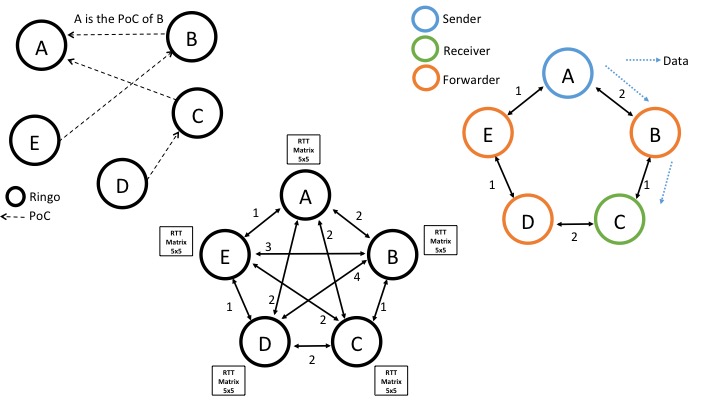
# CS 3251 - Spring 2018

# 2nd programming assignment (a.k.a., “the project”)

# *Reliable data transfers over a self-organized optimal ring network*

**You can work on this assignment in groups of two students or individually. But you need to finalize this decision by the due date of the first milestone (Feb 27)**.

You can use C/C++/Python/Java. Please note that we will test your code on the CoC NetLab machines that run a special network emulator -- it is not sufficient if your code only works on your laptop.



## 1. Functional description and requirements

You are asked to design and implement a simplified peer-to-peer communication protocol that will be able to dynamically form an optimal ring network and to perform reliable data transfers over that network. Let us refer to this protocol as the ***“Ringo”*** (honoring *Ringo Starr* from the Beatles of course!)

Peer Discovery

Think of each peer (referred to as “Ringo”) as a process that will be running at one of our NetLab machines. When you run a Ringo X, it will initially know how to reach at most one other running Ringo Y – we refer to Y as the “Point-of-Contact” (PoC)” of X. Y will be provided to X as a command-line argument. One first task is to perform “Peer Discovery” – this term means that each active Ringo should try to discover all other active Ringos in the network. This can be performed, in principle, if the peers exchange with each other the identity of all other active Ringos they know (the “identity” of a Ringo includes the IP address and UDP port number of that process). You will need to design the details of this peer discovery mechanism.

Note that it may be impossible to discover all active Ringos, depending on the given initial PoC of each Ringo. You need to figure out the mathematical condition that the initial (peer, PoC) pairs should satisfy so that every Ringo can discover all other Ringos. You can assume that when we test your code, we will make sure that this condition is satisfied.

Churn and Keep-Alive mechanism

For simplicity, the total number of Ringos (denoted by N) will be given as a command-line argument to every Ringo. After each Ringo discovers the N-1 other Ringos, the process of peer discovery is completed. After that point, each Ringo will need to periodically check if all other Ringos are still active. It is possible that a Ringo may go offline for a randomly long period of time, and then come back online. This is referred to as “Churn” and it will be one of the most interesting aspects of this project.

You will need to design a “Keep Alive” mechanism, based on the exchange of periodic short messages between the active Ringos, so that they can automatically and quickly detect when one of them is offline. *For simplicity, you can assume that at most one Ringo can be offline at any point in time.* Further, you can assume that when a Ringo goes offline it stays in that state for at least 15 seconds, and when it is online it stays in that state for at least 1 minute.

Round-Trip Time (RTT) measurements

Each Ringo X will need to measure its RTT with every other Ringo. This (N-1)-dimensional vector is referred to as the RTT vector of Ringo X. The N Ringos will need to exchange their RTT vectors so that each of them can form an N-by-N RTT matrix that contains the RTT between every pair of Ringos in the network. Note that all Ringos will eventually have the same RTT matrix.

For simplicity, *you can assume that the RTTs do not have any significant variations*. This implies that you need to form the RTT matrix once.

Optimal ring formation

After the N Ringos have the complete RTT matrix, they need to compute the optimal ring network that interconnects all of them. A ring network is optimal when its links give the minimum cumulative RTT. You can assume that we will set the RTTs between NetLab machines so that there is only one optimal network (no ties). Note that the value of N will be quite small (say 4 or 5) – this implies that you should be able to calculate the optimal solution despite the computational complexity of this NP-Hard problem (also known as “Traveling Salesman Problem”).

Reliable Data Transport

One Ringo will be set (with a command-line argument) as the Sender. Another Ringo will be set as the Receiver. The N-2 other Ringos will act as Forwarders. The Sender should be able to originate a file transfer to the Receiver. The Forwarders are just moving each received packet from one Ringo to the next along the path (a completely “stateless” operation).

The file transfer needs to be reliable, meaning that the transferred file at the Receiver should be identical with the file at the Sender.

We will be using a Network Emulator that will be introducing random packet losses between the NetLab machines. The Emulator can also duplicate packets.

For simplicity, we suggest you design a simple reliable data transport protocol. For instance, you can use a variation of the Stop-and-Wait protocol (that we will also discuss in class). In that protocol, the Sender transmits one data packet at a time, and waits for that packet to be ACK-nowledged. If an ACK is not received within a certain Retransmission Timeout (that you will need to pick wisely), the packet is retransmitted by the Sender. Packets and ACKs need to have a sequence number (how many bits do you need for the sequence numbers?). Feel free to design or experiment with other reliable data transport protocols.

Some requirements about the transport protocol:

* You should not use TCP for the data transfer or for any other task in this project. You should only use UDP sockets.
* Your transport should be efficient. For example, a design that tries to deal with packet losses by transmitting each packet ten times would be a very bad design.
* The communication protocol between the Sender and Receiver should be connection-oriented (i.e., the Sender needs to “call” the Receiver and establish a connection before you start transferring data from the Sender to the Receiver).
* The Sender needs to terminate the connection when the transfer is complete.
* The transferred file may be binary.

Some simplifying assumptions about the transport protocol:

* The maximum size of the transferred file will be 1MB.
* The Sender and the Receiver will NOT go offline during an active transfer.
* At most one Forwarder can go offline during an active transfer.
* The Sender will NOT initiate a new transfer before the previous one is complete.

Routing -- and transfers in the presence of churn

A ring network has only two paths between every pair of nodes. The Sender Ringo will need to select the path that minimizes the RTT to the Receiver Ringo. Only that path should be used for the transfer.

However, one of the Forwarders may go offline during a data transfer. If that Forwarder is in the path that the Sender uses, the data transfer will not be able to proceed. It is not acceptable to just wait for the offline Forwarder to come back online (that may never happen). Also, it may be too complicated if you try to continue the data transfer while the remaining Ringos try to re-converge to a new ring. Instead, we request that you design your protocol so that the data transfer completes by using the second available path in the ring from the Sender to the Receiver, while putting the ring re-computation process on-hold. That re-computation process can take place after the data transfer is complete.

### 2. Ringo interface

The Ringo command-line should be as follows:

* Command line: **ringo <flag> <local-port> <PoC-name> <PoC-port> <N>**

**<flag>:** S if this Ringo peer is the Sender, R if it is the Receiver, and F if it is a Forwarder

**<local-port>:** the UDP port number that this Ringo should use (at least for peer discovery)

**<PoC-name>:** the host-name of the PoC for this Ringo. Set to 0 if this Ringo does not have a PoC.

**<PoC-port>:** the UDP port number of the PoC for this Ringo. Set to 0 if this Ringo does not have a PoC.

**<N>:** the total number of Ringos (when they are all active).

Example: ringo F 23222 networklab3.cc.gatech.edu 13445 5

**Ringo Commands**

After a Ringo process starts running, it should interact with the user through a basic text-based interface. The interface should be able to recognize the following commands:

* Ringo command: offline <T>

That Ringo peer should stop sending or processing any packets for a period of T seconds. After the end of that offline period, that Ringo should go back online but without remembering any prior information about other Ringos, the optimal ring or the RTT matrix. Think of a machine that goes back online after a crash/reboot.

* Ringo command: send <filename>

This command will be executed only at the Sender. It triggers the creation of a new connection with the Receiver (does not need to be specified because there is only one Receiver at the network). The name of the file to be transferred is *filename* and it should exist at the local directory.

Example: send foo.jpg

* Ringo command: show-matrix

This command should print (in an easy-to-understand format) the RTT matrix between all active Ringos.

* Ringo command: show-ring

This command should print the sequence of Ringos in the optimal ring.

* Ringo command: disconnect

This command (gracefully) terminates that Ringo process.

## 3. Milestone-1: Design Report (**due on February 27**)

The design report will need to describe at least the following:

* A description of the overall architecture of your Ringo protocol.
* A description of the header structure for each type of Ringo packet (e.g., the Keep-Alive, RTT-vector, Data packets, ACK packets, etc).
* Any relevant timing diagrams that would help illustrate the behavior of your protocol.
* Algorithms/pseudocode for any non-trivial Ringo functions (e.g., how you compute the optimal ring network).
* Description of the key data structures you plan to use (e.g., a matrix for the RTTs, a vector will all information about other Ringos, etc)
* Thread architecture: we highly recommend that you use thread programming for this project. If you do so, make sure that you design report identifies the threads that you will be using for each Ringo process. For example, you may need to have a separate thread for exchanging and processing Keep-Alive packets.

Please note: we expect that your design will evolve over the course of this project – this is fine. However, it is necessary that you submit a complete design report by the Feb 27 due date. Your design report will be graded as “Successful” (100% credit) if it is submitted on time and it includes at least the previously mentioned sections. We will not grade it based on the details of your design.

At the end of the project, after you submit the final version of the code, we will ask you to resubmit your Design Report. The expectation is that that Report will describe the final version of your design.

## 4. The two programming milestones (**due March 15 and April 12**)

By milestone-2 (due March 15), you will need to submit a working version of your code. We will only test that version of your code however WITHOUT the following two sources of complexity:

1. No churn. Ringos never go offline.
2. The network emulator does not cause packet losses.

By milestone-3 (due April 12), you will submit the final version of your code, together with the final Design Report. We will test again your code at that point but including the previous sources of complexity.

5.Testing on an Unreliable Network

We have set up several physical machines to test your code. These machines are configured to delay packets, duplicate packets and to reduce the capacity of the network. This will allow you to test your implementation under adverse conditions. To access these machines, you need to be either on the Georgia Tech network (i.e., using GT machines or connected through the GT WiFi network) or using a Georgia Tech VPN client. Steps to install the VPN client are given by OIT (see <http://anyc.vpn.gatech.edu>). Make sure to start and login to the VPN client every time you plan to use the remote machines.

In order to access these special machines, you need to *ssh* as follows:

ssh <gt\_username>@networklabX.cc.gatech.edu

where X is an integer between 1 and 8.

***Remember to use 130.207.107.\*/127.0.0.1 as destination or source address (to listen on) in your code.*** For transferring your code to the remote machines, you may use *scp, sftp,* or the more user-friendly [*filezilla*](https://filezilla-project.org/), which has a GUI.

We have used *netem* along with *tc* in order to setup adverse network conditions. If you feel more comfortable testing/debugging on your laptop, we show how to setup the network emulator at your laptop in Section-7 of this document.

## 6. Submission instructions

**For the first milestone,** you will only need to submit a PDF file – the Design Report. You do not need to submit any code at that point.

**For the second and third milestones,** please follow these instructions:

Please turn in well-documented source code, a README file, and a sample output file called sample.txt. The README file must contain :

* Your name (or names for group projects), email addresses, date and assignment title
* Names and descriptions of all files submitted
* Detailed instructions for compiling and running your programs
* Design Documentation as described in section 3
* Any known bugs or limitations of your program

You must submit your program files online. Create a ZIP/tar archive of your entire submission.

Use T-Square to submit your complete submission package as an attachment.

An example submission may look like as follows -

**pa2.zip**

| -- **pa2/**

| -- **ringo.py**

| -- README.txt/.pdf

| -- sample.txt

We will use an automated script to test the code which may fail if you use a different naming convention.

Only one member of each group needs to submit the actual code and documentation. The other group members can submit a simple text file in which they mention their partner’s name that has submitted the actual assignment.

## 7. Grading

The following table gives the maximum number of points for each component of this programming assignment.

|  |  |
| --- | --- |
| **Task** | **Points** |
| Milestone-1: Design report | 25 |
| Milestone-2: Test peer discovery & RTT matrix exchange | 10 |
| Milestone-2: Test ring formation (no churn) | 10 |
| Milestone-2: Test data transfer (no churn) | 15 |
| Milestone-3: Test ring formation (with churn) | 20 |
| Milestone-3: Test data transfer (with churn) | 20 |

## 8. How to configure the network emulator at your laptop (only if you choose to do so – not required)

The goal of the following rules is to setup a network with the following network artifacts:

● Delayed packet delivery/transmission

● Lower transmission bandwidth

Please note that we may modify these parameters when testing your code. Also, we may add packet duplication and re-ordering.

The tc commands that we execute on the remote servers, are as follows

#parameters

IF = eth0

SUBNET = 130.207.107.12/30

BW = 10mbit

CORRUPT\_PCT = 0%

DELAY\_MEAN = 100ms

DELAY\_STD = 30ms

# tc commands

sudo tc ‐ s qdisc ls dev $IF

sudo tc ‐s filter ls dev $IF

sudo tc qdisc del dev $IF root

sudo tc qdisc add dev $IF root handle 1 : htb

sudo tc filter add dev $IF parent 1 : protocol ip prio 1 u32 flowid 1 : 1 match ip dst $SUBNET

sudo tc class add dev $IF parent 1 : classid 1 : 1 htb rate $BW

sudo tc qdisc add dev $IF parent 1 : 1 handle 1 0: netem delay $DELAY\_MEAN $DELAY\_STD

distribution normal corrupt $CORRUPT\_PCT

Rules for Local Host

The previous rules only work when the client and server run on different physical hosts. In order to test the program on your own computer, you will need to be running a Linux distribution (ubuntu, fedora, centos, mint) and execute these rules with the following parameters

IF = lo

SUBNET = 192.168.56.2/32 # your local IP address

Note that the IP that you use here should be used in your code as well. For example, if you

decide to use 127.0.0.1 in your code, you should use 127.0.0.1/32 here. Do not use the

IP 0.0.0.0 or *localhost* for testing purposes. Feel free to post on piazza if you face any issues.

References

http://www.linuxfoundation.org/collaborate/workgroups/networking/netem

http://onlinestatbook.com/2/calculators/normal\_dist.html