LAB 7 - Application Layer Protocols, HTTP, Designing a Custom Application Layer Protocol

An Application Layer Protocol defines how two computer processes, that may be running on different computers, might pass messages to each other. The protocol defines rules that determine when a message should be sent, how that message is formatted, and how that message should be responded to when it is received.

The Application Layer Protocol also determines which Transport Layer Protocol it will utilize to send its messages across the network. FTP (File Transfer Protocol) uses two TCP ports, one to send control messages and the other to send and receive file data. TFTP however uses a single UDP port to send files. The Application Layer Protocol should be designed to be as loosely coupled to the needed Transport Layer functionality as possible. When these two layers are loosely coupled it is easier to swap out the Transport Layer protocol without making many changes to the Application Layer.

The design of an Application Layer Protocol is a vast subject. The following is not required reading for this lab, but to have a proper understanding of Application Layer protocols it would be helpful to read:

http://www.highteck.net/EN/Application/Application_Layer_Functionality_and_Protocols.html (Gives an Idea of the Functionality Required by the Application Layer, and how the lower layers can help service that functionality)

https://gradeup.co/application-layer-protocols-dns-smtp-pop-ftp-http-i-ba1194bd-c5ab-11e5-9dcb-5849de73f8e1 (A list and explanation of common Application Layer protocols)

Prerequisites

Lab 4 - We set up Network Address Translation and Port Forwarding here will be helpful but not required.
Lab 5 - Taking a Deeper Look into the TCP connection will be helpful but not required.

Backing Up a VM

Throughout these labs we'll be making configuration changes to our Ubuntu Server installation. It can be easy to break our server, but we don't have to worry about that, VirtualBox allows us to backup the current state of our server. It's strongly suggested that you backup your virtual server now. Here are a few online resources that cover this process:
https://www.youtube.com/watch?v=xqNIvyZlHts
https://www.osradar.com/how-to-backup-vms-on-virtualbox/

Fiddler

In this lab we'll use a tool called Fiddler to view the HTTP protocol headers. When you sign in it will lead you through an initial tutorial. Read through the tutorial.
https://www.telerik.com/download/fiddler-everywhere (~90 MB)

The HTTP Protocol

HTTP is the most common application layer protocol used on the internet today. HTTP is the communication protocol that is the central foundation of the World Wide Web.
FreeCodeCamp gives a good introduction to the HTTP protocol, paying close attention to its messages and message format:
https://www.freecodecamp.org/news/http-and-everything-you-need-to-know-about-it/

Here is a much more in depth introduction to the HTTP protocol. We want to read up to the "HTTP Request Message" & "HTTP Response Message" sections:

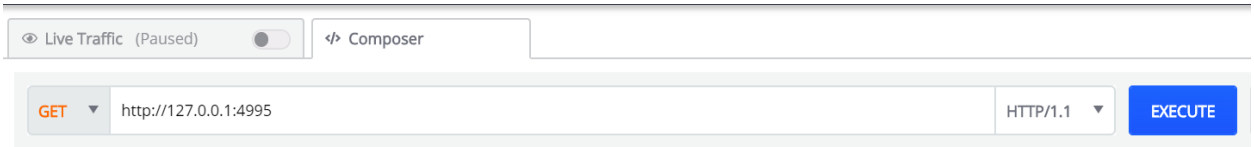
https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/http_basics.html

HTTP Request & Response Message

Tutorials Point has a good explanation of the HTTP Request Message header and its fields:
https://www.tutorialspoint.com/http/http_requests.htm

And the related tutorial on HTTP Response Message headers:
https://www.tutorialspoint.com/http/http_responses.htm

Use Fiddler to make an HTTP GET request to your Ubuntu Server's Web Server. http://127.0.0.1:4995



Find the RAW text for both your HTTP Request header and your HTTP Response header. Save the text from each.

OTHER HTTP REQUESTS

The HTTP Protocol Defines more Request Methods than just GET and POST. **Task**: Search online for a list of all Request Methods. For each method, briefly describe its purpose.

REQUIREMENTS FOR AN APPLICATION LAYER PROTOCOL

According to RFC 4101 in order to understand the requirements needed for an Application Layer Protocol we must ask several questions about that protocol:

1. What problem is the protocol trying to achieve?

    - The most critical task that a protocol model must perform is to explain what the protocol is trying to achieve. This provides crucial context for understanding how the protocol works, and whether it meets its goals. Given the desired goals, an experienced reviewer will usually have an idea of how they would approach the problem and, thus, be able to compare that approach with the approach taken by the protocol under review.
2. What messages are being transmitted and what do they mean?
    - The HTTP Protocol uses GET and POST messages among others.
3. What are the important, but unobvious, features of the protocol?

https://tools.ietf.org/html/rfc4101

Designing a Custom Application Layer Protocol: DIR_LIST Protocol

The *DIR_LIST* protocol will be a protocol designed to list the contents of a particular directory on a remote machine.

1. *What Problem is this protocol trying to solve?*
   The *DIR_LIST* protocol will be designed to be used in a client-server architecture. The client wants to list all the files in a specific directory on the server. The client will need to send a request message to the server. The server will receive and process that message, and send a response message back to the client, which will display the files in the desired directory.

2. *What messages are being transmitted and what do they mean?*
   Your protocol will need to define a specific format for each of these type of messages:

   - **REQUEST_DIR_LIST** Message Format

TYPE	ID	COMMAND	DIR	END

				MESSAGE
REQ	0001	LIST	../subdir1/	END

Since We want to send a REQuest to the server we can make the first field in our message “REQ”
Our applications may need to keep track of an ID number for each Request and Response
Our message is sent when we want to LIST the files in a specific DIR of the remote computer.
We may need to know when a REQUEST message ends, so we’ve defined the end of a message as anytime we see the keyword END.

- **RESPONSE_DIR_LIST** Message Format

  Design a Response Message Format. It must be a response to the above Request. These fields need some way to be able to list all the files in a specific directory. Think how you can accomplish this if there is a variable amount of files to be listed.

Preparing For Future Labs

Start Researching the FTP protocol. All of the links referenced in this lab were sourced from a simple Google search for “HTTP Message Headers”. You can search something similar to research the FTP protocol for future labs.

Deliverables:

Deliver to Canvas The following:

1. Raw Text from the HTTP Request Header generated when you used Fiddler to connect to your Web Server.
2. Raw Text from the HTTP Response Header.
3. A list of every HTTP Request Method, other than GET and POST, with a 1-sentence description of what each method does.

4. Your design for the RESPONSE_DIR_LIST message format.

Place the four deliverables into a single document, then upload it to the LAB 7 Submission on Canvas.