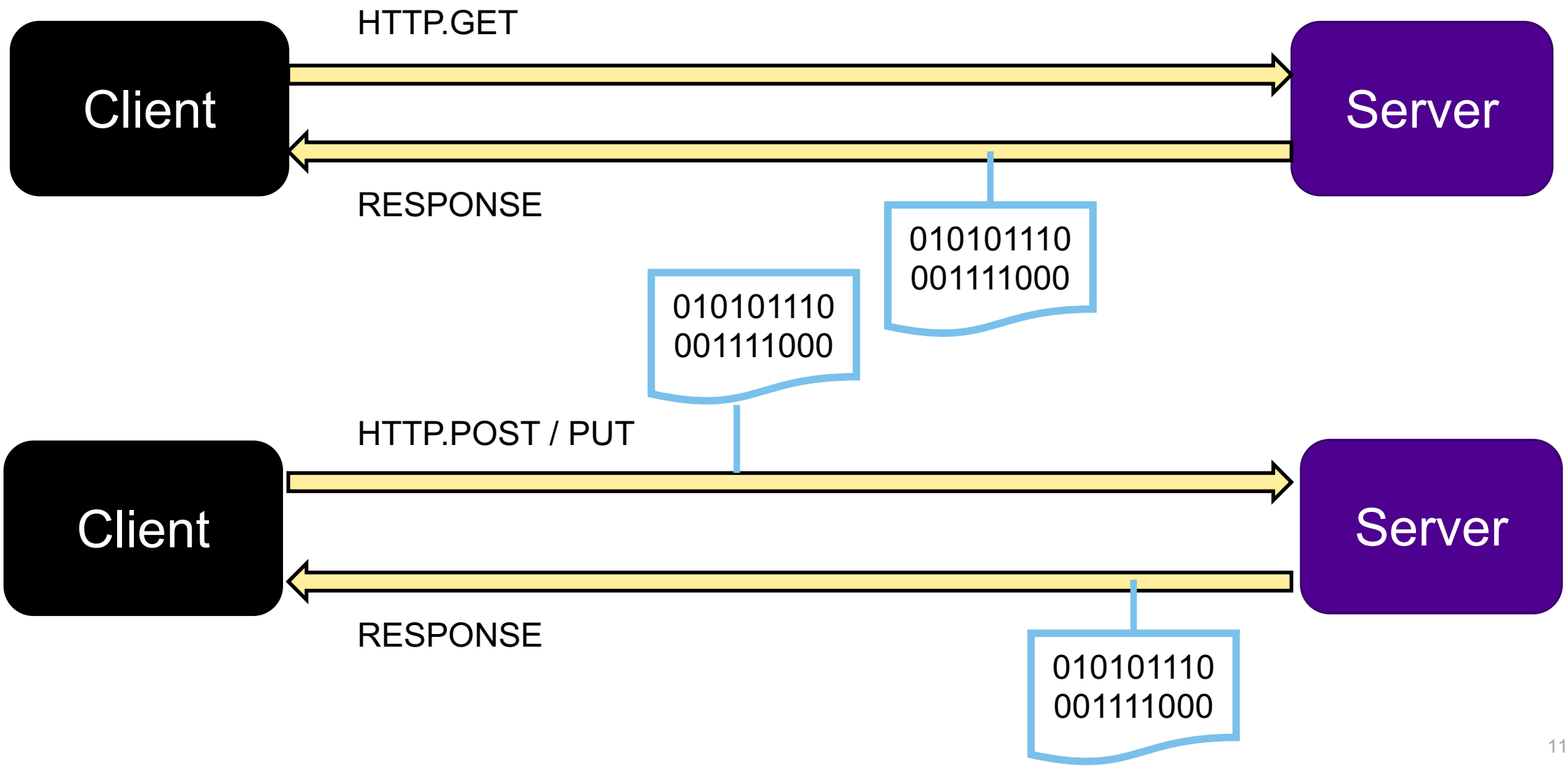
Tampere University

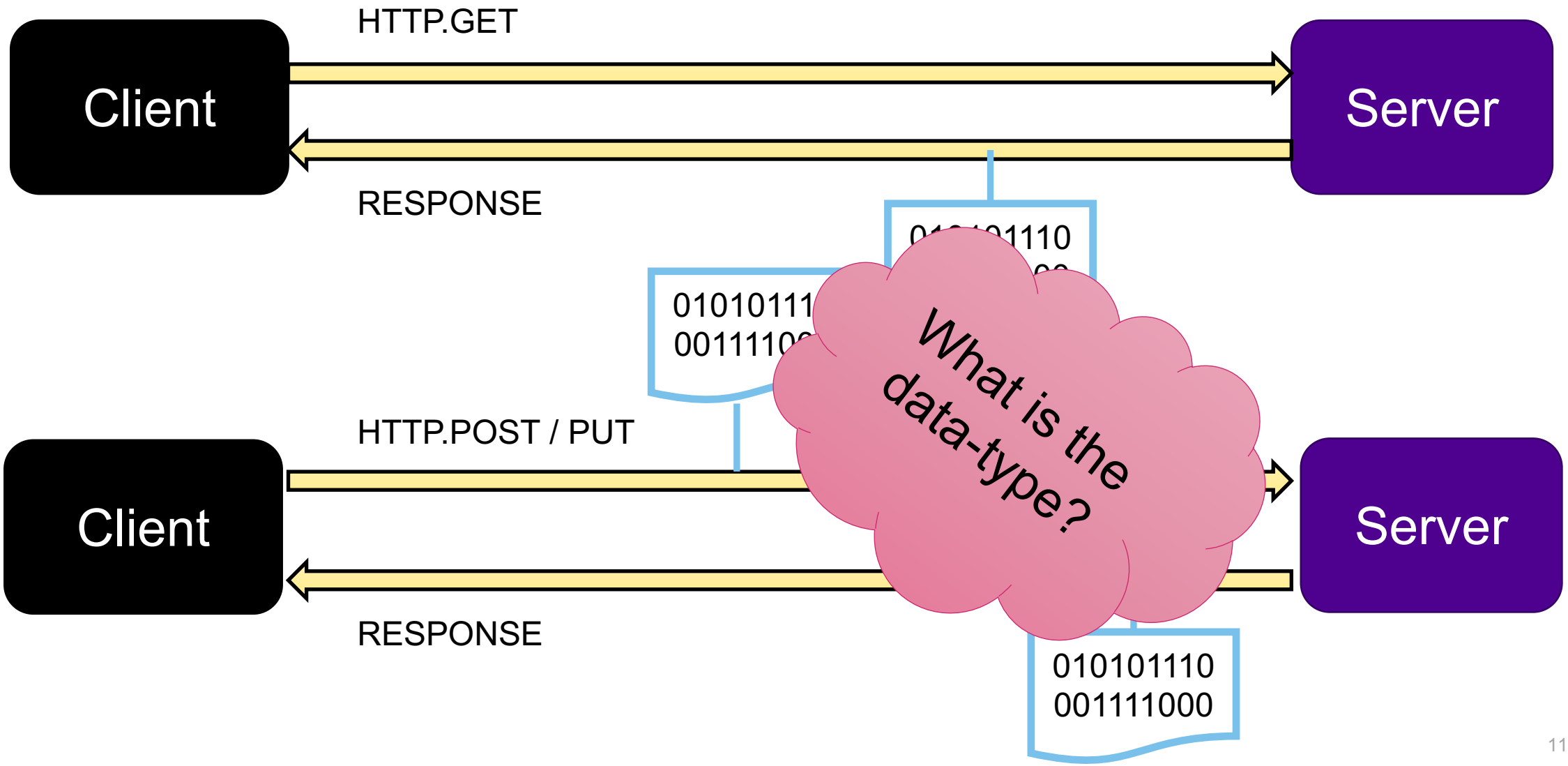
COMP.SE.140
11.10.2022

Tampere University

Very brief recap of HTTP



Very brief recap of HTTP



Lets make a small test

```
const http = require('http');
const requestListener = function (req, res) {
    console.log("METHOD: " + req.method);
    console.log("HEADERS");
    for (i in req.headers) {
        console.log(" - " + i + " : " + req.headers[i]);
    }
    if (req.method == "POST") {
        var body = '';
        req.on('data', function (chunk) {body += chunk.toString();});
        req.on('end', () => {console.log("BODY: " + body);});
    }
    res.writeHead(200);
    res.end('Hello, World!');
}

const server = http.createServer(requestListener);
server.listen(8082);
```

```
curl  -X POST 192.168.1.170:8082 -d "Hello"
```

```
METHOD: POST
HEADERS
 - host : 192.168.1.170:8082
 - user-agent : curl/7.79.1
 - accept : */*
 - content-length : 5
 - content-type : application/x-www-form-urlencoded
BODY: Hello
```

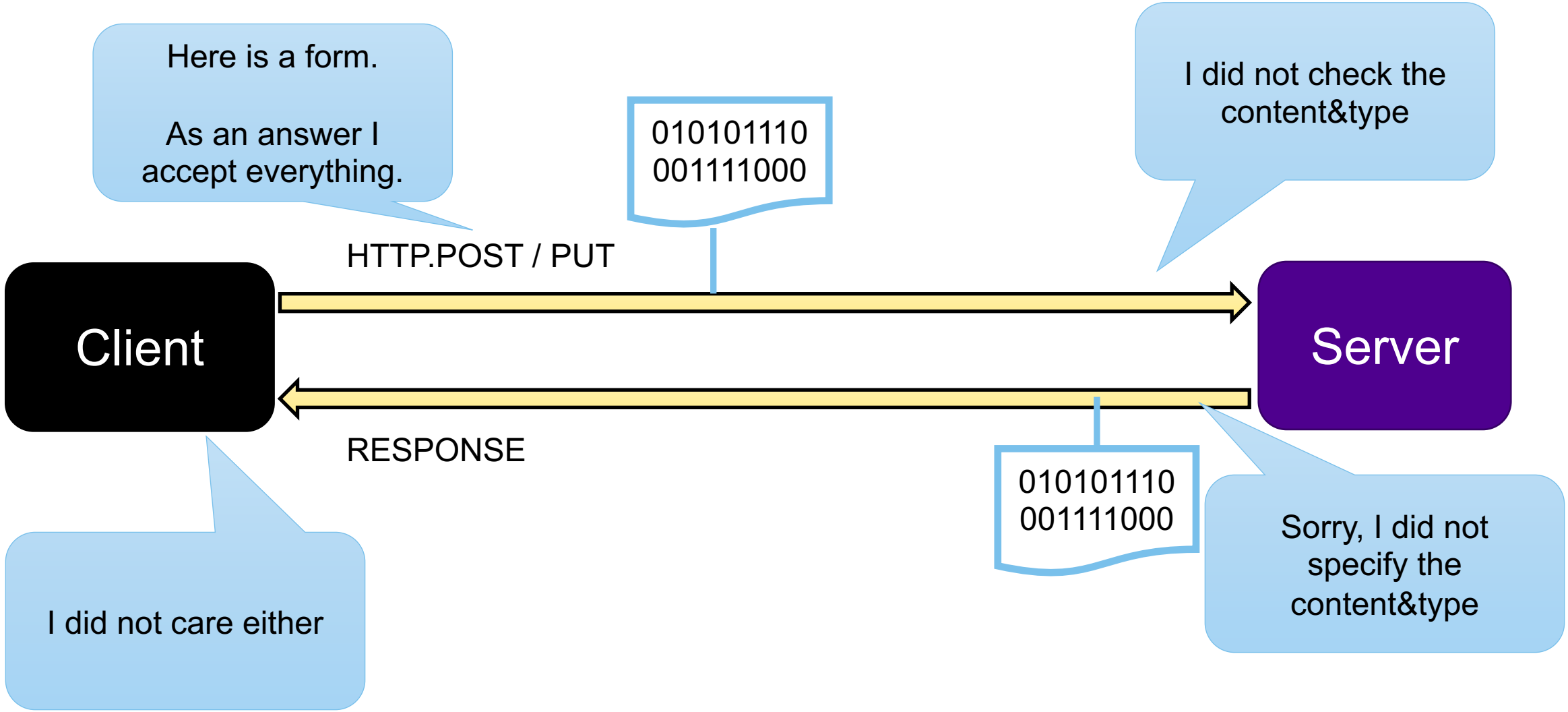
Tampere University

Debugging curl

```
$ curl -X POST 192.168.1.170:8082 -d "Hello" -v
Note: Unnecessary use of -X or --request, POST is already inferred.
*   Trying 192.168.1.170:8082...
* Connected to 192.168.1.170 (192.168.1.170) port 8082 (#0)
> POST / HTTP/1.1
> Host: 192.168.1.170:8082
> User-Agent: curl/7.79.1
> Accept: */*
> Content-Length: 5
> Content-Type: application/x-www-form-urlencoded
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Date: Fri, 07 Oct 2022 09:52:09 GMT
< Connection: keep-alive
< Transfer-Encoding: chunked
<
* Connection #0 to host 192.168.1.170 left intact
Hello, World
```

Tampere University

What happened in our case



Tampere University

From: https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods/POST

- "A POST request is typically sent via an HTML form and results in a change on the server. "

```
POST /test HTTP/1.1
Host: foo.example
Content-Type: application/x-www-form-urlencoded
Content-Length: 27

field1=value1&field2=value2
```

Tampere University

Simple HTML form

```
<HTML>

<BODY>

<form action="/doit" method="post">

 <label for="fname">First name:</label><br>

 <input type="text" id="fname" name="fname"
  value="John"><br>

 <label for="lname">Last name:</label><br>

 <input type="text" id="lname" name="lname”
  value="Doe"><br><br>

 <input type="submit" value="Submit">

</form>

</BODY>

</HTML>
```

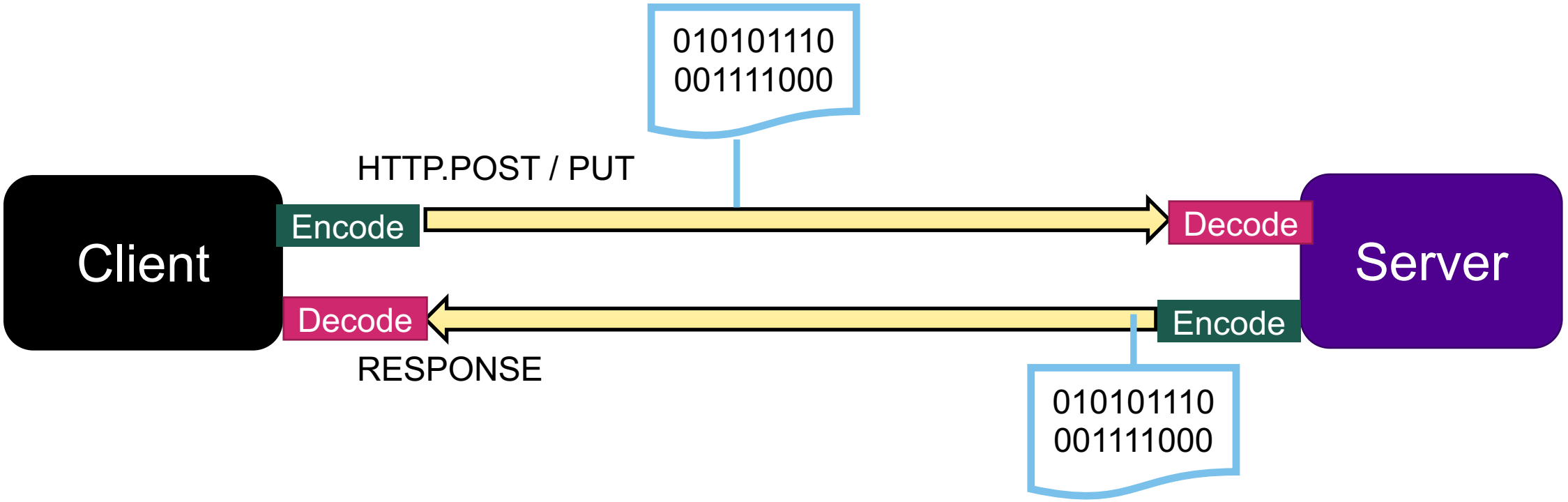
```
METHOD: POST
HEADERS
 - host : 192.168.1.170:8082
 - origin : http://192.168.1.170:8082
 - content-type : application/x-www-form-urlencoded
 - accept-encoding : gzip, deflate
 - connection : keep-alive
 - upgrade-insecure-requests : 1
 - accept :
text/html,application/xhtml+xml,application/xml;q=0.
9,*/*;q=0.8
 - user-agent : Mozilla/5.0 (Macintosh; Intel Mac OS
X 10_15_7) AppleWebKit/605.1.15 (KHTML, like Gecko)
Version/16.0 Safari/605.1.15
 - referer : http://192.168.1.170:8082/mimetest.html
 - content-length : 20
 - accept-language : en-GB,en;q=0.9
PATH = /doit
BODY: fname=John&lname=Doe
```

Question in the lecture

- What if first name is “John&”
- Answer is
  - fname=John%26&lname=Doe

  I.e. url-encoding is used

How it should be?



New test

```
const http = require('http');
const requestListener = function (req, res) {
    console.log("METHOD: " + req.method);
    console.log("HEADERS");
    for (i in req.headers) {
        console.log(" - " + i + " : " + req.headers[i]);
    }
    if (req.method == "POST") {
        var body = '';
        req.on('data', function (chunk) {body += chunk.toString();});
        req.on('end', () => {console.log("BODY: " + body);});
    }
    res.writeHead(200);
    res.end("verb=Hello&target=World");
}

const server = http.createServer(requestListener);
server.listen(8082);
```

```
$ curl -X POST 192.168.1.170:8082 -d "Hello"
verb=Hello&target=World
```

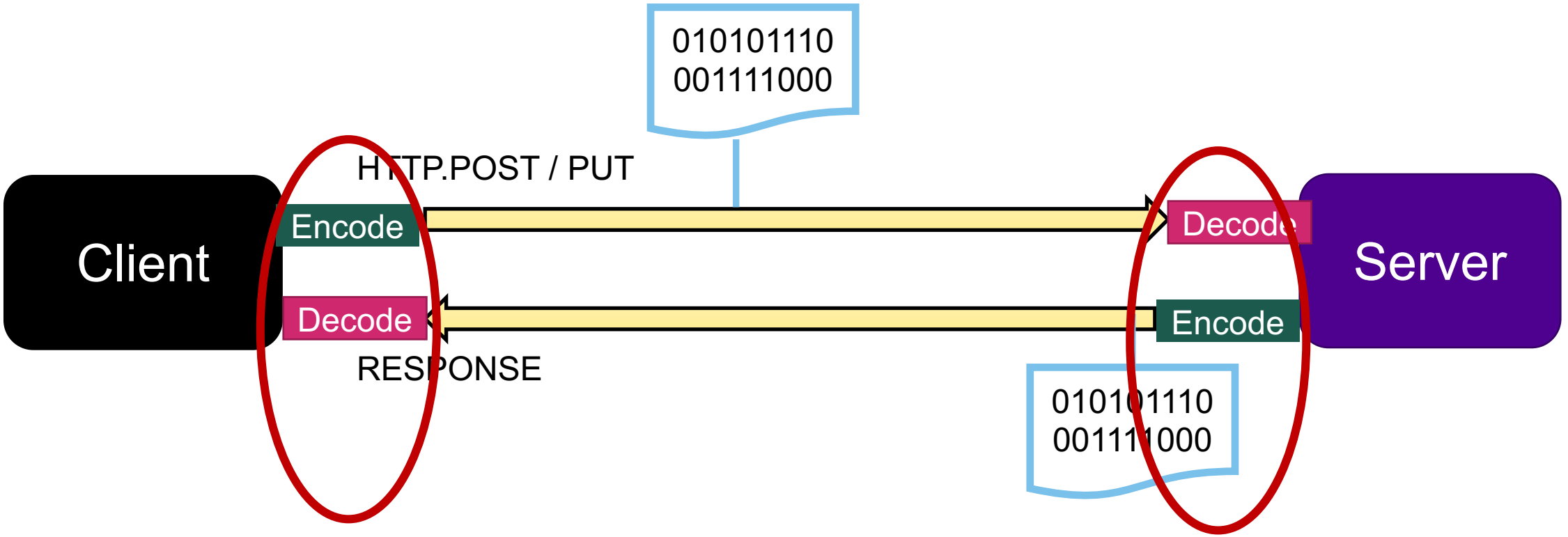
New test

```
const http = require('http');
const requestListener = function (req, res) {
    console.log("METHOD: " + req.method);
    console.log("HEADERS");
    for (i in req.headers) {
        console.log(" - " + i + " : " + req.headers[i]);
    }
    if (req.method == "POST") {
        var body = '';
        req.on('data', function (chunk) {body += chunk.toString();});
        req.on('end', () => {console.log("BODY: " + body);});
    }
    res.setHeader("content-type", "application/x-www-form-urlencoded");
    res.writeHead(200);
    res.end("verb=Hello&target=World");
}

const server = http.createServer(requestListener);
server.listen(8082);
```

```
$ curl -X POST 192.168.1.170:8082 -d "Hello"
verb=Hello&target=World
```

In your further submissions to this course be aware



If your HTTP-library does something automatically

Examples of officially registered MIME-types

application/javascript
application/json
application/ld+json
application/msword (.doc)
application/pdf
application/sql
application/x-www-form-urlencoded
application/xml
text/plain
text/css
text/csv
text/html
text/xml

Policy in this couse

- Testing of projects will require a specific content type
- You should assume that the teacher does:

```
curl –header ”accept: text/plain”
```

Unless otherwise stated

Ansible exercise

Tampereen yliopisto
Tampere University

Ansible (https://www.ansible.com)

Automation engine for

- Provisioning
- Configuration Management
- App Deployment
- Continuous Delivery
- Security Automation
- Orchestration

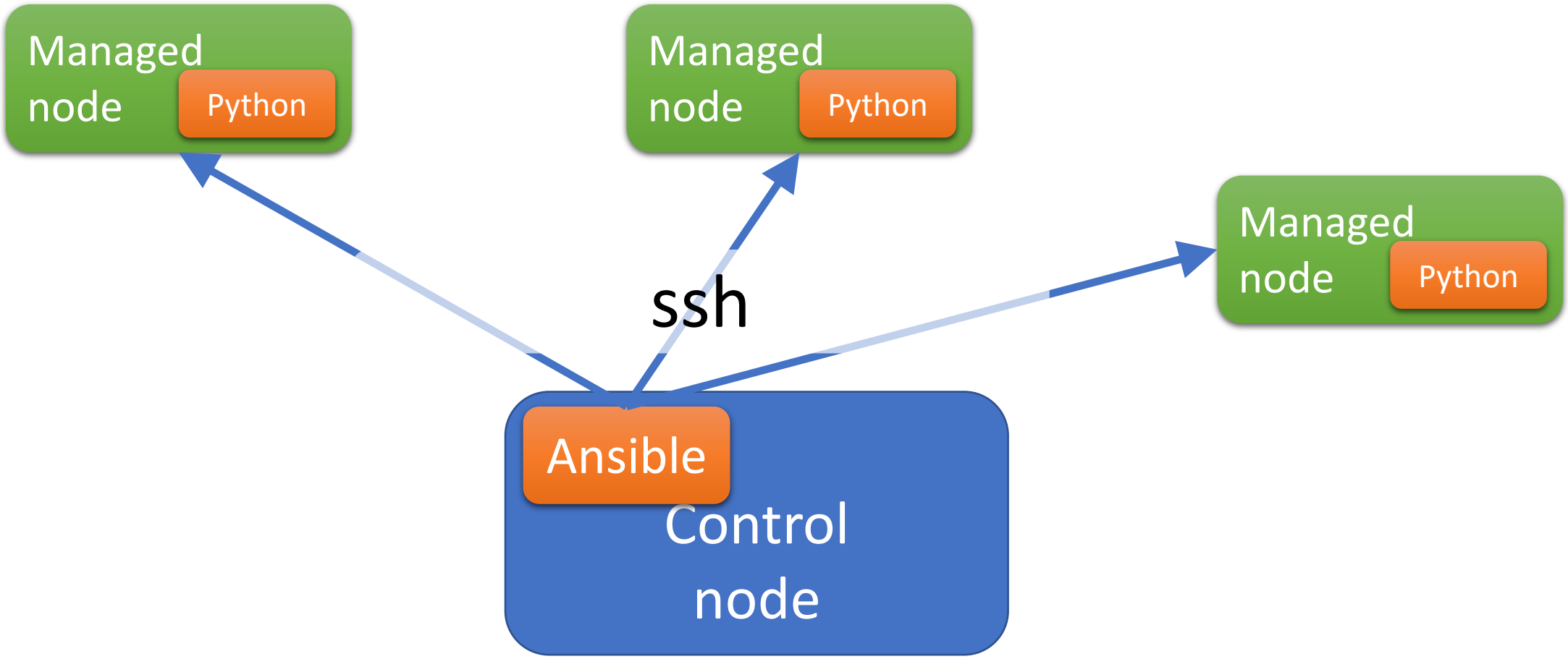
uses YAML, in the form of Ansible Playbooks

Tampereen yliopisto
Tampere University

Ansible

- Ansible works by connecting to your nodes and pushing out small programs, called "Ansible modules" to them.
- These programs are written to be resource models of the desired state of the system.
- Ansible then executes these modules (over SSH by default), and removes them when finished.
- Your library of modules can reside on any machine, and there are no servers, daemons, or databases required.
- Typically, you'll work with your favourite terminal program, a text editor, and probably a version control system to keep track of changes to your content.
- A short video:
  - https://www.ansible.com/resources/videos/quick-start-video

Architecture



Tampereen yliopisto
Tampere University

Docker containers as targets

- Since we do not enough virtual machines, lets use Docker images
- Complicates the exercise,
- but allows you to learn more about Docker

How would you use A/B-testing or HYPEX in SISU