|  |  |
| --- | --- |
| Networking and the raspberry PI.  11th November 2019.  Prof. Randall Brouwer | Daniel Ackuaku Haleluya Woldesenbet |

**Part 1**

|  |  |  |
| --- | --- | --- |
| **Property** | **Lab PC SB 076-19** | **Raspberry Pi #5** |
| Host Name | SB 076-19-70251 |  |
| IPv4 Address | 153.106.113.107 | 153.106.112.139 |
| MAC Address | B0-83-FE-B2-14-3C | b8-27-eb-cf-14-ca |

The Wireshark screen documents all the data packets received by the device, detailing the packet number, the time of receipt, the packet source the destination, transfer protocol and length of the packet.

Received packets are color coordinated due to the kind of transfer protocol.

No activity is shown when the filter is applied because the raspberry pi is not initiating any data transfer on the network.

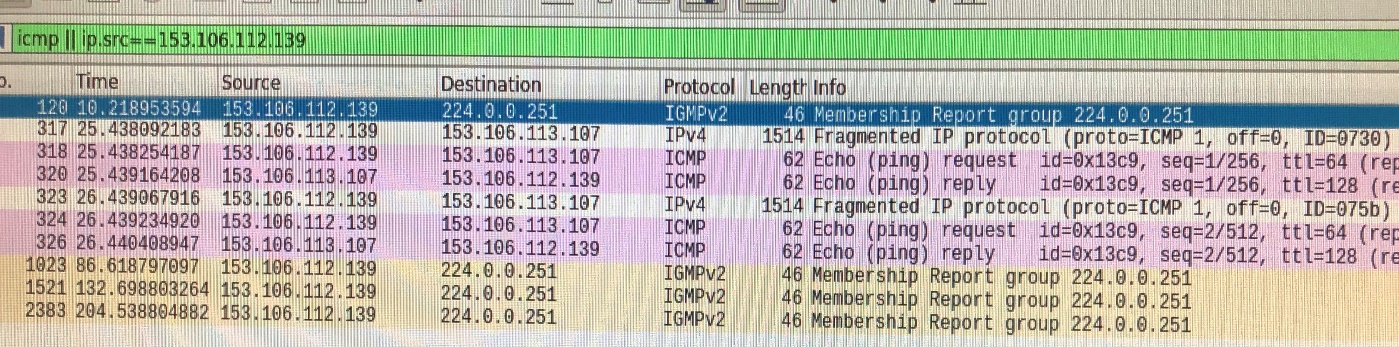
**Part 2**

1. No activity is shown when (see handout for explanation).
2. Six messages in total we created from the ping instruction. 3 requests and 3 replies.
3. Graphical representation of the ICMP, IP and Ethernet II protocols.

|  |  |
| --- | --- |
| **Layer** | **Internet Control Message Protocol (ICMP)** |
| Type | 0(Echo ping reply) |
| Code | 0 |
| Checksum | 0xbfd [correct] |
| Identifier (BE) | 1034 (0x040a) |
| Identifier (LE) | 1034 (0x040a) |
| Sequence number (BE) | 1 (0x0001) |
| Sequence number (LE) | 256 (0x0100) |
| Request frame | 11442 |
| Response time | 0.600ms |
| Timestamp form ICMP data | 10/31/2019 11:20:23.842450000 EDT |
| Timestamp form ICMP data (relative) | 0.001540936 seconds |
| Data (48 bytes) |  |

|  |  |
| --- | --- |
| **Layer** | **Internet Protocol (IP)** |
| 0100= Version | 4 |
| 0101 = Header Length | 20 bytes (5) |
| Differentiated Services Field | 0x00 (DSCP: CS0, ECN: Not-ECT) |
| Total Length | 84 |
| Identification | 0x7ab5 (31413) |
| Flags | 0x0000 |
| Time to live | 128 |
| Protocol | ICMP (1) |
| Header checksum | 0xzb28 [] |
| Source | 153.106.113.107 |
| Destination | 153.106.112.139 |

|  |  |
| --- | --- |
| **Layer** | **Ethernet II** |
| Destination | Raspberry\_cf:14:ca (b8:27:eb:cf:14:ca) |
| Address | Raspberry\_cf:14:ca (b8:27:eb:cf:14:ca) |
| LG Bit | 0 |
| IG Bit | 0 |
| Type | IPv4 (0x0800) |
| Source | Dell\_b2:14:3c (b0:83:fe:b2:14:3c) |
| Address | Dell\_b2:14:3c (b0:83:fe:b2:14:3c) |
| LG Bit | 0 |
| IG Bit | 0 |
| Type | IPv4 (0x0800) |



Part 3: Working with Telnet.

1. We had about 15 messages sent all of the same type – Telnet data.
2. Telnet allows you to connect the raspberry pi over the network and create a command prompt window over the network. We don’t think this is a very efficient way of communicating. Since anyone with access to you IP address could connect to that raspberry pi (provided they could crack the user name and password). However, we do see instances where this could be very useful. Supposing the device you needed to connect to isn’t located in close proximity to the end user it allows for remote connectivity.
3. Once the -ls command was entered telnet transmits the names addressees of the directories found on the raspberry pi over the network as data strings.

Part 4: Working with SSH:

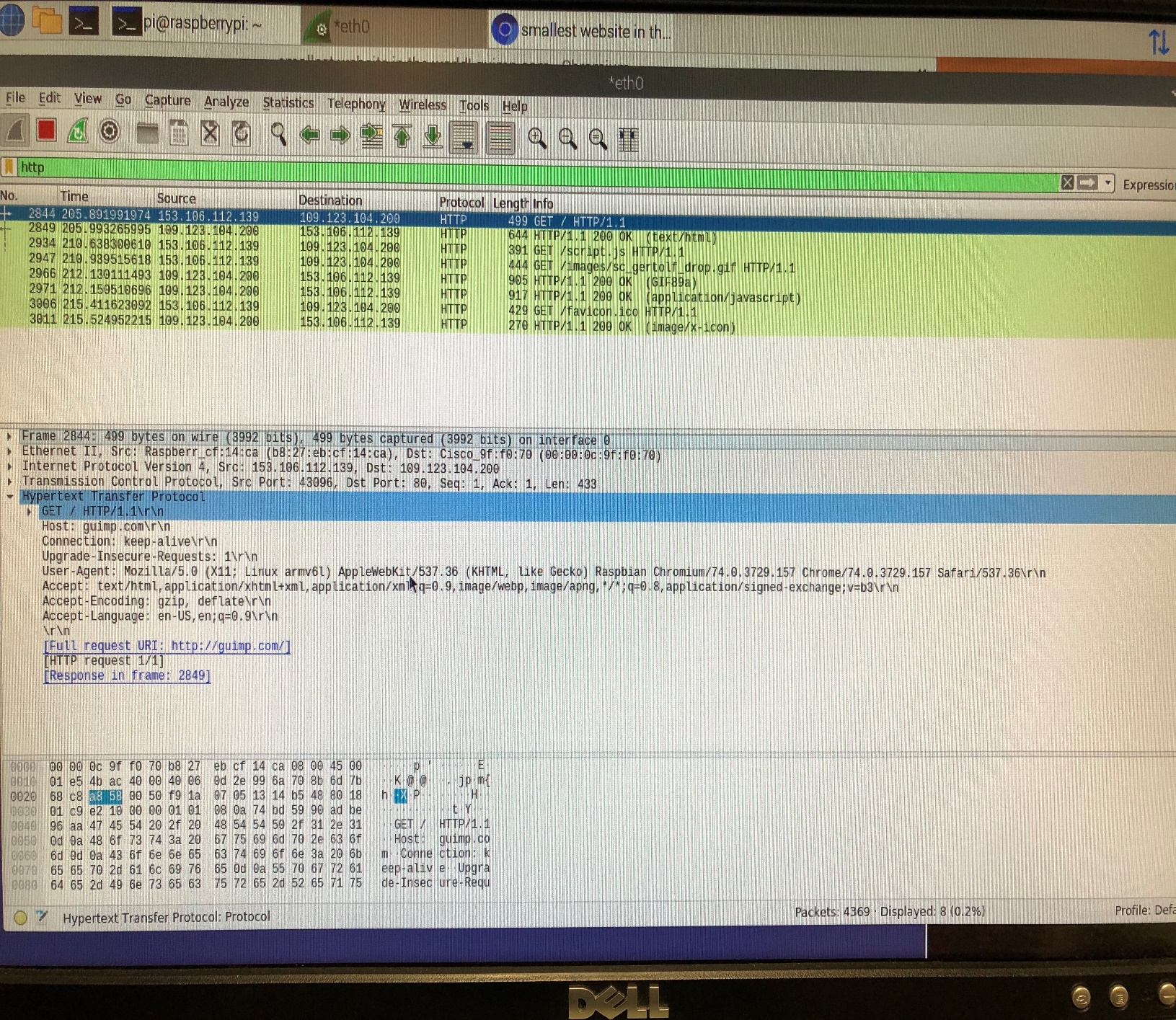
1. This time when we created the connection there were no visible messages between the raspberry pi and the PC. The -ls command also did not yield any visible connections in Wireshark. None of the data character that were visible in Wireshark while suing Telnet were visible when the ssh protocol was used. It is hard to tell the number of packets that were produced during the ssh protocol communication because Wireshark didn’t show them but, if we were making an educated guess, we would say that based on the Telnet protocol, yes there were individual packets for each letter.

Part 5: Working with ARP

1. The first portion of message in the Wireshark window asks on the network which device has a specific IP address (the IP address of the switch)
2. The next portion reports the MAC address if the switch to the Raspberry PI.

Part 6: Playing with Wireshark

We decided to take a look at the messages generated from accessing websites. We tried this with the Calvin website, but the website has a large multimedia footprint. Instead, we used the world smallest website (guimp.com) to investigate the http messages / data sent when we access data from a website.



Using the http filter, we were able to deduce that anytime you visit a website, your device issues a get command to the server on which that website is stored asking for the URL (HTTP/1.1). Next, if the website has a valid address the server returns the request has succeeded (HTTP/1.1 200 OK). The device and server repeat this get / confirm success transaction until the all the necessary files for the website have been read. In this case our website just had a simple JavaScript file with a gif, an icon for the tab bar and an image.

Lab Reflections

1. We enjoyed the lab it was a welcome departure from the DE2 boards it was nice to analyse the network capabilities of the raspberry pi. This lab gave us a lot of insight to rudimentary networking an just an understanding of the sheer scope of messages being trafficked while devices might be seemingly idle.
2. It would be a good idea to keep the lab.
3. It would be interesting to see what students would be able to come up with when given eh time an freedom to experiment with technologies like the raspberry pi. We think it would be even more beneficial if the lab was split evenly between using the DE2 board and the raspberry pi.