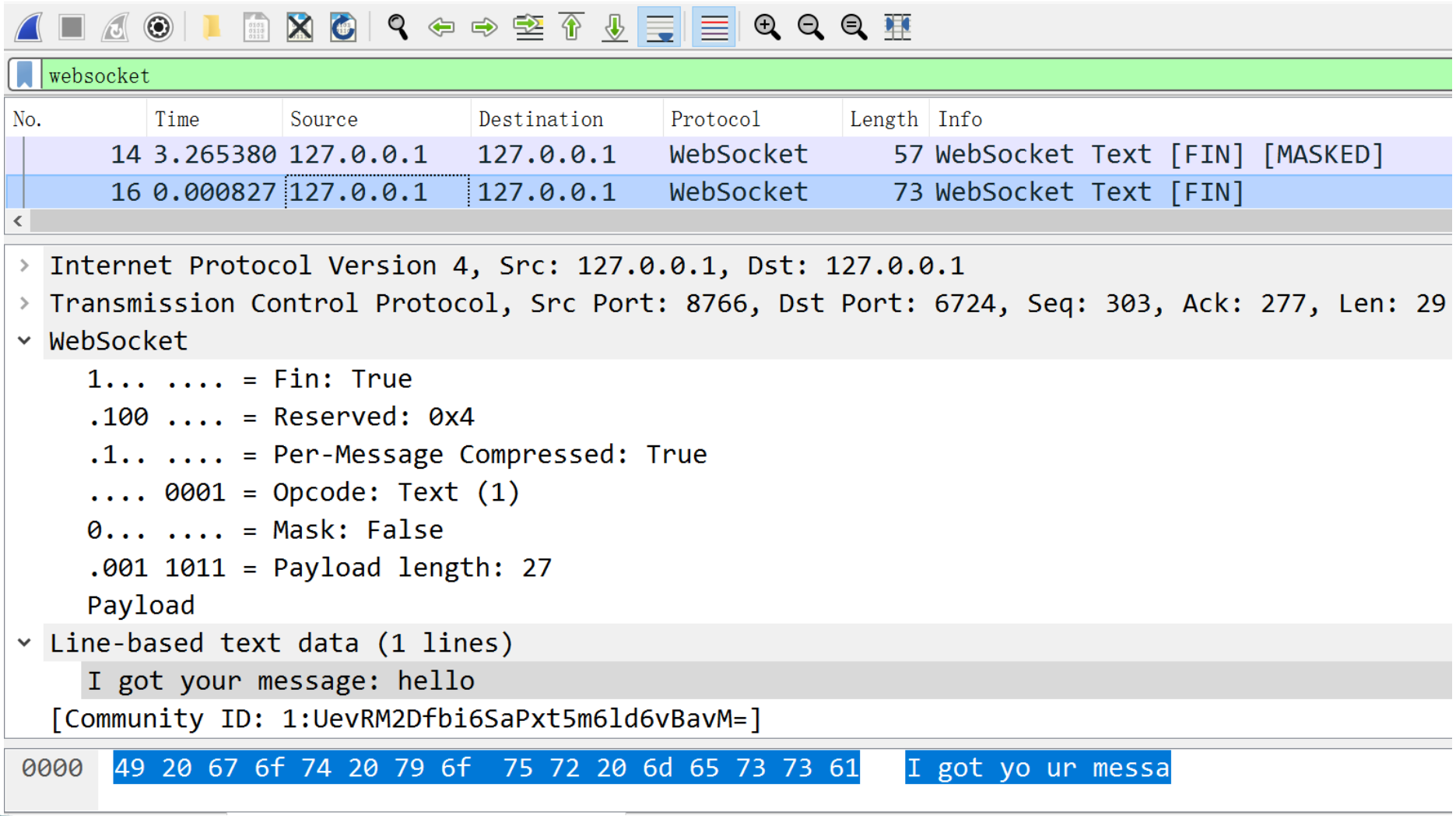



SUSTech
Southern University of Science and Technology


Apply a display filter ··· <Ctrl-/>
No. Time Source Destination Protoc Length Info
10 0.000387 127.0.0.1 127.0.0.1 HTTP 307 GET / HTTP/1.1
11 0.000035 127.0.0.1 127.0.0.1 TCP 44 8766 → 6724 [ACK] Seq=1 Ack=264 Win=2619648 Len=0
12 0.002069 127.0.0.1 127.0.0.1 HTTP 346 HTTP/1.1 101 Switching Protocols
Transmission Control Protocol, Src Port: 8766, Dst Port: 6724, Seq: 1, Ack: 264, Len: 302
Hypertext Transfer Protocol
HTTP/1.1 101 Switching Protocols\r\n
Upgrade: websocket\r\n
Connection: Upgrade\r\n
Sec-WebSocket-Accept: TD/u6+TjNMIH6bkOMbPh0f4KFwA=\r\n
Sec-WebSocket-Extensions: permessage-deflate; server_max_window_bits=12; client_max_window_bits=12\r\n
Date: Tue, 05 Apr 2022 04:10:41 GMT\r\n
Server: Python/3.9 websockets/10.2\r\n
\r\n
[HTTP response 1/1]
[Time since request: 0.002104000 seconds]
[Request in frame: 10]
[Request URI: http://127.0.0.1:8766/]
[Community ID: 1:UevRM2Dfbi6SaPxt5m6ld6vBavM=]
激活 Windows
转到"设置"以激活 Windows。

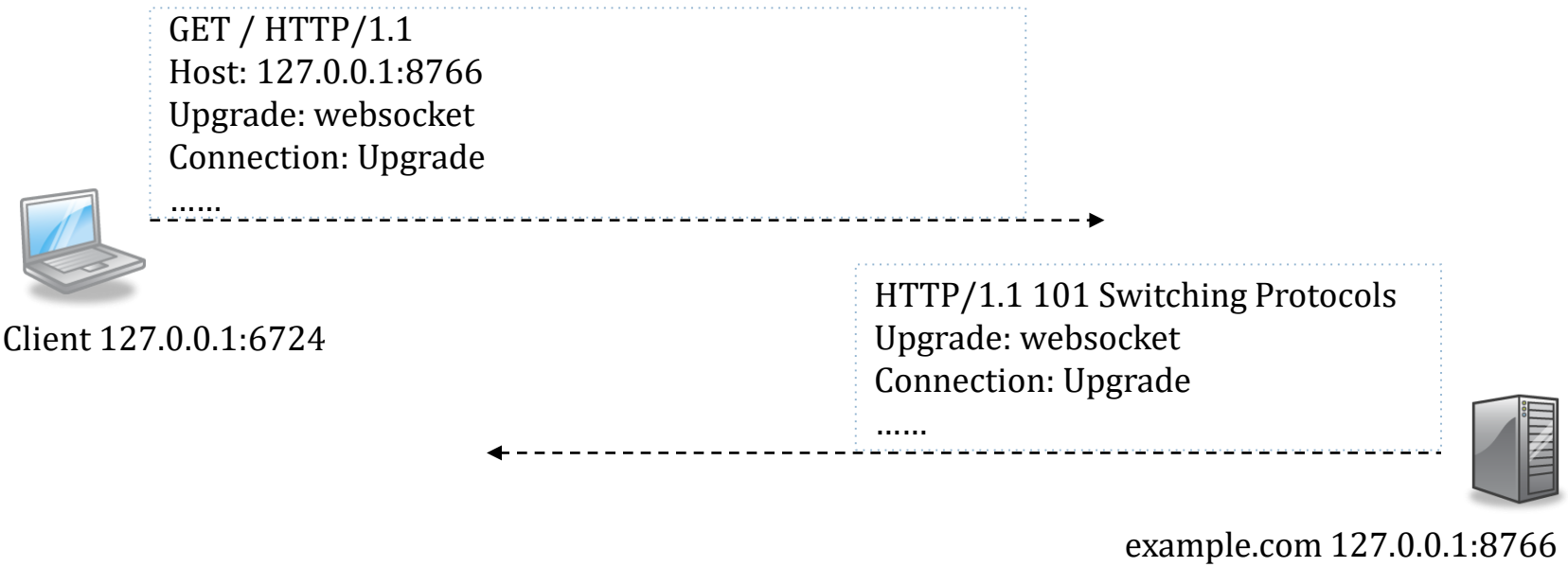



SUSTech
Southern University of Science and Technology




SUSTech
Southern University of Science and Technology

# Process analysis



# Practise 6.1 Finding a CDN user

- Use curl to get a resource from web which using CDN to upgrade the accessing speed and balance the traffic load
  - How can you tell that this web is using CDN
  - Use nslookup/dig in your computer to find the IP address of this web sit
  - Ask your friend who is in another province to practice the same thing(using nslookup/dig to find the IP address of the web site which using CDN, find the IP address of this web site.)

# Practise 6.2 Implement a simple だんま(danmaku) system

- Use python to implement a simple だんま(danmaku) system.
  - Use a python file as a server and a html file as a client.
  - The system should listen and accept WS messages.
  - Receive danmakus from clients, and send them to all clients and display them on their screens.
  - You should have at least 3 clients, i.e., on at least 3 browser tabs, to show whether danmakus are correctly sent and appear simultaneously on each client.
  - Use Wireshark to capture and analyze the packets when running the system, observe what would the server do after receiving a message from one client.

# Practise 6.2 Implement a simple だんま(danmaku) system