NWEN302 LAB1 REPORT - David Burrell 300209541

The files I made changes in are **network_layer.py, link_layer.py** and **myswitch.py**

No mininet topology was used/changed from the one provided in part a

The requirements I believe I achieved during part a of the lab are:

- ○ Each node should have a full address table containing the other nodes MAC to IP address mappings.
- ○ Request/Solicitation packets are not sent when an appropriate entry already exists in the address table
- ○ Use standardised ARp and NDP headers with the correct values.
- ○ The address table is updated on receiving a request or solicitation message.
  Address entries timeout.
- ○ Address table is of limited size.

And requirements of part b of the lab achieved are:

- After 30 seconds the switch's address table should have complete mappings for all attached hosts.
- Subsequent unicast messages hsould only be forwarded on the correct switch interface, host receiving only multicast/broadcast traffic, and traffic addressed to them
- Packets should not be altered by the switch.
- Address entries timeout
- Address table is of limited size

Implementation

Part a: uses ExpiringDict as a data structure to hold IPv4/IPv6 to MAC mappings, this was chosen as it simplistically implements the advanced requirements of the lab.
When asked by the network layer to return or find a MAC address mapping, the program does not block waiting, and instead returns immediately, this was less of a choice and more of a complexity versus remaining time issue, it was easier to leave as it was, rather than implement a blocking procedure.
The program differentiates between ARP and NDP headers, and deals with them appropriately.
The program sends request ARP or advertisement NDP when required.

Part b: Again using ExpiringDict to maintain the IP/MAC mappings within the switch, a simple implementation that also satisfies advanced lab requirements.
The Packets destined to an unknown MAC are broadcast on all except the incoming interface, this implementation was already written in the basic myswitch.py provided, it just had to be worked around to add the mappings and check the mappings.

Reflection:
A lot of these solutions came after an arduous amount of time spent trying to parse what the lab brief was asking of us and the specifics of each requirement were split across multiple headers inside the brief.
The implementations in part a aren't working as intended as of my last run of them before hand in, this was mainly due to the realisation that all the IPv4 work done was not being used as testing the hosts communicating begins with IPv6 and I didn't have time to figure out when or if it even attempts an IPv4 packet, my attempt at implementing NDP in IPv6 led me to almost running out of time.
The way the brief is written led me to work on the IPv4 and ARP implementations to begin with and I had made the assumption that these would be being used when the hosts run the part a program, thus I am unsure if they behave correctly, I am confident in my implementations otherwise.
Using ExpiringDict to cover the requirements of timeout and limited size was an idea gathered from another student in passing, I would credit them if I knew who they were.
Many problems arose from the use of the VM, being limited to one workstation was a pain, and getting used to running mininet and the hosts was not an enjoyable experience, when the xterms would close for the most simplistic reasons, such as being dragged, or pressing backspace.
I found it annoying that the code supplied within the Virtual Machine provided was different to the one supplied on the lab page in the archive file, which had significantly more commented information on what we were supposed to be doing. The bare bones version in the VM was insufficient.
The only problems I encountered when working on part b were surrounding the VM and attempts to follow the command line arguments supplied in the various readmes, as such I was never able to install switchyard using venv. This and most of the other problems simply came from not having permissions to run certain commands.
I was unable to run the tests provided at the bottom of
https://github.com/jsommers/switchyard/blob/master/examples/exercises/learning_switch/learning_switch.rst
And I couldn't install wireshark to be able to see if my implementation even works, it seems sound enough to me, looking specifically at the flowchart on the page and making sure the switch does those steps.
.https://github.com/jsommers/switchyard/blob/master/examples/exercises/learning_switch/learning_switch.rst
Again the implementation of ExpringDict was used for its simplicity in how it satisfies the requirements of timeouts and limited sizing within a single data structure.

The only testing done was during part a, and it was a simple implementation of print statement tests, using the mininet hosts and having the network layer print some information depending on which type of header arrived, initially this was a more basic implementation of printing if a message was even received. Obviously this was not successful enough as I didn't notice the IPv6 messages were being sent first but doing all the IPv4 implementation first, which worked for a short time before I added NDP handling. Part b testing was non existent due to problems stated above with the VM environment and the various commands not being permitted.

I would've liked to complete the implementation of IPv6 NDP as I ran out of time leading to the deadline and thus left commented out code and framework methods unused, and it would've let me see if the rest of my part a implementations worked as I thought they were until that last hour.