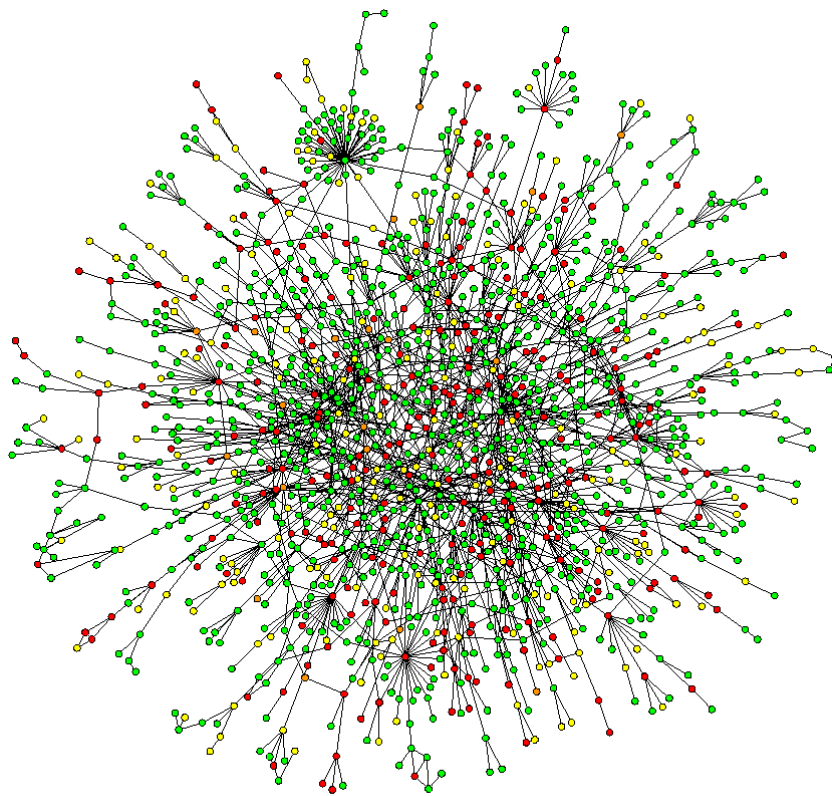


LINK STATE ROUTER IMPLEMENTATION



18/03/2018

Techniques avancées des réseaux

Voisin Adrien – Henallux/MASI – 2017-2018

INTRODUCTION

Link-State routing is one of the big family of routing protocols. Famous implementations of this technique are OSPF and IS-IS. The goal of this assignment is to implement a simplified version of a Link-state router and to discover the Socket features of a programming language.

DIRECTIVES

- You can work on this project by group of two
- The choice of the programming language is up to you. Just first check if the language support network sockets and threading
- Your routing software must run on a Linux Debian operating system

EVALUATION AND DELIVERABLES

- A small report explaining your implementation
- Your code
- After the project deadline, each student has to install its software on a Debian system running on an Alix card computer. The goal is to interconnect each routing implementation of the students and to make them communicate.

DESCRIPTION OF THE PROJECT

Router

A router is a device acting at the network layer. It serves as a relay for the transmission of packets from a source to a destination. Transmissions lines between routers will be simulated using UDP Sockets. The router must be able to discover the topology of a network composed of similar routers, thanks to the routing protocol described in the rest of this document. They must also build their routing table and be able to route data packets sent by any router of the network to any router in the same network.

Each router is identified by a unique ASCII string, representing its name. A configuration file with the following format is attached to each router (assuming that it has N neighboring routers).

[Router name]

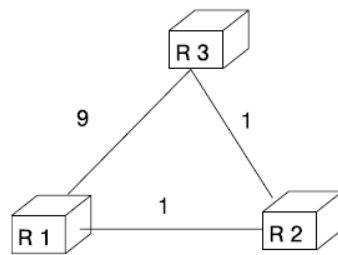
[Local UDP port number]

[Neighbor1 name] [Neighbor1 IP] [Neighbor1 UDP port] [Link cost]

:::

[NeighborN name] [NeighborN IP] [NeighborN UDP port] [Link cost]

All link costs are positive integers.



```

R1
30000
R2 130.104.78.8 31000 1
R3 130.104.78.37 30000 9

R2
31000
R1 130.104.78.8 30000 1
R3 130.104.78.37 30000 1

R3
30000
R2 130.104.78.37 31000 1
R1 130.104.78.8 30000 9
  
```

FIGURE 1

Routing protocol

The Link-State routing protocol presented in this section will allow routers to build their routing tables via the exchange of information describing the state of their links. Each router sends messages to all of its neighbors. Those messages are used to gather information about the network topology and share this information with all the other routers in the network. We now describe the types and format of messages to be used in the protocol.

- HELLO : Used to discover active neighbors (adjacency establishment). Hello messages are regularly sent to all neighbors every HELLO DELAY seconds. HELLO DELAY is a constant that must be configurable at startup. Its default value is 5.
 - The format of HELLO messages is: HELLO [Sender Name] [Receiver Name]. For example, an HELLO sent by a router R1 to a router R2 must be compliant with format "HELLO R1 R2".
- LSP : Link-State Packet, used to inform routers of the state of network links such that they can build their routing table. A router generates an LSP whenever: i) A link becomes active or inactive; ii) The last LSP emitted by the router has been generated since more than MAX LSP DELAY seconds. MAX LSP DELAY is a constant that must be configurable at startup. Its value is 60 by default.
 - The format of LSP messages is: LSP [Sender Name] [Sequence Number] [Adjacent Active Links]. In turn, links are represented as a list of pairs (neighbor, cost of the link). Inactive links are not included in the LSP. For example, "LSP 12 R1 00 R3 9 R2 1" is the LSP message, with sequence number 12, sent by router R1 to specify that it has two active links, one with R3 with cost of 9 and another with R2 with cost of 1 (R1 can have other inactive links, but they are not specified in the LSP).
 - The sequence number is a string of two characters. When an LSP is generated, its sequence number must be greater (modulo the limitation in the number of characters in it) than the one 2 of the previous LSP. Each router keeps track only of the last received LSP from each neighbor, that is, each router discards all the previous LSPs from one neighbor when it receives a new LSP from the same neighbor.
- LSACK : Acknowledgement of LSP packets, used to enforce the reliability of LSP message exchange. To prevent routers from having incoherent descriptions of the network topology, each

2 char max

received LSP must be acknowledged. If an LSP has not been acknowledged on an interface more than 5 seconds after it was sent, it must be retransmitted on that interface.

- The format of LSACK messages is: LSACK [Router Name] [Sequence Number]. For example, "LSACK R1 12" is the LSACK message used by router R1 to acknowledge the reception of LSP 12. All messages are represented as ASCII strings. Note that integers (e.g., link costs) must also be encoded as strings.

Building the routing table

In order to build a map of the network topology, each router must maintain a link state database (LSDB). This database contains the most recent link state packets (i.e. with the higher sequence number) generated by each router of the network. When a router generates a link state packet, it starts by adding it to its LSDB, then **transmits the packet to all its neighbors**. It also keeps a reference of this transmission until it is acknowledged by each neighbor. When a router receives a LSP, it first checks if that packet is already in the LSDB. If not, it is added in the database, then transmitted to all neighbors. It also keeps the reference of the transmission time on each interface for that packet. The LSDB is used to build the routing table of the routers in the three-steps process described in the following.

- Transform the database into a directed graph with routers as nodes and network links as weighted links. A link is only considered in the graph if it is bidirectional (2-way Connectivity Check), i.e., both routers connected to the link announced it in their LSP.
- Use Dijkstra Algorithm to build the shortest path tree rooted at the router1.
- For each destination in the network (i.e., any other router), store the next-hop on the shortest path to the destination in the routing table.

To simplify the project we assume that, in case of multiple shortest paths between one source and one destination, each router selects only one next-hop towards each destination.

The routing table is then a data structure containing an entry for each router in the network. Each entry specifies the destination and the next-hop to which data packets have to be forwarded.

For example, in the topology of figure 1, the routing table of R3 must contain the following two entries:

- R1 : via R2
- R2 : via R2

Data packets

The routing table described in the previous paragraph is used by router to forward data packets to their destination. A data packet has the following format : DATA [Source Name] [Destination Name] [Message] Observe that DATA is the identifier of data packets, "[Source Name]" and "[Destination Name]" are the identifiers of the original source and destination of the messages (not changed by intermediate routers traversed by the message), and "[Message]" is a string of maximum 300 characters representing the content of the data packet.

When a router receives a data packet, it first checks if the destination is itself. If yes, the packet is at destination and the content is for example displayed in the console. If not, the packet must be forwarded towards its destination. If no entry in the routing table matches the destination of the packet (like for a Destination Unreachable), then the router shows an error message on the console ("[Destination Name]: Destination Unreachable") and discards the packet.

Recommended steps

As with any implementation project, you should of course start with the design of the global architecture.

Take times to carefully think about the number of threads you need, the classes that you will implement and the relations between the different objects. Once you have a clear idea of what your application will look like once finished, you should start implementing incrementally.

In order to facilitate this implementation process, you are suggested to follow four implementations steps:

Exchanging data packet with the neighbor

First, you should focus on the communication between your router and its neighbors. Once you have parsed the configuration file and opened your communication socket, you should be able to exchange UDP datagrams with the configured neighbors. For now, focus on neighbor-to-neighbor exchanges of data packets.

When receiving data, you should first verify that the UDP datagrams are valid, i.e. they come indeed from a valid neighbor and their content respects the format given in section “data packet”, then display the content of the message in a console.

Also, if you want to check the sending capability of your router, you should provide an interface such that a human can ask the router to send a message to some other router. Typically, this interface will be a command line with a syntax like: `send [Router Name] [”message”]`. For example, if the administrator of R1 wants to send a message to R2, it will type `send R2 ”Hi R2, I’m R1”`.

Also, you should already implement control packets in case of forwarding error (e.g., Destination Unreachable).

Data packet forwarding

Now that your router is able to send and receive data packets from its neighbors, you should focus on the forwarding of data packets that are not designated to the local router. You will then need a routing table to find to which neighbor you will forward the data packet such that it will finally reach its destination. For the moment, build a static routing table that you configure manually, and check that your router is able to route messages to distant destinations.

Managing routing protocol messages

Once the router is able to route data packets, it is time to consider the routing protocol itself. Add the protocol control messages to your implementation (HELLO, LSP and LSACK), and implement the corresponding behavior. You will have to build a Link State Database and maintain different information about the received protocol control packets.

Don’t forget to carefully test each new functionality (LSP reception, acknowledgement, retransmission...), for example by displaying in a console the received/sent messages and the corresponding actions.

Dynamic management of the routing table

Now that you are able to exchange information about link states with all routers of the network, you will build the routing table from the LSDB. This step is already detailed in subsection “building the routing table”. You are allowed to reuse existing code for your graph representation and for computing the shortest path trees, of course at the condition that you provide your source.

REFERENCES

<http://cnp3book.info.ucl.ac.be/2nd/html/>