Programming Assignment 2

Nicolas McGarry

Marcus Loo

CS 3251

4/17/18

Peer Discovery

Our algorithms for constructing the ring network are separated into 3 distinct parts. The first step in this process is to discover all of the reachable nodes in the network. To discover all of these nodes, we use a simple process, where for each node:

1. On initialization:
    a. Create a list of nodes, and append both itself and its Point of Contact (PoC) to the list, from here dubbed the “Known” list.
    b. Send a data packet containing its Known list to its Point of Contact.
2. Until either no new data has been added to the Known list after multiple updates, OR when the size of the node’s Known list is equal to the total number of nodes in the network (N):
    a. If it receives a data packet from any node:
        i. Append any new data to the Known list.
        ii. Create a separate list of nodes that contains all of the nodes added to the Known list, called the New list.
    b. If the New list is not empty:
        i. For each node u in the New list:
            1. Send the Known list to u.

This algorithm aims to create a cohesive list of every reachable node in the network, and distribute it to all nodes. By constructing this graph, we are able to then move on to the second step in the process, which is to construct the Round-Trip Time Table (RTT). Before considering interferences such as churn, there are three possibilities to consider:

1. Cyclic networks
    a. In cyclic networks, this algorithm will construct the graph quickly, and there will be a period in which all nodes contain the entire graph; this case is the largest reason for the second terminating condition of the loop.
2. Acyclic networks
    a. Acyclic networks work much like cyclic networks do.
3. Disconnected networks
    a. In a disconnected network, the graph of all reachable nodes is constructed, but not all nodes are reachable.
        i. This creates a problem in RTT construction, but for the purposes of discovery, this is acceptable.
    b. This is the main case in which the first terminating condition of the loop is necessary: if there are no updates to the Known list, then it can be inferred that there are no new nodes to add to the list, even if the size of the Known list is not equal to N.

Churn and Keep-Alive Mechanism

After peer discovery has happened, we are guaranteed that all the Ringos will be online for at least a minute. Each Ringo has a timer set to 1 minute, and a list is created with each element being true representing the state of the Ringo. After a minute passes, each Ringo sends a message to every Ringo in its Known list. If there is a response, The Ringo marks the Ringo that responded as true if it is not already, and then the timer is reset to 1 minute and the process repeats. If there is no response, the Ringo tries 3 more times. Upon the 3rd failure, the Ringo will mark the unresponsive Ringo as false in its churn list. Since offline Ringos are guaranteed to be offline for at least 15 seconds, the timer of the Ringo that did not get a response will be set to 15 seconds. This entire process is handled on a thread that is alive for the duration of the program.

Round-Trip Time Table Construction

After the entire network has been discovered, and each node in the network has access to the host name and port of every other node, they can then begin to measure round-trip times. The algorithm is straightforward:

1. On initialization:
    a. Create a dictionary with the key being the current Ringo and the value being the round-trip times for that Ringo, and populate each entry with positive infinity. This is the RTT Vector.
2. For each member m in the Known list:
    a. Send a data packet to m, and begin recording time.
    b. Upon receipt of the ACK packet, populate the associated entry in the RTT dictionary list with the time elapsed since the packet was sent.
3. After this is completed, send the RTT-vector packet to each member in the Known list.
4. Create a list of dictionaries of the packet created by the current Ringo and all the packets sent by the other Ringos. This list of dictionaries will be the RTT table.

If the network is disconnected, there is a problem: it is impossible to create a NxN RTT matrix where all non-diagonal entries are not positive infinity. This case does not require a special workaround, however; because the round trip times are initialized at positive infinity, any time a node would look to forward a packet, it will not choose any of the unreachable nodes, as they are both invalid and larger than any other path in the matrix.

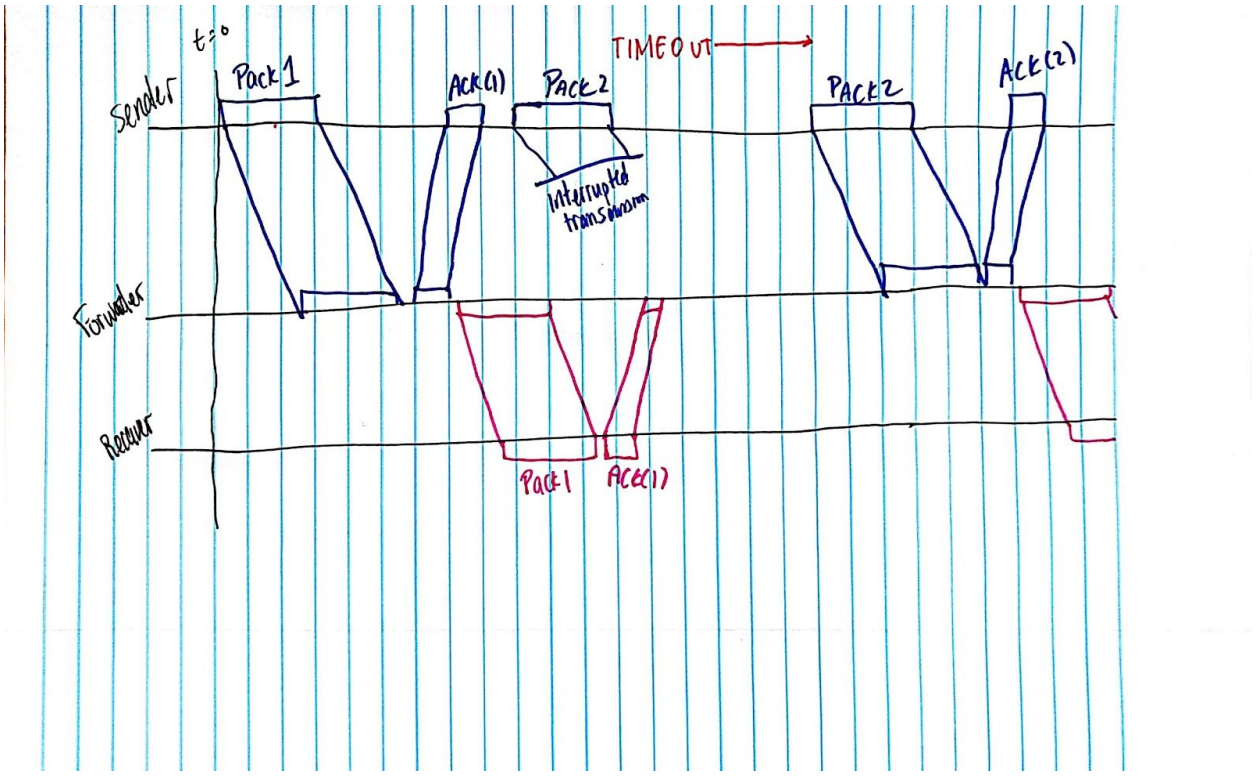
Optimal Ring Formation

Once the RTT measurements are found and each of the N Ringos have formed an NxN RTT matrix, we will be using an implementation of the traveling salesman problem in order to find the optimal network. We are currently thinking of using the Held-Karp algorithm, which is a dynamic programming algorithm that runs in O(n^2 * 2^n). Every node will perform this computation.

Upon completion, each node will send a message to every node in the Known list that that task has been done, and their neighbors have been determined.

Reliable Data Transport

For this project, we will be implementing a basic form of Stop-and-Wait protocol. Each data packet sent will have a flexible 2 to 14 bit sequence number. Every time a packet is sent, the Ringo will wait for an ACK signal to be returned before it times out, or it will resend the packet. This process will continue until the data transfer is complete, and the recipient Ringo has confirmed receipt of all packets. Further details about the ACK packets and data packets can be found in the section titled “Header Data”. Below is a timing diagram describing our planned Stop-and-Wait protocol.



Routing

In case the Sending router never receives an ACK signal from the Receiver, the Sender will select the other path. After the data transfer has completed, the network will re-construct itself as described in the earlier sections.

Header Data

There are several types of packets in this project, each with different purposes.

The ACK packet is as follows:

‘A’ [8]	**‘C’ [8]**	**‘K’ [8]**	**Packet Seq Number [8]**

When the Packet Seq Number is 0xFF, the ACK is not responding to a data packet.

The RTT packet is as follows:

‘RTT’ 0000 0000 [32]
Payload: sizeof(dictionary)

This packet is used to pass the RTT vectors around the network, to create the RTT matrix.

The Keep-Alive packet is as follows:

‘KEEP’

The Data packet is as follows: in total, the max Data packet size is 1026B.

‘DATA’ [4B] + Packet Sequence Number [2-14B]
Source [4B]
Destination [4B]
Payload: (0, 1000 B]

The Peer Discovery packet is as follows:

‘PEER’ [32]
Payload: sizeof(List)

Threading

In this project, we used threads to take care of simultaneous ACK-receive and packet-sending mechanisms. For peer discovery, we had two functions running simultaneously: one for sending the PEER packet, and the other for receiving said packet, and returning an ACK. Similarly, this is also how the data transfer and RTT calculation functions work. We used mutex locks from Python's threading library to ensure no data collided with itself during read/write. The Ringo commands were also run on a thread that last the entirety of the program past ring formation. The aforementioned Keep-Alive thread is also run on its own thread.