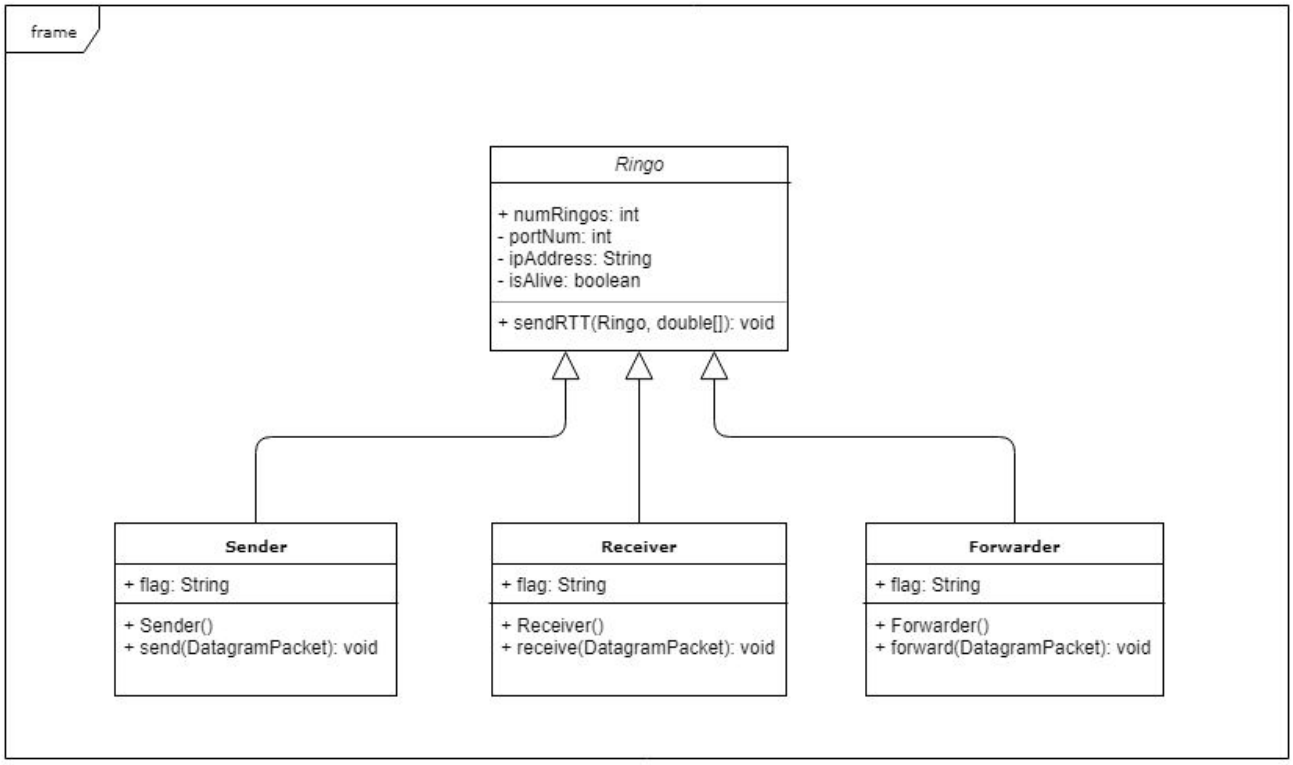
Rudy Lowenstein
Kenneth Scharm
01 March 2018
CS 3251
Constantine Dovrolis

Ringo Project Design Report

Overall Architecture



We plan on implementing the Ringo Protocol by abstracting the idea of a Ringo. We decided on abstract class because a Ringo by itself cannot exist and does not exist in the system. Every Ringo in the system will have the number of total Ringos, its own port number and IP address, as well as a boolean that represents the alive state of that Ringo. All Ringos have a method to send its own RTT vector to other Ringos.

We classify the categories of Ringo, or sub-classes, as the following: Senders, Receivers, and Forwarders. The type of Ringo will be implemented with a String (or perhaps an enum) that designates that particular Ringo's role. Sender Ringos can send data to other Ringos (via the send method). Forwarder Ringos can forward data from one ringo to the next via the forward method. Receiver Ringos can receive data from a Ringo via its receive method.

Packet Header Structures

- Keep-Alive Packet

String Destination IP Address	Int Destination UDP Port Number	String Source IP Address	Int Source UDP Port Number

- Round-Trip-Time Vector

String Destination IP Address	Int Destination UDP Port Number

- Data Packets

Int Sequence number	Int Destination UDP Port Number	String Destination IP Address	String Source IP Address	Int Source UDP Port Number

- ACK Packets

String Destination IP Address	Int Destination UDP Port Number

Algorithms

Optimal Ring Calculation

To compute an optimal ring, we will be using a brute force approach. We plan on achieving this by testing all permutations of Ringos, and picking the sequence with the minimum sum of RTT time. With brute force, the solution can be computed in O(n!) time. This does not work well when n is large, but n is very small for this assignment, so the brute force approach is actually more efficient than the Held-Karp dynamic programming solution, which has $O(n^2 2^n)$ running time.

Key Data Structures

- Each Ringo object has a local RTT array of size N - 1, where each entry is a double representing the RTT to a particular Ringo
- Global RTT map from Ringo ID to and array of size N - 1, where each entry is a double representing the RTT to a particular Ringo
- Global optimal ring structure backed by a circular doubly linked list, where each node points to two other nodes, namely a previous and a next

Thread Architecture

Each Ringo will be on its own thread to send and receive keep-alive packets. We will be extending Java's Thread class to implement our own threads. Additionally, we will have a main thread for sending/receiving data packets (including ACK packets). On this thread, we will implement the reliable data transport protocol over UDP. If a Forwarder goes down during transfer, we will use the alternate path to complete reliable transfer. After transfer is complete, we will use the main thread to recalculate the optimal ring. No separate thread is needed because data transfer will halt until the ring is rebuilt.