HTTP and CoAP Comparison Lab

NOTE

Please read carefully the whole document before starting with your laboratory or asking questions.

Objective

To understand the differences between the RESTfull protocols HTTP and CoAP.

For this, CoAP and HTTP client tools will be run within an Ubuntu Desktop machine sending requests to CoAP and HTTP test servers in the internet. The student will compare the behaviour of both protocols based on observations of logs captured with the Wireshark tool.

Running the Virtual Machine

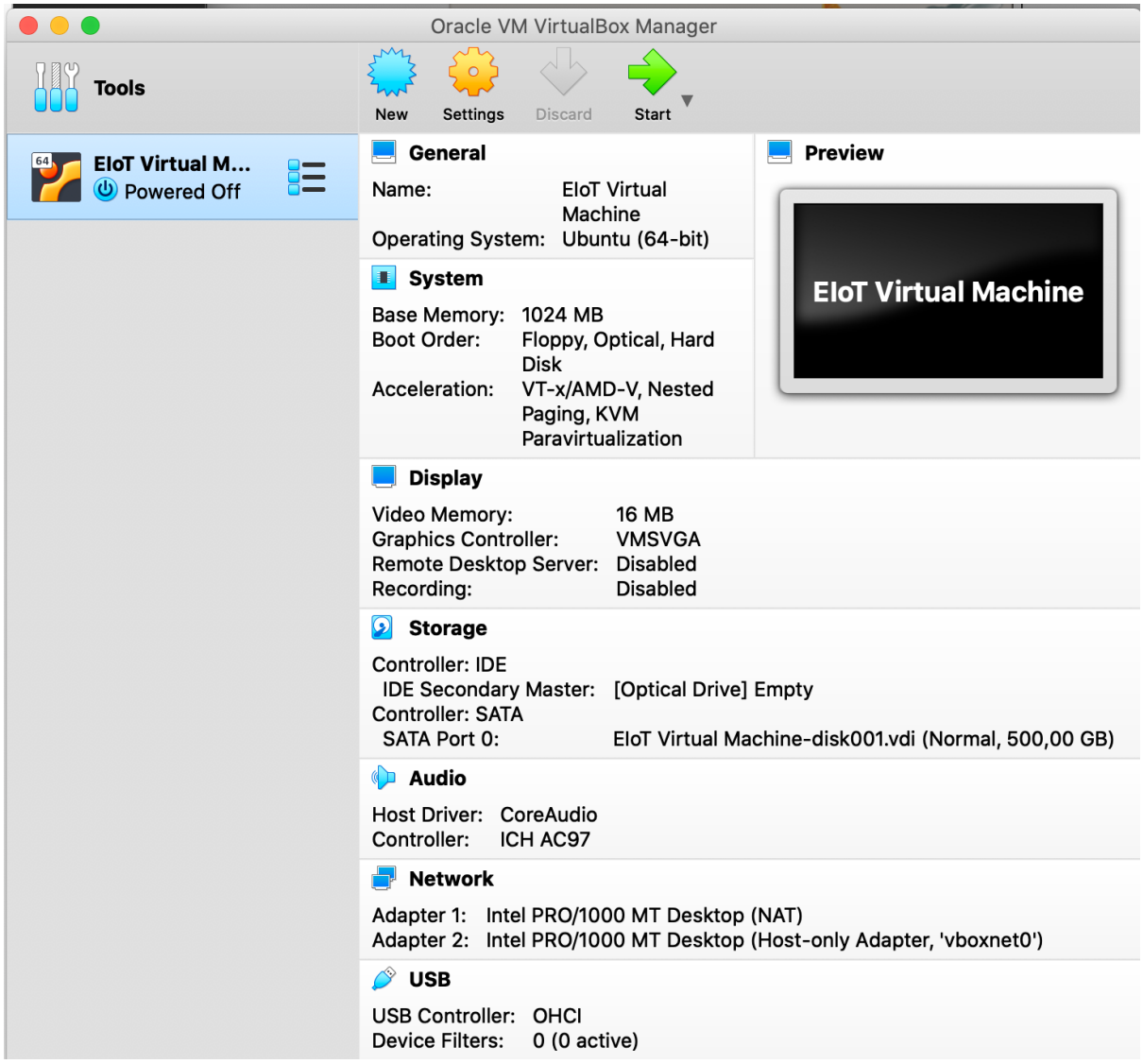
Students should install VirtualBox from Oracle and download the given Virtual Machine image (file with OVA extension) in its own laptop computer. The Virtual Machine image requires around 10 GB of hard-drive space.

If during the import of the Virtual Machine by VirtualBox appears the E_INVALIDARG error, it may be due to insufficient space on the host hard drive, or due to importing the source image from a USB memory. VirtualBox may temporarily need a larger (than 10 GB) amount of space in the host hard drive for importing the image.

The Virtual Machine contains the necessary tools for running the laboratory. It is an Ubuntu Desktop machine with Wireshark.

Importing and running the virtual machine in Virtual Box can be done following the instructions found in https://docs.oracle.com/cd/E26217_01/E26796/html/qs-import-vm.html

After importing the Virtual Machine, VirtualBox Manager will display the imported machine on its left hand side list and show the parameters on its right hand side, as shown below:



By pressing the Start button (green arrow pointing right) on the menu bar, VirtualBox will boot up the EIoT Virtual Machine (based on Ubuntu Desktop) in a new window. Log in to the Ubuntu Desktop machine using the password provided.

The password for superuser and eiot user accounts of the Ubuntu machine is ***Nowowiejska15/19***

If you are experiencing problems with the password, it might be due to a misconfiguration of the keyboard layout between the host and virtual machines. In such cases, the easiest way to proceed is

to open a text editor in the host computer, write the password, copy such text in the host clipboard, and then paste it on the virtual machine login box.

Running the HTTP Client tool in Ubuntu

On the quick-launch application bar in Ubuntu Desktop there is a quick access to the Terminal and to the WireShark application. Their icons are displayed below. These two are the only applications that will be used for this laboratory.

The top icon is the Linux terminal, the second is the Eclipse IDE for C/C++, and the third is Wireshark application. In this laboratory, we will not be using Eclipse.

In order to run the HTTP client tools in the Ubuntu Desktop, you need to first open a new Linux terminal.

The name of the HTTP Client tool we use in this laboratory is curl. curl is a tool to transfer data from or to a server, using one of the supported protocols (DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, MQTT, POP3, POP3S, RTMP, RTMPS, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, TELNET and TFTP). The format to execute the curl command for an HTTP request can be seen below.

```
curl --location --request method URI
```

where `method` refers to one of the HTTP methods: GET, POST, PUT, CREATE, etc, and `URI` is an HTTP unique resource identifier for the object of the request.

More information about the curl command can be found in:

https://curl.se/docs/manpage.html

Running the CoAP Client tool in Ubuntu

In order to run the CoAP client tools in the Ubuntu Desktop, you need to first open a new Linux terminal. You may continue using the same terminal window from the HTTP Client tool.

Next, you need to set up the path location of the linked library by typing the following command in the Terminal:

```
export LD_LIBRARY_PATH=/usr/local/lib
```

Finally, verify that the CoAP client command utility runs correctly. You may do so by typing the following command with no parameters.

```
coap-client
```

The output of the command invokation should show you in the Terminal a list of available parameters to the `coap-client` command.

In order to send a single CoAP request from the coap-client to a specific resource, you should use the command:

```
coap-client -m method URI
```

where `method` can be either `get, post, put or delete, fetch, patch or ipatch`, and `URI` is the URI of the resource being queried. Without any additional flags, the CoAP client will issue a confirmable request. Only by adding the flag **`-N`** between the `method` and the `URI` to the command line, the CoAP client will issue a non-confirmable request.

After the invocation, the Ubuntu shell will display a text representation of the CoAP response obtained from the CoAP server, or the error code given by CoAP. The code `4.04` means that the CoAP server could Not Found the requested resource. The full list of error codes can be found in RFC 7252 from the IETF.

More information on the CoAP client utility can be found here:

http://manpages.ubuntu.com/manpages/bionic/man5/coap-client.5.html

Lab Procedure and Questions

Recommendations before starting:

- Make sure having a fresh (new) launch of the Ubuntu Desktop machine and that both HTTP and CoAP client utilities work as expected in Ubuntu.
- Start capturing traffic using Wireshark in the enp0s3 interface before executing any HTTP or CoAP client request command.
- Stop capturing traffic using Wireshark 15 seconds after observing no more traffic.
- Save the Wireshark capture file separately for each point.
- Analyse the behaviour of the traffic for each point below, before going forward.
- Filter the Wireshark log as to only show messages from either CoAP/UDP or HTTP/TCP, depending on your needs, but only to the selected server.

For each question, explain and argument your answers based on the IoT lecture given. Include a message sequence diagram (https://en.wikipedia.org/wiki/Sequence_diagram) of the HTTP/TCP and CoAP/UDP messages exchange only between these two machines and screenshots of your Wireshark captures for each question below.

Procedure:

1. Send a HTTP GET request to the URI: `http://postman-echo.com/time/now`
    a. What is the response displayed in the Linux terminal? How many characters were shown (counting spaces and punctuation marks)?
    b. How many HTTP messages were exchanged?
    c. How many bytes does the payload of the transport layer protocol needed to encode the HTTP request? How may for the HTTP response?
    d. Divide the number of characters displayed in the screen (question 1.a) by the number of bytes included as the transport layer protocol payload of the HTTP response (question 1.d).
2. Send a (confirmable) CoAP GET request to the URI: `coap://coap.me/test`
    a. What is the response displayed in the Linux terminal? How many characters were shown (counting spaces and punctuation marks)?
    b. How many bytes does the payload of the transport layer protocol needed to encode the CoAP request? How may for the CoAP response?
    c. Divide the number of characters displayed in the screen (question 2.a) by the number of bytes included as the transport layer protocol payload of the HTTP response (question 2.b).
    d. Considering your answers from questions 1.c. and 2.b., which protocol is more efficient in terms of encoding? Why?

The grade of your laboratory depends only on the answers given in a written report to the following questions.