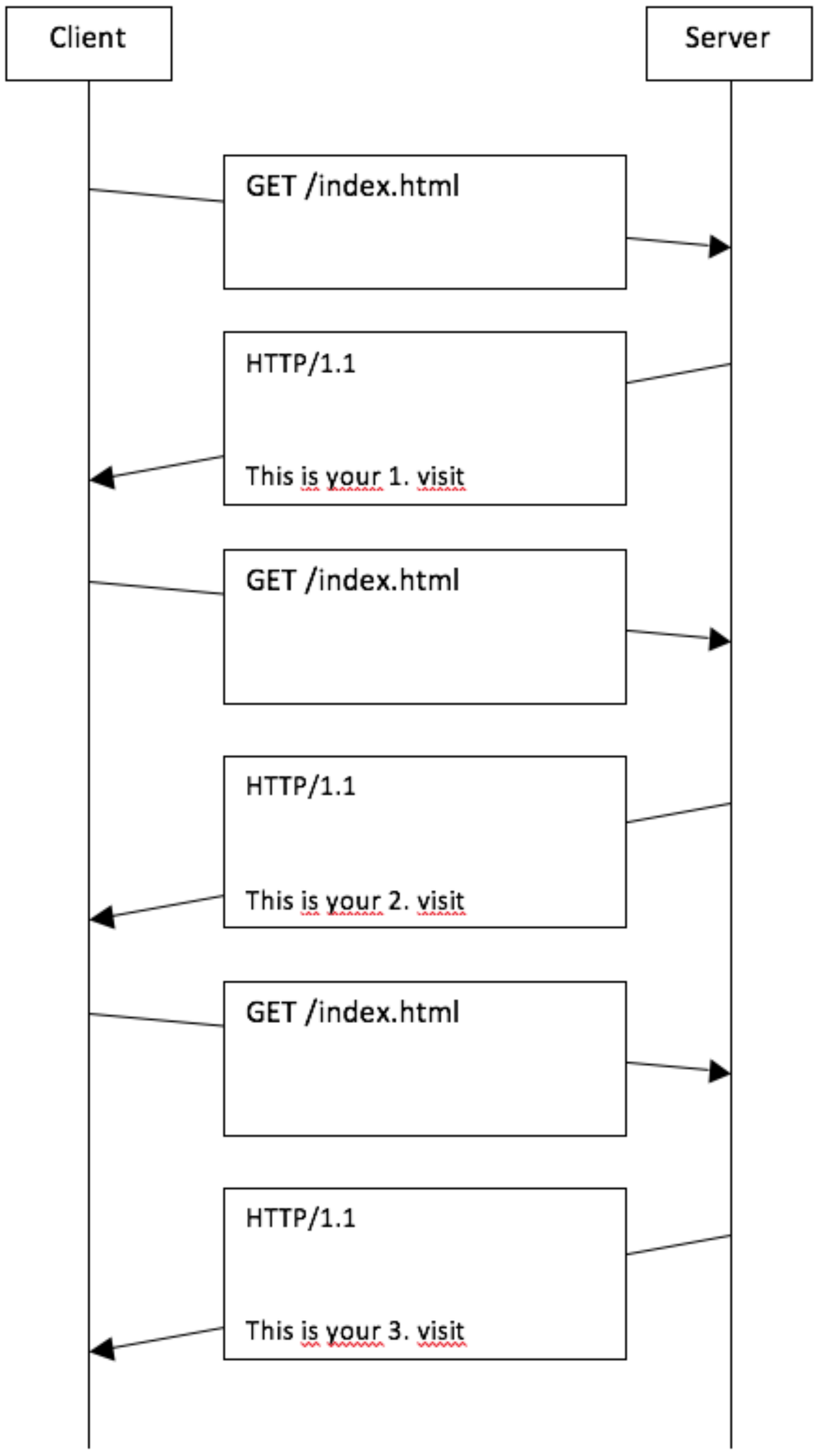
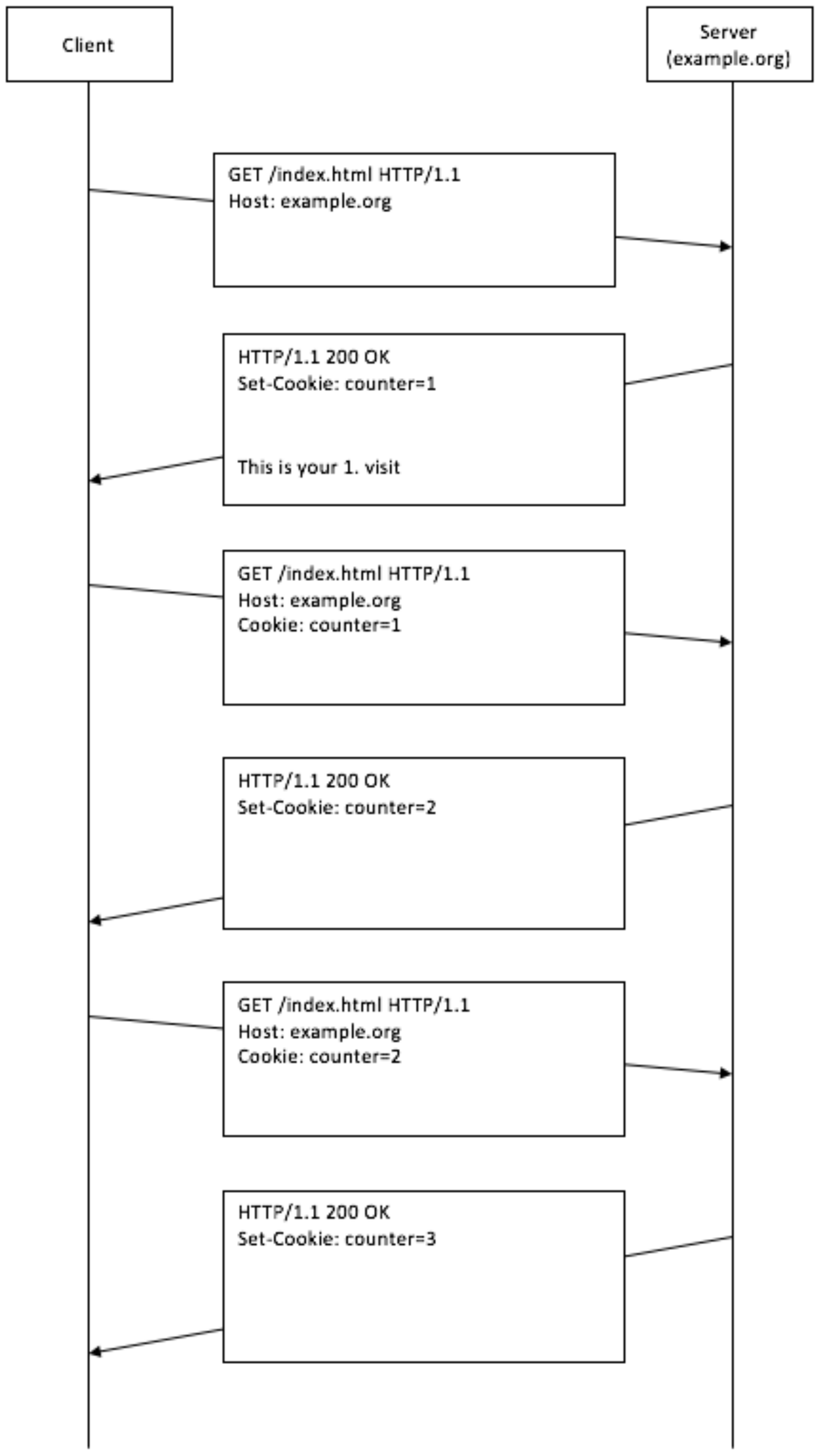
6. Tutorial

Task 1

1. What is the purpose of HTTP cookies?
2. Consider the following scenario. A client requests the same resource on the server for several times. Each time he requests a resource, the number of prior requests is returned in the resource representation. Complete the following diagram with missing HTTP headers and status codes required for implementation of the above scenario:



3. Extend the template below towards the above scenario.



```
GET /index.html HTTP/1.1
Host: example.org
```

```
HTTP/1.1 200 OK
Set-Cookie: counter=1

This is your 1. visit
```

```
GET /index.html HTTP/1.1
Host: example.org
Cookie: counter=1
```

```
HTTP/1.1 200 OK
Set-Cookie: counter=2
```

```
GET /index.html HTTP/1.1
Host: example.org
Cookie: counter=2
```

```
HTTP/1.1 200 OK
Set-Cookie: counter=3
```

Task 2

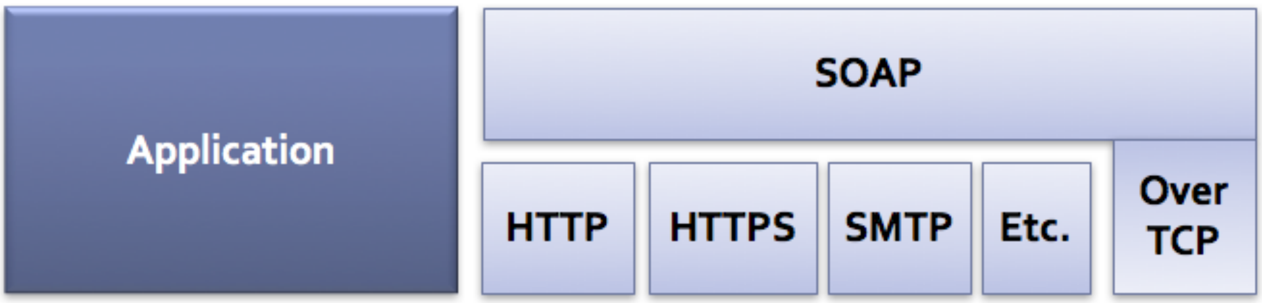
1. Answer the following questions.
   a. What is SOAP?

   b. Where would you situate SOAP within the Internet Protocol stack?
   c. Which parts are in a SOAP Message?
2. Under http://pauline.informatik.tu-chemnitz.de/SoapWebService/Service.asmx[1] you will find a simple Web service, which is able to compute a sum of two integers (the *Add* operation). Implement a client for sending SOAP1.2 requests to the above service based on the template below. Print the result of the calculation to the console.

Tutorial6-Task2-Template.zip
Shared on Dropbox

Tutorial6-Task2-Solution.zip
Shared on Dropbox

SOAP is a protocol for the exchange of information in a distributed environment. SOAP messages are encoded as XML documents and can be exchanged using a variety of underlying protocols



- SOAP is a protocol for the exchange of information in a distributed environment. SOAP messages are encoded as XML documents and can be exchanged using a variety of underlying protocols
- A SOAP message is encoded as an XML document, consisting of an `<Envelope>` element, which contains an optional `<Header>` element, and a mandatory `<Body>` element. The `<Fault>` element, contained within the `<Body>`, is used for reporting errors.

---

[1] Accessible from university network or over VPN
[2] Send manually constructed SOAP messages over HTTP – do not use any code-generation tools or third party libraries

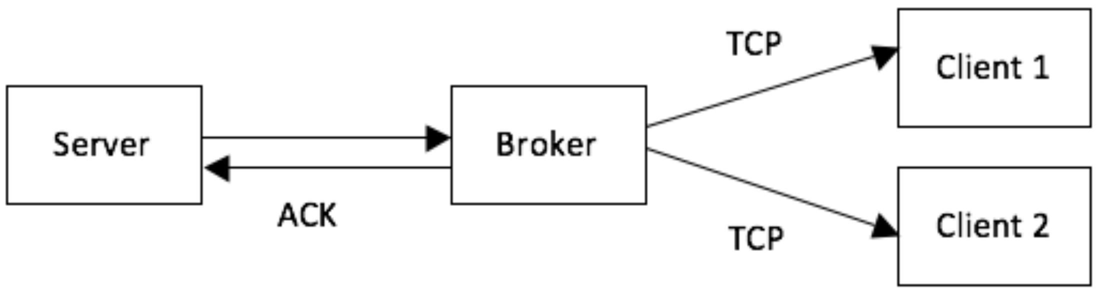
Assignment 1

Complete the *SoapClient.Invoke* method of the *template below*, which should be able to send requests to ANY SOAP service using given service URL, service namespace, operationName and named parameters list (for simplicity request and response parameters can be only strings or integers). Test your class on the service above and the *ConcatenatorService* under http://pauline.informatik.tu-chemnitz.de/ConcatenatorService/ConcatenatorService.asmx

Tutorial6-Assignment1-Template.zip
Shared on Dropbox

Assignment 2

The project *below* contains a distributed system including a server, a broker and two clients. The Server sends regular stock price updates to the broker, which distributes them to the both clients. The broker is often overloaded and therefore the server should retransmit the message if broker doesn't acknowledge the message receipt (robust one-way MEP pattern). The communication between the broker and clients should follow simple one-way MEP pattern.



1. Extend the *ReceiveMessage* method of the broker to send SOAP-based receipt notifications back to the server (BUSY or RECIEVED)[1].
2. The broker should not be able to read the contents of some SOAP messages. Update the server and client 1 to use shared secret to encrypt and decrypt contents of SOAP messages[2].
3. Client 2 should not be equipped with any encryption/decryption facilities and therefore ignore encrypted messages. Make server send appropriate SOAP headers to signal the need of decrypting content before processing it.

Tutorial6-Assignment2-Template.zip
Shared on Dropbox

Tutorial6-Assignment2-Solution.zip
Shared on Dropbox

---

[1] Cf. http://www.w3.org/TR/xmlp-scenarios/#S5 for examples

[2] You can use Encryption.Encrypt and Encryption.Decrypt methods