CS 226 Project: OSPF

Aman Kansal 170050027
Saksham Goel 170050045
Ansh Khurana 170050035
Yash Sharma 17D070059

1 Specifications of Main Module

- 8 Input ports of width (7 downto 0)
- 8 Output ports of width (7 downto 0)
- *djikstra_on*: A signal to generate the routing table

2 Overview

1. We have implemented the OSPF Routing Protocol (Version 2) with some simplifications. We have followed RFC 1583 for the implemented.
2. The Main OSPF components consists of several components that are connected together to complete the OSPF functionality. Some of the major components include HelloActParse, Neighborhood Machine, Dijkstra, LinkStateOutputMachine, LSR-Action interfaceOutput and LSU-gen

3 Functionality

1. LinkStateUpdateMachine: This machine acts on all the Link State Update Packets (Parsed first by LSU-Parser that adds the Link state advertisements in the LSU to a queue for the LSA parser). It process and checks each LSA and updates the link state database if needed.
2. NeighborHoodMachine and HelloActParse: The neighborhood state machine and it's corresponding neighborhood data structure is managed by these modules. This machine also exchanges Database Description packets and acts on them.
3. Dijkstra: This module implements Dijkstra's algorithm to compute the shortest paths upon reading the topology from the LSDB
4. FloodingFSM: Floods specified link state advertisements out on a subset of routers' interfaces.
5. LSU-Gen: Monitors neighborhood states all of the interfaces and if there is a change in state from/to the "FULL" state, it generates a new link state advertisement for the router and edits the Link state database
6. LSR-Action: Upon receiving the link state advertisements from a neighbor, this module fetches the corresponding link state advertisements from the memory and sends it on the required interface.
7. Hello-alive: Generates hello messages periodically with current neighbour information
8. InterfaceOut: Maintains output queues for each interf
9. Parser: At the beginning of the state machine we take in as input the packet byte by byte and then map it to an output port. Depending on the fields of the packet as read by the parser there are various validities telling the identity of the output, i.e. whether it is the length of the packet, or the LSA or a Neighbour etc.
10. Interface Output: In the end we also have an arbiter like interfaceOutput machine which takes in LSAs and LSA headers (for generating LSAcks) from 2 different queues, makes an appropriate packet out of it and sends on the output port.
11. MainModule: This module incorporates the all the components and maps them together for the complete functioning.