CMSC417
Apr 01, 2015
Alexander Kyei, Daniel McLaughlin, and David Sanvar

Project Design Specification

1.Introduction

The purpose of this project is to design and implement an over-lay circuit-switched traffic routing system. The routing system will use link state routing at the application level to pass messages between arbitrary nodes. The project specification is open ended, thus the purpose of the design specification is to demonstrate a clear understanding of the project and to consider potential algorithms for implementation. The design specification will cover the general design and possible security extensions.

2. General Overview and Design

Choice of Algorithms

The choice of algorithms was decided upon after carefully considering the steps needed to implement link state routing. The 5 steps of link state routing are as follows:

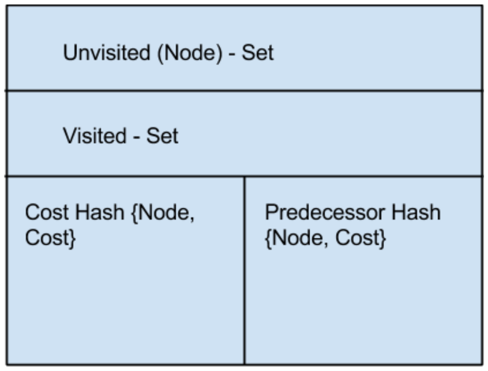
1. Discover its neighbors and learn their network addresses
2. Set the distance or cost metric to each of its neighbors.
3. Construct a packet telling all it has just learned.
4. Send this packet to receive packets from all other routers.
5. Compute the shortest path to every other router.

By using a graph data type as an abstraction for the network, Steps 1-3 can be easily met through references to each node's adjacency matrix/list. Step 4 will use the concept of flooding as described in *Computer Networks* by Tanenbaum & Wetherall. Under the assumption that the previous steps (steps 1-4) are successful, Dijkstra's algorithm will allow for each node to construct a routing table in the form of a shortest path tree to all other reachable nodes. Steps 1-5 will be executed on a node-by-node basis whenever it is apparent that the network topography has changed (either by a change in a node's configuration file, or receiving a link state packet from a node with a valid sequence number).

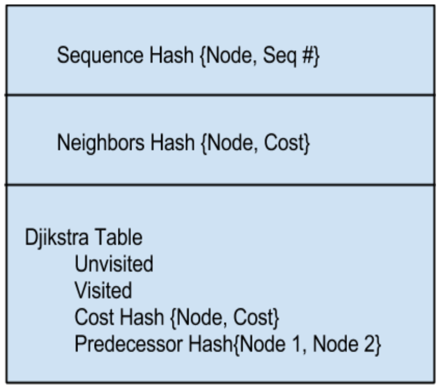
Data Structures

In order to implement the overlay routing protocol, nodes will need to keep track of several data states. The preliminary implementation uses various data structures in order to efficiently keep track of and look up data. The main data structures are nodes and packets. Each node will contain a sequence hash and routing table data structure. The hash will map nodes to sequence numbers from link state packets, in order to ensure only the latest link state packets are used in calculating new routes. Message packets will have a source node, a destination node, payload type, and payload. Through the information provided by the Node data structure, graph algorithms and techniques can be used to apply the routing protocol. By sequentially analyzing nodes, it will be possible to provide a pathway to send packets from one node to another node as long as there are intermediate nodes connecting them.

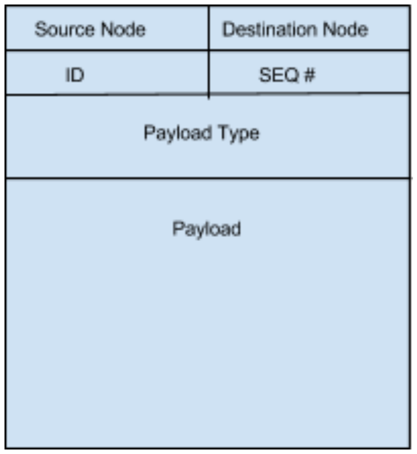
Dijkstra Table (Routing Table)
- @Unvisited(Node) - Set
- @Visited(Node) - Set
- @Cost Hash {Node, cost}
- @Predecessor Hash {Node 1, Node 2}



Node
- @Sequence – Hash(Node, Seq #)
- @Neighbors - Hash{Node, cost}
- @Routing Table (Dijkstra Table)
  - Unvisited(Node) - Set
  - Visited(Node) - Set
  - Cost Hash {Node, cost}
  - Predecessor Hash{Node 1, Node 2}


Sequence Hash {Node, Seq #}
Neighbors Hash {Node, Cost}
Djikstra Table
Unvisited
Visited
Cost Hash {Node, Cost}
Predecessor Hash{Node 1, Node 2}

Packet
- @Source - Node
- @ID
- @Sequence Number
- @Destination - Node
- @Payload Type - String
- @Payload - String



Message Passing

Control Messages are necessary to direct and carry traffic along our links. Each node will need to know how to handle a message that it receives. For example, if a message is received at a node that is not the final destination for the message, the node must parse the message and forward it to the appropriate node. Furthermore, if a destination node receives its respective packet, it must send back the proper response based on the payload type. From a packet perspective, different messages are distinguishable based on the value in the *Payload Type* field present in the packet data structure. In reality, each node will receive a string from a nearby node, and will parse that message to create a new packet. A new message will then be constructed based on the received packet's contents and forwarded to the appropriate node. The message delivery process is as follows:

1. Source Node constructs packet
2. Packet is parsed based on payload type and is either decoded into a string that ends with a special character ('EOF') (Proceed to Step 3) or accepted by the destination node (Finish).
3. String is sent in increments of **MAXIMUM PACKET SIZE** through sockets.
4. String is received by neighboring node.
5. String is parsed and used to create a packet object
6. Packet is then forwarded appropriately (Repeat Steps)

This method allows for our system to send messages, however, there is no surefire way for the source node to know that its message has been successfully acknowledged by the end destination. Each intermediate node is simply responsible for handing off the packet to another node.

The message types that our implementation will use are as follows:

Project Specific Messages

SENDMSG [DST] [MSG]*EOF*
PING [DST] [NUMPINGS] [DELAY]*EOF*
TRACEROUTE [DST]*EOF*

Output Messages – When a message (SENDMSG) reaches its final destination, it will output to standard out the message it received in this format

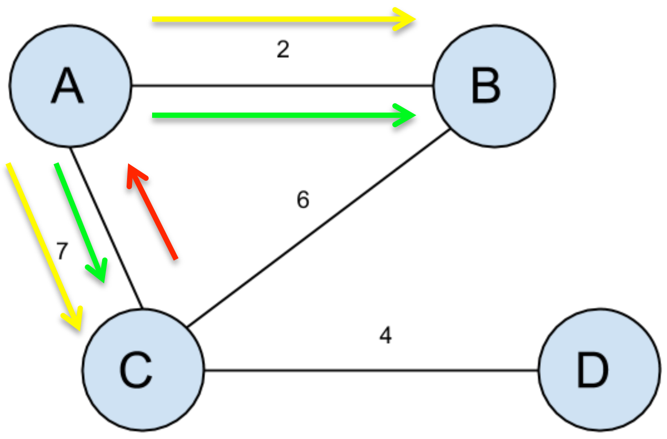
RECEIVED MSG [SrcIP] [MSG]\n

Unique Implementation Messages

REQ [SRC]\n – This message is sent to neighbor nodes to see if they are alive.
Example: REQ A*EOF*, REQ A*EOF* (Messages to B and C from A)

ACK [DST]\n – This message is sent as an acknowledgement after receiving a REQ message ("I'm alive).
Example: ACK [A] (Response from C to A)

LSP [SRC] [SEQ] [NEIGHBORS]*EOF* – This is the Link State packet.
Example: "LSP A 4 B:2 C:7*EOF*"



3. Security Extensions

Onion Routing

As a security extension we will encrypt packets on a hop by hop basis similar to the Tor network as described by Jill Scharr, "Each relay [node] decrypts only enough of the data packet wrapper to know which relay the data came from, and which relay to send it to next." (http://www.tomsguide.com/us/what-is-tor-faq,news-17754.html).

The implementation we would like to propose makes several assumptions. The assumptions are as follows:

1. Every node in the network is able to decrypt part of a message using the same decryption methods.
2. Source and Destination nodes are able to agree on an end-to-end encryption format
3. Destination nodes can create a public/private key pair and send the public key to source.
4. There is a method in place that keeps nodes from discovering too much of the path.
5. The network topography is known, and a path from the source node to destination node can be determined.

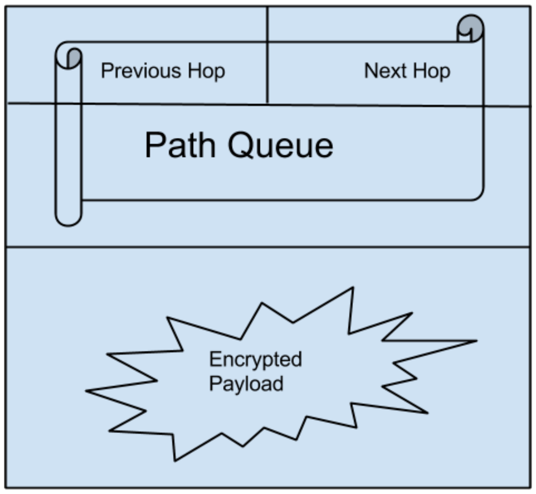
To achieve this implementation, new message types and data structures will be needed. The new message and data structure are as follows:

```
ENC [SRC] [NEXT HOP] [ENC PAYLOAD] [ENC PATH]\n
KEY [DEST] [PUBLIC KEY]\n
```

The data structure used to create the ENC message is depicted below. The previous hop, next hop and path queue members (inside the scroll) are all encrypted using a method that all nodes are able to decrypt (Assumption 1). The *Encrypted Payload* is encrypted using the public key provided by the destination node.

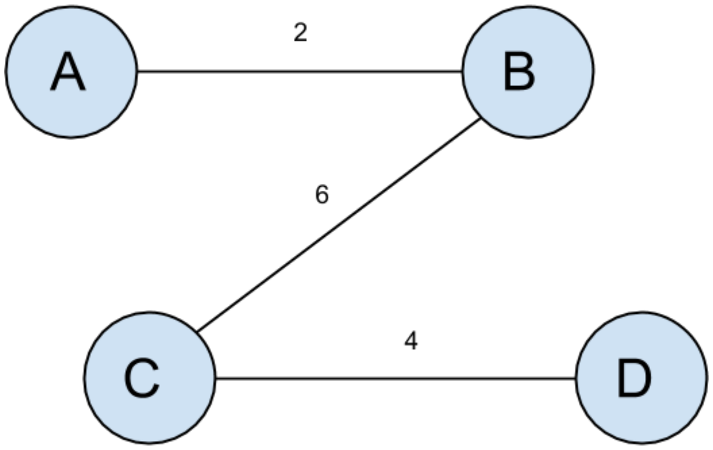
Encrypted Packet

@Previous Hop – Node
@Next Hop – Node
@PathQueue – Stack
@Encrypted Payload



The algorithm for sending an encrypted packet from node A to D is as follows:

1. Using the normal means of sending messages, decide on an end-to-end encryption method, send public key.
2. Using its routing table, Node A constructs a path queue to Node D and encrypts it.
3. Each node is capable of dequeuing only one node in the path queue and decrypting it (Assumption 3). After a node decrypts a path node, it forwards the packet to the next node.
4. When the queue is empty the message is decoded with the private key By D



A potential drawback to this implementation is that if links go down while the encrypted packet is transmission, then the packet will be lost. For example, if A would like to send an encrypted packet to D--if A successfully transmits to B, but the link between B and C is broken, the packet will never reach D (until the link is reestablished). The source node will need to know when to resend the packet or timeout if this is the case.

4. Production Design Approval

The students that have signed below have reviewed the Project 3 design specification and agree with the approach it presents. Any changes to the design specification will be coordinated with and approved by the students.

Name: Alexander Kyei
Name: Daniel McLaughlin
Name: David Sanvar