Assignment 2: Distance Vector Routing

Program design:

The problem was to implement a variation of distance vector routing algorithm. The solution code is written in Java programming language. Poisoned reverse or any the other extension has not been implemented in this solution.

The solution is contained in the file **dv_routing.java** which consists of four classes- dv_routing, MessageSender, ReceiverUpdater and DistanceVectorFormat. The class dv_routing contains the main method and controls the whole program. It parses the command line arguments, sets a UDP socket on port given by the argument, sets the initial routing table and starts the threads for exchanging messages. The program expects the config files to be in the same directory as the source file. The system was designed using multiple threads for each router. The threads perform the following jobs-

- One of threads continuously sends messages to its neighbours at five second interval
- The other thread listens for messages and performs necessary calculation to update its table using the Bellman-Ford equation upon receiving a message.

Threads communicate with each other by using shared objects that both threads can manipulate. The program uses built in Java API such as DatagramSocket, DatagramPacket and data structure implementations HashMap and ArrayList. Different instances of the program with different arguments and configs can act as different routers in a network.

It is assumed that the costs do not change during the operation of the program. Since each router sends message continuously, the program has an inherent level of tolerance for errors regarding sending and receiving packets. It is taken into account that not all routers may start at the same time which makes the determination of whether a router has stabilized difficult. In this program, it is assumed that no new router starts more than 90 seconds after the first one since problem specification requires that the program should finish within 3 minutes. The program outputs its routing table upon completion according to the format specified in the problem. A makefile has also been included with the solution for compiling and removing compiled class files.

Message Format:

The message data is produced by converting a serializable java object array into byte stream. Serialization in java is a mechanism of writing the state of an object into a byte stream. A class called DistanceVectorFormat is defined in line 181-203 in the code and instances of this class

contain the routing information i.e. the distance vector. The fields of the class include destination router id, next hop and cost. Objects of this class are directly sent and received by the routers (line 110) which uses serialization (line 106 in code) and de-serialization (line 157 in code) to convert array of objects to byte array and vice versa. A datagram packet is formed using its constructor where the DatagramSocket and IP address is passed along with the converted byte array and its length. The packet created by the java method is sent to the destination router where the data is converted to object array which contains the information required to update its own table. Using byte stream representation of java objects in messages instead of String keeps the implementation simpler and more intuitive.