Benediction Bora
WES 237A: Introduction to Embedded System Design (Winter 2026)
Lab 4: Network Communication
Due: 2/16/2026 11:59pm

To report and reflect on your WES 237A labs, please complete this Post-Lab report by the deadline by submitting the following 2 parts:

- Upload your Lab 4 report, composed by a single PDF that includes your in-lab answers to the bolded questions in the Google Doc Lab and your Jupyter Notebook code. You could either scan your written copy or simply type your answer in this Google Doc. **However, please make sure your responses are readable.**
- Answer two short essay-like questions on your Lab experience.

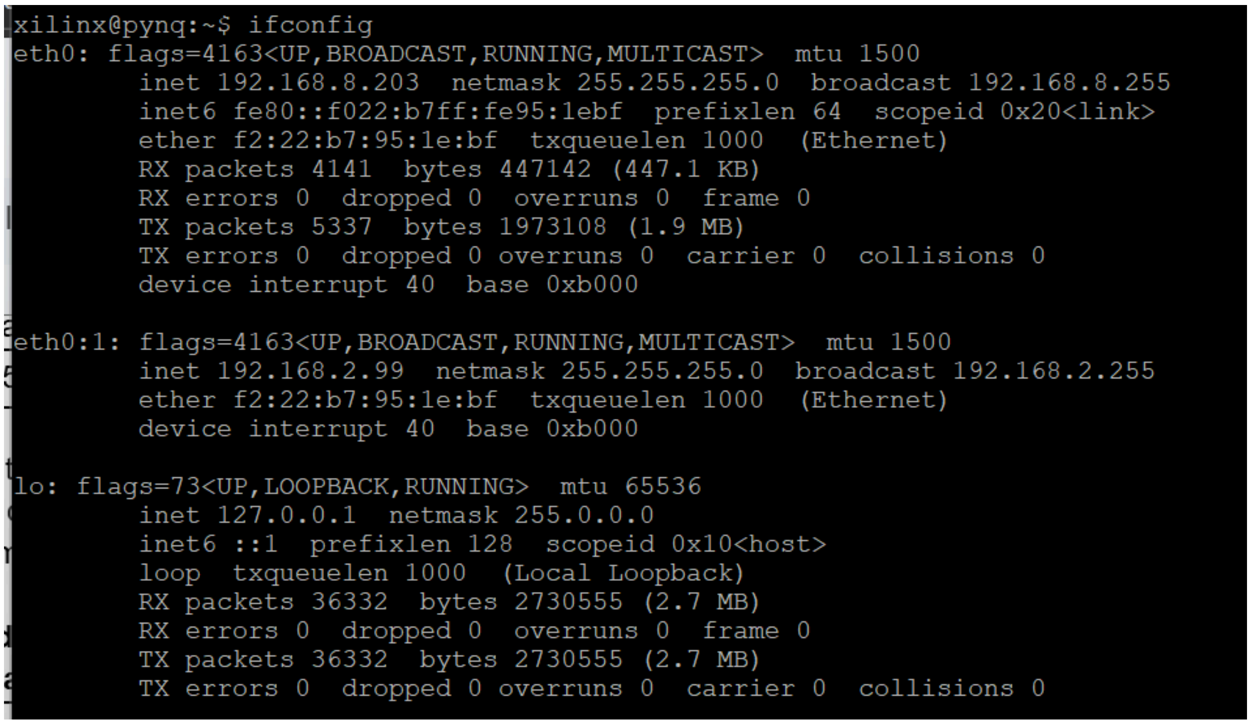
All responses should be submitted to Canvas. Please also be sure to push your code to your git repo as well.

Locating IP Addresses of Devices in your Network

1. Connect the PYNQ board to the network switch over Ethernet.
2. Run '$ ifconfig'. This is the *Interface Configuration* command and will tell you the different interfaces on your PYNQ board.
    a. **How many IPv4 addresses are assigned to the board? What is the IPv4 address assigned to the 'eth0' or Ethernet interface? What is the netmask of this address?**
        i. Note: `eth0: 1` or `usb0` is a virtual interface through the USB cable. This assigns your board an IP address over USB. This is a static IP address, so you can always reach your board from this IP address over USB.

IPv4: 192.168.8.203, netmask: 255.255.255.0

3. Connect your computer to the campus WiFi (UCSD-PROTECTED)
4. On your computer, open a command prompt and run `$ ipconfig` on Windows and `$ ifconfig` on MAC/Linux (it may take a second to connect, so wait a minute and then run the command)
    a. **How many IPv4 addresses are assigned to this machine? What IPv4 address has the same netmask as the PYNQ board?**



```
xilinx@pynq:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.8.203  netmask 255.255.255.0  broadcast 192.168.8.255
        inet6 fe80::f022:b7ff:fe95:1ebf  prefixlen 64  scopeid 0x20<link>
        ether f2:22:b7:95:1e:bf  txqueuelen 1000  (Ethernet)
        RX packets 4141  bytes 447142 (447.1 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 5337  bytes 1973108 (1.9 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
        device interrupt 40  base 0xb000

eth0:1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.2.99  netmask 255.255.255.0  broadcast 192.168.2.255
        ether f2:22:b7:95:1e:bf  txqueuelen 1000  (Ethernet)
        device interrupt 40  base 0xb000

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 36332  bytes 2730555 (2.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 36332  bytes 2730555 (2.7 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5. Right now, your local machine and your PYNQ board form a network! However, we're more interested in networking two PYNQ boards together rather than your local machine and your PYNQ board. Luckily, every device hooked up to the switch is assigned an IP address on this network. That means we can communicate with any other board in the class. **Below, compile all the IP addresses of the PYNQ boards in your group.**

192.168.8.203, 192.168.8.162

6. To access your PYNQ board Jupyter notebooks, go to <PYNQ-IP>:9090

PYNQ-PYNQ Communication with Python

- Here, we're going to implement basic message-sending functionality in Python between two PYNQ boards.
- Download 'sockets_example.ipynb'
- Go through and complete the code. Answer the following questions.
    - **What does `socket.SOCK_STREAM` mean (hint: search the documentation link in the notebook)?**

socket.SOCK_STREAM tells python that we want to use a TCP socket and creates a communication channel

    - **What is the order of operations for starting a client socket and sending a message?**

1. Create TCP the socket, 2. Create message to send, 3. Send message, 3. Close socket

    - **What is the order of operations for starting a server socket and receiving a message?**

1. Create TCP socket, 2. Bind the socket to the server port, 3. Accept connections, 4. Receive message, 5. Close socket.

Wireshark

1. On your local machine (or lab machine), install Wireshark

   If you are on Windows:

   a. Open the Firewall and Network Protection
   b. Click ‘Allow an app through firewall.’
   c. Click ‘Change Settings’
   d. Scroll down to ‘Python’
   e. Select all ‘Python’ applications and all ‘Public’ boxes for each ‘Python’

✓ Python	✓	☐	✓	No
✓ Python	✓	☐	✓	No

   f. Open the program `IDLE (Python 3.7 64-bit)`

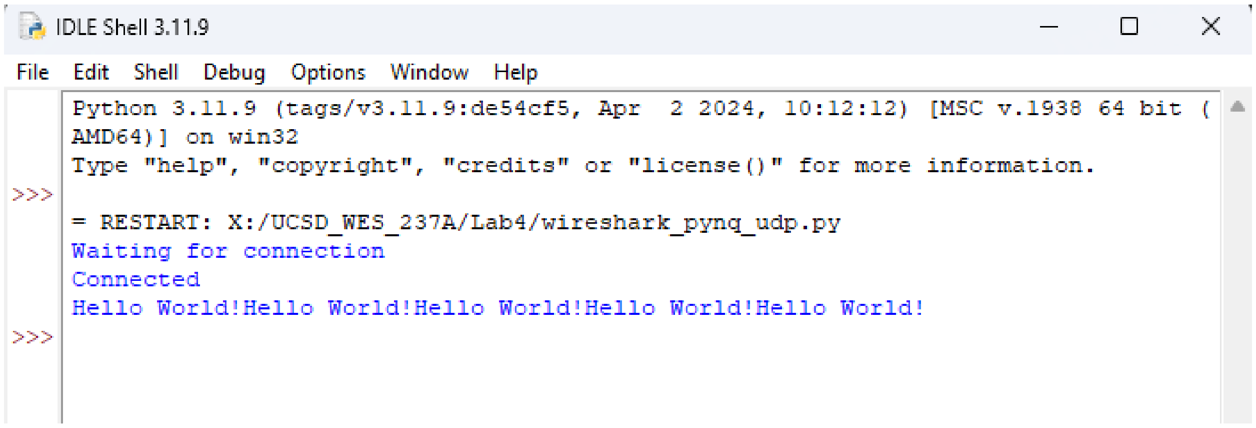
   If you are on Linux / MacOS:

   g. Run the following command:
      sudo ufw status
   h. If you get Active status, run this:
      Sudo ufw allow 12345/tcp
   i. Run idle3 from the terminal (Linux) or search for IDLE in applications (Mac)
2. Click File->New File and paste the following code **(Check for tab v space errors when copying and pasting)**

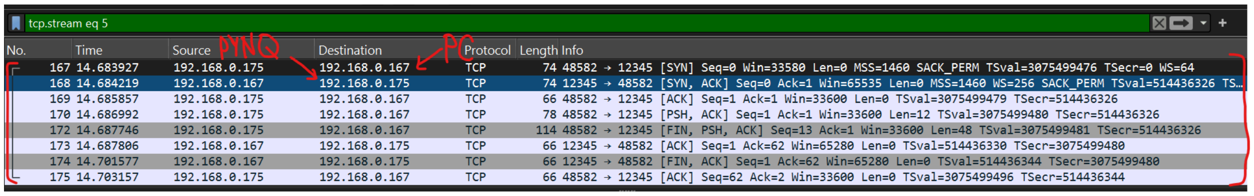
```
import socket
import time
import signal
import sys

def run_program():
	sock_l = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
	sock_l.bind(('0.0.0.0', 12345))
	sock_l.listen()
	print('Waiting for connection')
	conn, addr = sock_l.accept()
	print('Connected')
	with conn:
		data = conn.recv(1024)
	print(data.decode())

if __name__ == '__main__':
	original_sigint = signal.getsignal(signal.SIGINT)
	signal.signal(signal.SIGINT, exit)
	run_program()
```

3. Save the file, then select ‘Run -> Run Module’. This is a slight variation of your server. It is waiting on port 12345 on the local lab machine.
4. From your PYNQ board, connect your client to
   a. IP: local lab IP
   b. Port: 12345

5. Send the message “Hello world\n”



```
Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: X:/UCSD_WES_237A/Lab4/wireshark_pynq_udp.py
Waiting for connection
Connected
Hello World!Hello World!Hello World!Hello World!Hello World!
>>>
```

6. You should see it displayed in the Python terminal
7. Now open Wireshark
8. Double click ‘Wi-Fi’ or ‘Ethernet’, depending on how you connected to the network. You’re now capturing a trace of the network traffic between your machine and the PYNQ board, via the router. Look at a few of the traces. Notice which are between your PYNQ board and the local machine (check the IP addresses) and which involve the router. There will only be a difference if you also connect the PYNQ board directly to your local machine.
9. Where it says ‘Apply a display filter’ at the top, type ‘tcp’
10. 

```
No.   Time        Source        Destination   Protocol Length Info
  167 14.683927   192.168.0.175 192.168.0.167 TCP      74 48582 → 12345 [SYN] Seq=0 Win=33580 Len=0 MSS=1460 SACK_PERM TSval=3075499476 TSecr=0 WS=64
  168 14.684219   192.168.0.167 192.168.0.175 TCP      74 12345 → 48582 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=514436326 TS…
  169 14.685857   192.168.0.175 192.168.0.167 TCP      66 48582 → 12345 [ACK] Seq=1 Ack=1 Win=33600 Len=0 TSval=3075499479 TSecr=514436326
  170 14.686992   192.168.0.175 192.168.0.167 TCP      78 48582 → 12345 [PSH, ACK] Seq=1 Ack=1 Win=33600 Len=12 TSval=3075499480 TSecr=514436326
  172 14.687746   192.168.0.175 192.168.0.167 TCP     114 48582 → 12345 [FIN, PSH, ACK] Seq=13 Ack=1 Win=33600 Len=48 TSval=3075499481 TSecr=514436326
  173 14.687806   192.168.0.167 192.168.0.175 TCP      66 12345 → 48582 [ACK] Seq=1 Ack=62 Win=65280 Len=0 TSval=514436330 TSecr=3075499480
  174 14.701577   192.168.0.167 192.168.0.175 TCP      66 12345 → 48582 [FIN, ACK] Seq=1 Ack=62 Win=65280 Len=0 TSval=514436344 TSecr=3075499480
  175 14.703157   192.168.0.175 192.168.0.167 TCP      66 48582 → 12345 [ACK] Seq=62 Ack=2 Win=33600 Len=0 TSval=3075499496 TSecr=514436344
```

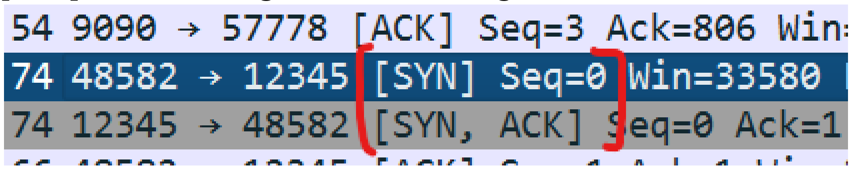
PC IP: 192.168.0.167

PYNQ Modem assigned IP: 192.168.0.175

11. Repeat steps 9-11
12. Check the packet trace for any changes
    a. **What is the sequence number of the TCP SYN segment that is used to initiate the TCP connection between the PYNQ board and the local machine? What is it in the segment that identifies the segment as a SYN segment?**

Sequence Number: 0 (relative sequence number)
Sequence Number (raw): 2799253429

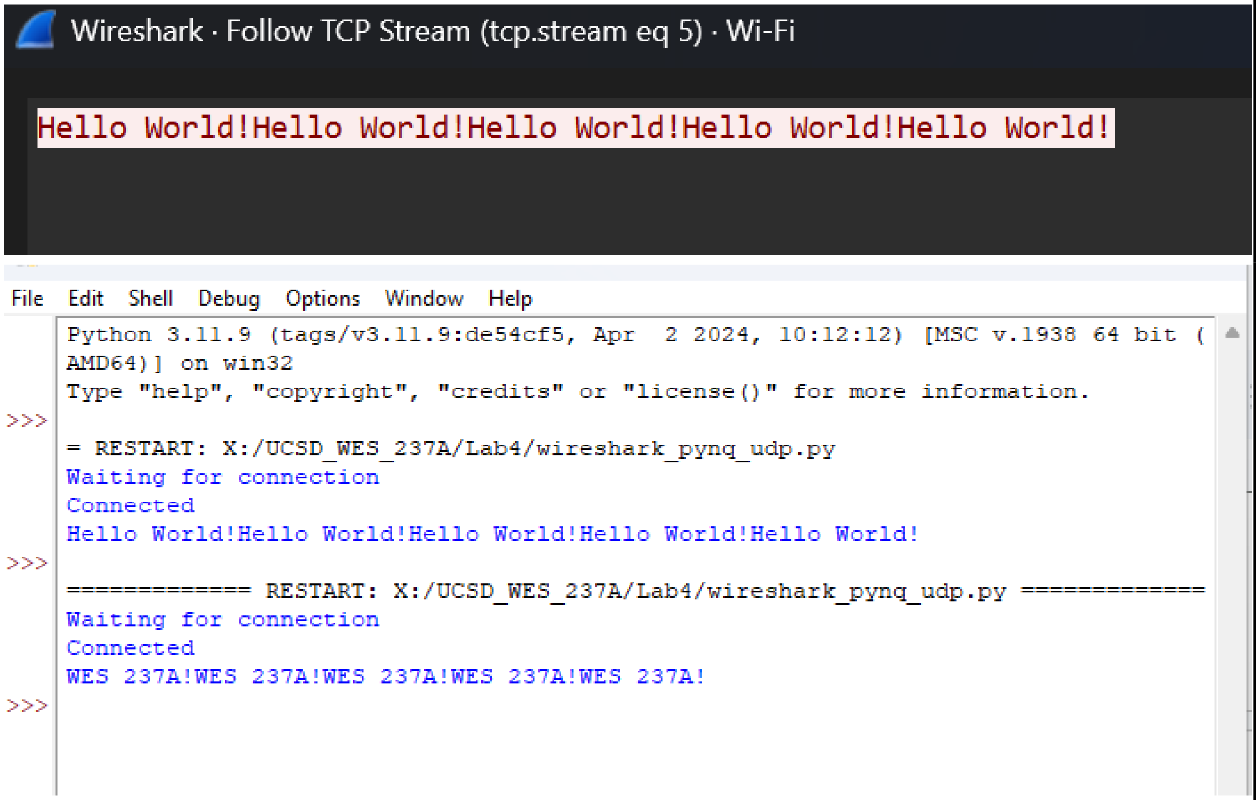
[SYN] identifies segment as SYN segment



```
54 9090 → 57778 [ACK] Seq=3 Ack=806 Win
74 48582 → 12345 [SYN] Seq=0 Win=33580
74 12345 → 48582 [SYN, ACK] Seq=0 Ack=1
```

    b. Right-click this trace and select ‘Follow->TCP Stream’

c. **Repeat a few times with different messages. Describe what's happening in the 5-10 steps of the TCP sequence for this communication. You can refresh your TCP flags here.**


Wireshark · Follow TCP Stream (tcp.stream eq 5) · Wi-Fi

```
Hello World!Hello World!Hello World!Hello World!Hello World!
```

```
Python 3.11.9 (tags/v3.11.9:de54cf5, Apr  2 2024, 10:12:12) [MSC v.1938 64 bit (
AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: X:/UCSD_WES_237A/Lab4/wireshark_pynq_udp.py
Waiting for connection
Connected
Hello World!Hello World!Hello World!Hello World!Hello World!
>>>
============= RESTART: X:/UCSD_WES_237A/Lab4/wireshark_pynq_udp.py =============
Waiting for connection
Connected
WES 237A!WES 237A!WES 237A!WES 237A!WES 237A!
>>>
```

[SYN] - First SYN initializes request to receive packets from PYNQ
[SYN ACK] - Second segment gets the packets and Acknowledges receipt from PYNQ
[PSH] - Fourth segment routes the data packets
[FIN] - Seventh segment closes socket
[ACK] - Last segment acknowledgement of closure