Assignment 2 - Jack Kilrain (u6940136)

Preface To Marker

Before you run the program there are some modifications to the framework that I should note. These are not changes that alter the messaging or knowledge the routers have, they are merely output convenience changes.

`-l` Flag

I have added a flag "`-l <none|normal|verbose>`" which allows you to set the logging level of the routers. This is important to note, there are three levels:

1. `none`: Routers do not log to console
2. `normal`: Routers provided logging for:
    a. Message forwarding
    b. Pathing state of found/not found
    c. Message termination (received at destination)
    d. Sending of topology update
    e. Updating neighbours with router IDs
    f. Pathing results
3. `verbose`: Logging provided for the above and also (this will spam your console with a **LOT** of messages. You have been warned):
    a. Updating channel mappings
    b. Updating DVR table connections
    c. Topology update forwarding
    d. Receiving topology update
    e. Processing topology update
    f. Topology update invalidation
    g. CIDR block assignment
    h. Sending of topology updates

`-c` Flag

I have provided an implementation of the -c flag to print the connections between routers. This will display in table format with the following connection formats:

- `0` = No connection between routers
- `1` = Connection exists between routers
- `*` = Self connection

Router Messaging Results

The second addition to the framework is a log to console of the results of the messages sent to through the network.

This will display:

- Time for all messages to be received
- Minimum hop count
- Maximum hop count
- Average hop count

Overview

Implementing a network of routers with arbitrary topology requires dynamic identification of network structure and neighbour mappings. This implementation is based around Distance Vector Routing (DVR) with local only tables. Each of the routers implements a IPv4 dynamic address for managing subnet changes locally, and also re-pathing for dynamic topology changes.

Message Types

In order to communicate network structure, self identification and actual message passing, there are three types of messages implemented. Firstly, is the Topology Update, which is designed to store the path the message took whilst travelling around the network. At each node, the pathing is updated in the DVR table and forwarded to neighbours. It is invalidated when it reaches a cycle, such that the current node has been previously visited and is in the path list.

Second type of message is the Neighbour Update. Here, the routers send this message to their neighbours to identify which channel maps to the connection between them, and the ID that should correspond. Without circulating this message, routers cannot identify which of their neighbours to send a given message to when finding the shortest path.

The last type of message is the Envelope, provided alongside the framework. No modifications are made to this message and it is purely for sending through the network until reaching its stored destination.

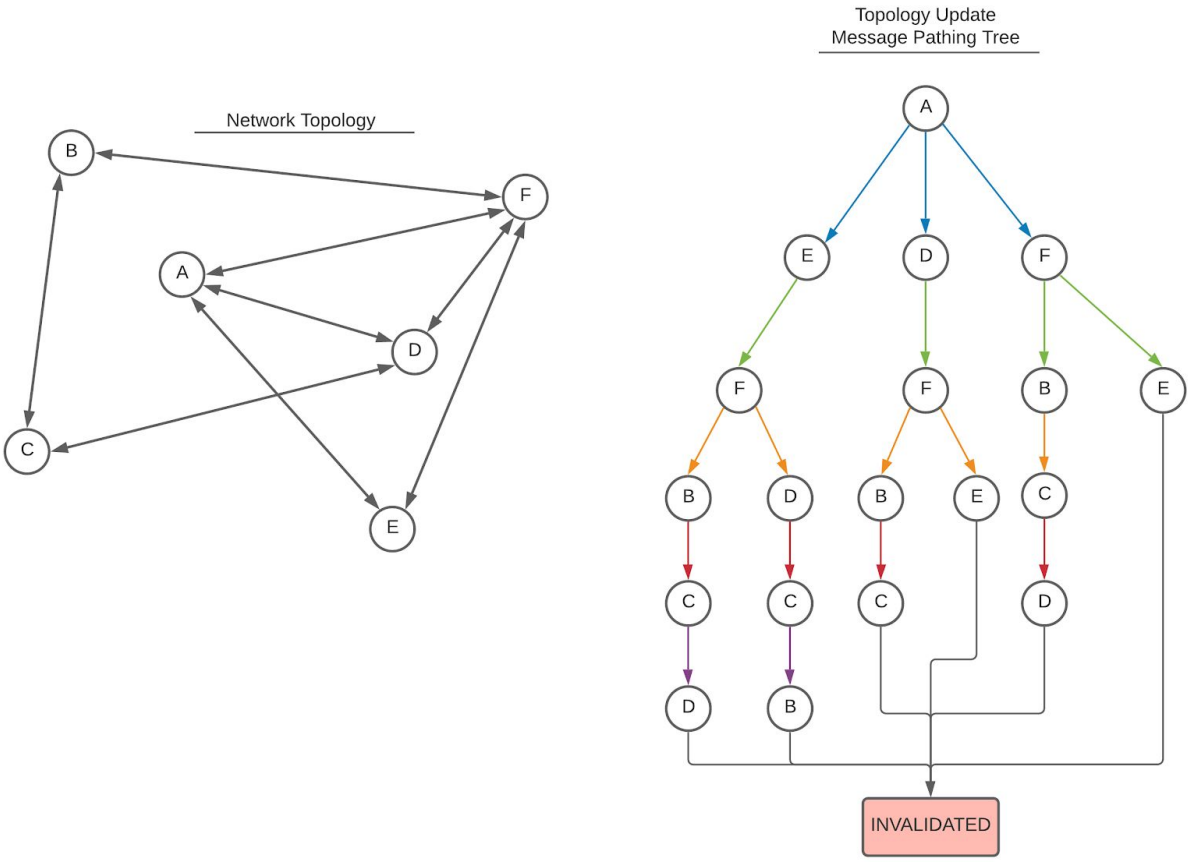
Network Mapping

In order to route messages, knowledge of the network topology is required. Routers utilise the `TopologyUpdate` message to spread through each path starting from themselves in order to map the network connections.

In order to reduce cyclic redundancy, at each branching traversal, the message is passed only to routers who haven't yet received the message. It is important to note that since every router

is passing these messages this reduces the stress to the network but also doesn't lose the guarantee of each router having knowledge of the network (assuming all routers have at least one connection.

Below is an example network with the topology update pathing tree for each message split. Note the lack of looping in the tree, guaranteeing acyclic pathing until invalidation, at which point there is no alternative.



Identifying Neighbours

In order to send messages to routers by ID, the router needs to map the channel addresses to router IDs. This is the job of the `NeighbourUpdate` message, which contains the incoming channel address and the routers ID. When another router receives this message it checks its list of channels and finds the matching one, which tells it the mapping of router ID to channel, thus completing the indexing.

In this implementation of routing, the channel IDs act kind of like MAC addresses (of different format), as they identify physical existence of network destinations. However, the usage of these channels is not like that of MAC addresses in typical routing implementations.

Pathfinding

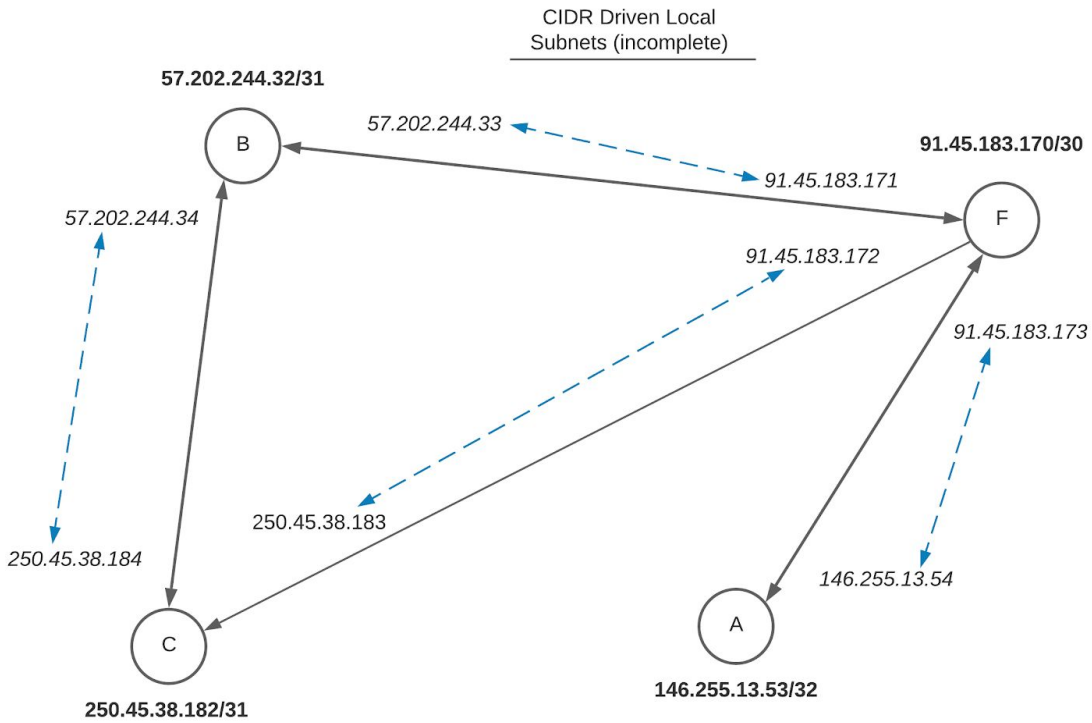
It's all well and good to know what the network looks like, but without being able to traverse it, it becomes redundant. Here Dijkstra's shortest path algorithm is used to path through the mapped network for a given destination. Note the efficiency of this algorithm drops with larger quantities of routers, however for most networks it is sufficient.

CIDR Block Addressing

Each router is assigned a random dynamic IPv4 address at startup, and a CIDR prefix based on the amount of neighbours it has. Using classless subnets allows for immediate identification of neighbouring nodes and also relative addressing changes based on topology changes.
Using the CIDR prefix, routing messages within a given subnet becomes a matter of deterministic connectivity, and also provides instantaneous invalidation of the current subnet prefix. Given any changes, a recalculation can be done in one of three ways:

- Static addressing
- Dynamic addressing
- DNS reconfiguration

This implementation uses dynamic addressing as it was the happy medium of the three, however not complete as you will note with the inconsistent subnet overlap. Implementing DNS servers would have been optimal, but time constraints permitted otherwise.



Static addressing

If the host address is required to be static, then a re-assignment can be made with a subsequent re-calculation of CIDR prefixes and updated to the supernet. However, this can be costly and should be reserved for full topological supernet changes.

Dynamic addressing

If the host address is dynamic then the subnet can be reconfigured and an update to the CIDR prefix can be made to reflect these changes. This allows the global supernet addressing to remain intact whilst having localised changes propagate as and when needed. This does not affect the ability to send and receive messages

DNS configuration

In the last situation, a DNS server sits as an intermediary between the supernet and subnet. Here we can make localised addressing changes and update the DNS records on the fly without losing actively queued messages or currently processing messages. A downside to this is that it does have a small dropout time when records are switched over, which can create isolated messages in a dead channel. However, this is extremely rare with DNS updates being microseconds at worst in this case.

Router Processing Sequence

Alongside the provided sources under `images/` is an SVG titled “*Routers sequence.svg*” (an additional png format at 330DPI is provided as well), this is a sequence diagram outlining the processing two routers perform with relative connectivity.