Assignment 3 – FAQs, Marking Criteria and Other Information

What should the submission contain?

A compressed (zip/tar) file containing:
-- a program that emulates the router functionality as described in the Assignment-3 text.
-- a makefile that compiles the submitted code into an executable "router", by typing 'make' or 'gmake'.
-- a README that describes how to run the program.

Marking Criteria:

Total: / 100

[/10] Basic requirements
/4 README
/3 Make works
/3 Internal documentation

[/10] Sending INIT packet
/5 Logging packets sent
/5 Values of the packets are logged correctly

[/10] Receiving circuit DB
/5 Logging packets received
/5 Values of the packets are logged correctly

[/10] Sending HELLO packet
/5 Logging packets sent to only neighbors
/5 Values of the packets are logged correctly

[/10] Receiving HELLO packet
/5 Logging packets received from only neighbors
/5 Values of the packets are logged correctly

[/10] Sending Topology DB contents to neighbors in response to HELLO
/5 Logging packets sent when HELLO is received
/5 Values of the packets are logged correctly

[/10] Topology DB
/3 Logging topology
/3 Values of the topology are correct
/4 Logged exactly when topology changes

[/20] Shortest paths calculated
/2 Logged RIB exactly when topology changes or when RIB changes
/3 Logged INFINITY for those known routers without known paths
/15 Correctness of the path calculated

[/10] Sending LSPDUs to neighbors
/2 Logging packets are sent
/2 Values of the packets are logged correctly
/3 Only sent when this LSPDU was not received before
/3 Only sent to those HELLOED neighbours

Overview of Assignment 3 (Program Flow):

- When a router comes online, it sends an INIT packet (struct pkt_INIT) to the network (aka, Network Environment aka, nse).

- The router receives a packet back from the nse that contain the circuit database (struct circuit_DB), which essentially lists all the links/edges attached to this router.

- This information about links from circuit_DB is put into router's internal database called Link State Database (aka LSDB, also called Topology Database or the Network Map). A Link State Database gives the overall picture of the network known to that specific router.
Initially this Database looks different for each router but converges to a same database as routers exchange information.

- Each router then sends a HELLO_PDU to tell its neighbour. This means that a HELLO packet (struct pkt_HELLO) is sent on all the links that the router discovered in the previous step.

- Each router receiving a HELLO packet from its neighbour, sends a set of LS_PDU packets (struct pkt_LSPDU) to that neighbour, representing its current Topology Database state

- LS_PDU stands for Link State Protocol Data Unit, and each LS_PDU corresponds to a (router_id, link_id) pair in the network. Each unique LS_PDU (unique (link_id, router_id) pair) forms an entry inside the router's Link State Database.

- Receiving a HELLO packet from a neighbour also indicates that this neighbouring router is now 'discovered', and hence will be included in all future communications.

- Whenever a router receives an LS_PDU from one of its neighbours, it does the following steps:

    -- Add this (unique) LS_PDU information to the router's Link State Database.
    -- Inform each of the rest of neighbours by forwarding/rebroadcasting this LS_PDU to them. Note that a router sending LS_PDUs should set the 'sender' and 'via' fields of the LS_PDU packet to appropriate values.
    (To avoid loops, only send this message once per neighbouring router, and do not forward duplicates packets in the future)
    -- The router runs a Shortest Path First (SPF) algorithm on the Link State Database. This calculates the path (links to be used) from this router to each of the routers in the network. The result of this algorithm is converted to a 'next-hop database/table' called the Routing Information Base (RIB).
    -- The Link State Database and the Routing Information Base (RIB) are printed to a log file.

- This process continues until each router has a complete list of the links and their costs in their Link State Database. At this point, no packets remain transient in the network.

FAQs and Useful Information:

- Number of routers are fixed to 5, and the network connections emulated by the NSE remains same.

- The NSE emulates the 'Test Topology' handout accompanying the A3-text, and you will be marked against that topology. You can check your program against this topology for correctness.

- nse is expecting little-endian ordered byte array inside a datagram (UDP) and will reply in the same manner. All packets should follow this specification. Whenever facing packets that seem to give gibberish values/number, always check for correctness of Endianness first.

- The nse can be assumed to be lossless and all packets will be received by their intended destination.

- nse will wait until all 5 routers are connected (5 INIT messages). Once 5 routers are connected, the NSE will start with sending out CIRCUIT DB messages and receiving HELLO messages.

- the program normally runs indefinitely, waiting for more incoming packets (HELLO and LS_PDU) as some packets might arrive with a delay. The router can be programmed to 'time out' after sometime - this behaviour is up to the programmer, but marks will be deducted if the program terminates before all packets are received.

- The program does not need to be multi-threaded (sending and receiving threads in router). There is enough delay between packets forwarded by nse to allow all packets to be received and processed by the router.

- LS_PDU structure elements are described (in more detail than the A3 document) as:

```
unsigned int sender; /* sender (router id) of the LS_PDU current sender */
unsigned int router_id; /* router id of the router having the link described below */
unsigned int link_id; /* link id of the link the LS_PDU is about */
unsigned int cost; /* cost of the link mentioned above */
unsigned int via; /* link id of the link through which the LS PDU is sent from the current sender */
```

- A router copying a received LS_PDU to its neighbours will change both 'sender' and 'via', while keeping 'router_id', 'link_id' and 'cost' from the originally received LS_PDU.

How to execute/test the program?

- The program execution contains 2 parts, executing Network Simulator (nse), and executing 5 separate instances of the 'router' program.
For testing purposes, these router instances can be executed using 5 different ssh-sessions in the undergrad linux environment or they can all be executed in a single ssh-session as background tasks.

-- execute Network Simulator:

```
$> nse <host_Y> <port_id_nse>
```

-- execute 5 separate router instances:

```
$> router 1 <host_X> <port_id_nse> <port_id_router_1>
$> router 2 <host_X> <port_id_nse> <port_id_router_2>
$> router 3 <host_X> <port_id_nse> <port_id_router_3>
$> router 4 <host_X> <port_id_nse> <port_id_router_4>
$> router 5 <host_X> <port_id_nse> <port_id_router_5>
```

where:

```
<host_Y>: IP/host of the host running the router instances
<host_X>: IP/host of the host running network simulator
(NOTE: host_X can be different/same from/as host_Y)
<port_id_nse>: port to identify socket of the Network simulator
<port_id_router_$>: socket for each of the router instances.
```

- Another way for initial testing and convenience is to send 5 INIT packets from the same router code. This allows running 1 instance of router program for rapid testing/bug-fixing/prototyping.

What to output?

Output, for each router instance, a log file (named: router(id).log) that shows the following information:
-- All messages/pkts received by the router.
-- All messages/pkts sent by the router.
-- Link State Database (also called the Topology Database in the Assignment-3 document) after each change to it.
-- Routing Information Base, after each re-calculation of Shortest Path.

NOTE: these should be printed in the format specified by the Assignment-3 document. Deviation from the format is not penalized, but it is preferred to follow the standard format.

- Assignment 3 document does not specify any format for the INIT, HELLO and CIRCUIT DB messages. For debugging purposes, you can log these messages like this:
e.g.

R1 sends an INIT: router_id 1
R1 sends a HELLO: router_id 1 link_id 1

R1 receives a CIRCUIT_DB: nbr_link 2
R1 receives a HELLO: router_id 2 link_id 2

- In RIB, for a node that is unreachable, the next hop can be set to 'INF' or a 'large numeric value'.
e.g.

R5 - > R1 -> INF, INF

or

R1 -> R2 -> INF, 65535

NOTE: INF can be also be set to be some large value like 65535. Use UINT_MAX or INF, for example. Grades will not be deducted for different approaches here.