StarNet Final Project

11/26/18

by Andre Hijaouy and Tony Sun
ahijaouy@gatech.edu
tony.sun@gatech.edu

Submission Details

star_node.py -- the main part of the project. This file contains the logic that runs a star node and allows it to do peer discovery, churn detection, round trip time calculations, and message broadcasts.

contact_node.py -- a class that represents a StarNode's contact and RTT info

contact_directory.py -- a class that stores all the peer nodes the main star node knows about

messages.py -- all the classes for the different types of messages used in the project

message_factory.py -- a factory that is used to create messages inside the star node

reliable_socket.py -- a class the contains all the necessary logic to send and receive UDP messages

socket_manager.py -- a class that acts as the interface between a the star node and the reliable socket. The SocketManager handles queuing up messages to be sent and putting received messages in queues that can be consumed by different threads in the StarNode.

logger.py -- a util class that handles creating the log for the starnode

Design Report

Contents

1. Star-Net Protocol Description

Peer Discovery

- Active nodes will try to discover all other active nodes
- A node is identified by a 1) Name, 2) IP Address, 3) Port Number.
- If a node is started without a point of contact, it will remain idle listening on its port until another node has connected to it.
    - If the node does not receive any packets within 2 minutes of startup, it will terminate with an error.
- If a node is started with a point of contact (POC), it will attempt to contact the POC node by polling it every 5 seconds.
    - If the node cannot contact its POC within 2 minutes of startup and has not received any other packets, the node will terminate with an error.
- Whenever a node A is attempting to establish a first connection with a node B, it will send a packet with node A's information as well as any information it may know about other nodes. Node B will receive this information from Node A, add it to Node's B inner data stores, and then send node A any information Node B may have about other nodes.
    - If either node A or node B learn about any new nodes from this process, they will repeat the process with the newly discovered nodes.
    - If node A does not receive a response back from node B, it will retry in 2 second intervals.
- Failure condition:
    - If a node is not listed as any other nodes' POC and does not have a POC itself it will never be able to connect to the network.

Churn & Keep Alive Mechanism

- "Churn": Possible for node to go offline for a random period of time and come back online.
    - When a Node comes back online, it will have no previous memory
- Max number of star nodes (N) will be provided as an initialization parameter
- Node A will send periodic heartbeat-packets to all other nodes it is aware of to communicate Node A is still online and check the status of other nodes.
    - Packets sent every 3 seconds
    - The request packet will include a 0 direction bit, and Node B will increment the direction bit to 1.
- If a Node B is unresponsive for more than 8 seconds, mark it as offline
    - This will mean a recalculation of Star Node is necessary

Round-Trip Time (RTT) Measurements

- Every node needs to periodically measure its RTT to every other active star node
    - RTT = Delay from X -> Y + delay from Y -> X
    - Done whenever the network changes or every 3 minutes
    - Stored as a list inside of Node A
    - If request is lost, repeat RTT measurement
- Each Node should update its RTT and send that sum to all other star nodes.

Optimal Network Formation

- The node with the shortest RTT should be considered the star node.
- When Node A is broadcasting, it should send the message to the star node who will then broadcast it out to every other node in the network.

Reliable Message Broadcasting

- In order to ensure messages are broadcast reliably on the network, nodes will send an acknowledgement (ACK) packet after receiving any packet.
- If a sending node does not receive an ACK from the recipient within the ACK_TIMEOUT window, the sender will resend the packet.
- The ACK_TIMEOUT window is 1.2 seconds.
- The sending node will continue to retry up to 5 times.

2. Packet Structures

All packets will share the same header but will contain different payloads depending on the packet type.

General Packet Structure

- packet_string[0]: Packet Type
- packet_string[1:17]: Origin Node Name
- packet_string[17:21]: Packet UUID
- packet_string[21:]: Payload

Discovery Packets

- packet_string[0]: Packet Type = “D”
- packet_string[1:17]: Origin Node Name
- packet_string[17:21]: Packet UUID
- packet_string[21]: Direction (0 if initial request, 1 is response)
- packet_string[22:]: JSON encoded string containing all known nodes from sender (only if direction is 1)

Heartbeat Packets

- packet_string[0]: Packet Type = “H”
- packet_string[1:17]: Origin Node Name
- packet_string[17:21]: Packet UUID
- packet_string[21]: Direction (0 if initial request, 1 is response)

RTT-sum packets

- packet_string[0]: Packet Type = “R”
- packet_string[1:17]: Origin Node Name
- packet_string[17:21]: Packet UUID
- packet_string[21]: Stage (0 if initial request, 1 is response, 2 if broadcasting RTT Sum)
- packet_string[22:]: Send Time (Only if initial request or response (stage=0 or stage=1)
- packet_string[22:]: Network Size (Only if broadcasting RTT Sum (stage=2)
- packet_string[23:]: RTT Sum (Only if broadcasting RTT Sum (stage=2)

Application messages

- packet_string[0]: Packet Type = “A”
- packet_string[1:17]: Origin Node Name
- packet_string[17:21]: Packet UUID
- packet_string[21]: Forward (0 if recipient accept and display message, 1 if should forward to all nodes)
- packet_string[22]: Is File flag
- packet_string[22:38]: Sender name
- FILE type message
    - packet_string[38:40]: File Name Length (x = 40 + file name length)
    - packet_string[40:x]: File Name
    - packet_string[x:]: File to broadcast
- packet_string[38:]: Message to broadcast

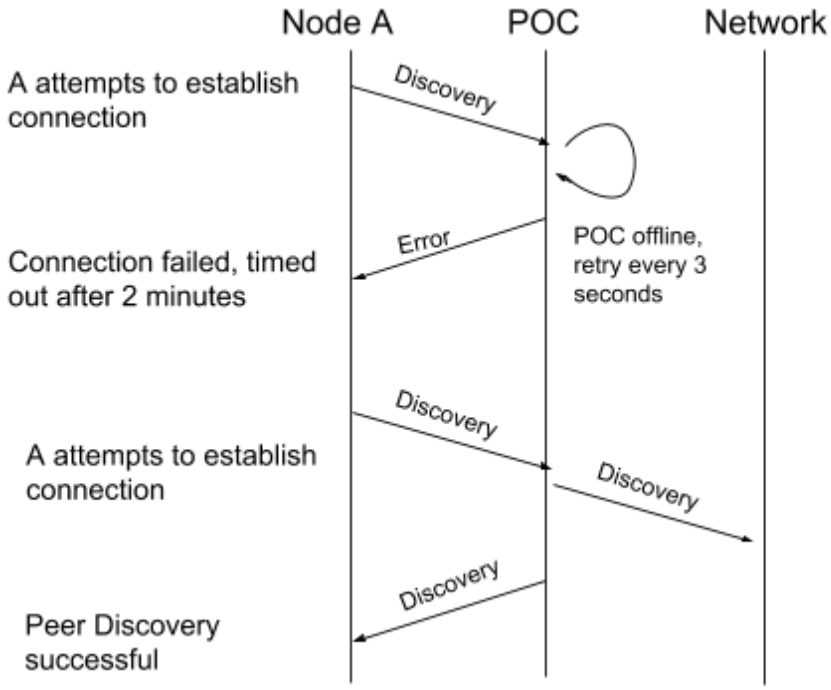
Ack packages

- packet_string[0]: Packet Type = “K”

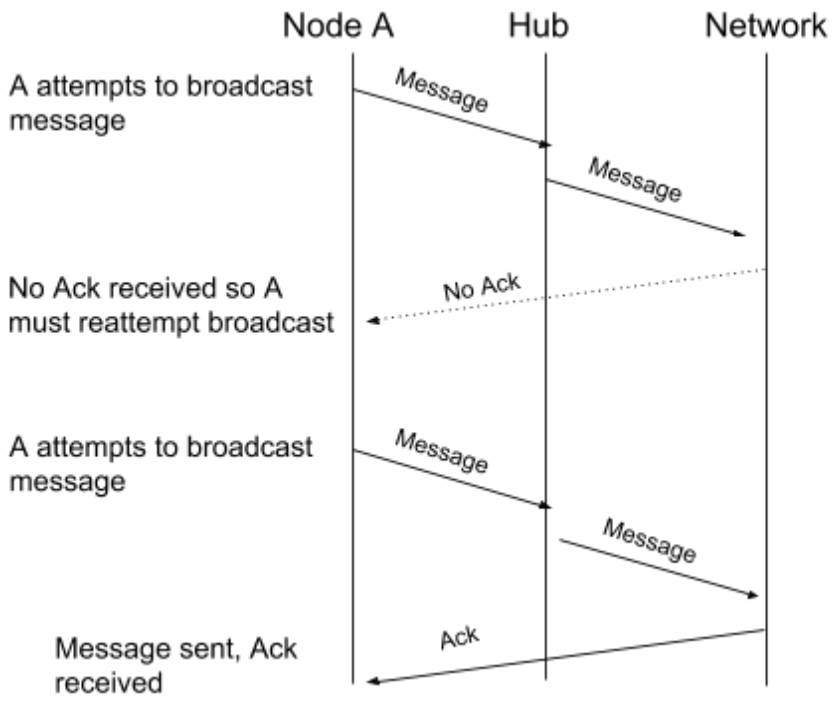
- packet_string[1:17]: Origin Node Name
- packet_string[17:21]: Packet UUID
- packet_string[21:]: Ack ID=(Origin Node Name + Packet UUID) of the packet being ACK'd

3. Timing Diagrams

- Peer Discovery



- Message Broadcasting



4. Algorithm Descriptions

- Node Churn
  - A node will be considered offline if it has been unresponsive for more than 15 seconds.
- Central Node Selection
  - Initially the node with the shortest RTT will be marked the central node.
  - RTT will be recalculated every 3 minutes regardless of network activity.
  - Whenever a new node joins the network or a node leaves the network, RTT will be recalculated
  - A new node will only replace the current Central Node if new_node_rtt < central_node_rtt

5. Data Structures

- Queues will be used to store messages.
  - The socket will put its messages into a queue onto the hub which will be broadcast in order.
  - Queue is thread-safe.
- Hash sets will be used to store the nodes that a node has discovered.
  - Each node will keep track of all the other nodes and must be able to efficiently add more nodes into the set without duplicates.
- Arrays will be used for storing round trip times.
  - Each node must calculate its RTT to other nodes and then determine the best RTT for the star node.

6. Thread Architecture

- There are several processes happening concurrently during the Star-Net protocol each on different timers. These processes must run on seperate threads.
- The nature of the StarNode protocol necessitates many threads to execute concurrent actions.
    - Socket Actions
        - 1 thread will be dedicated to listening for incoming packets
        - 1 thread will be dedicated to polling an outbox queue and sending messages on the socket when they are put in the outbox.
        - 1 thread will be dedicated to handling acknowledgements as they are received
        - 1 thread will be dedicated to re-sending packets if acknowledgements are not received within 5 seconds.
    - Peer Discovery
        - 1 Thread will be dedicated to listening for incoming peer discovery messages and handling them
    - RTT
        - 1 thread will be dedicated to calculating RTT every 10 seconds and everytime there is a change to the network. This thread will also broadcast RTT sums to other nodes
    - Heartbeat
        - 1 thread will be dedicated to sending out Heartbeat packets to other nodes every 3 seconds.
        - 1 thread will be dedicated to listening for heartbeat packets and handling them appropriately.