SE424: Distributed Systems	**Recitation 4**
Semester 1 5785	**25 Nov 2024**
Lecturer: Michael J. May	**Kinneret College**

Anti-Entropy Communication

To better understand the behavior of epidemic protocols, you will prepare a small took which works on an anti-entropy based protocol to compute a distributed average. We will build the tool in class and do some distributed experiments with it.

1 Tool Interface

A screen shot of the opening of the app is shown below:

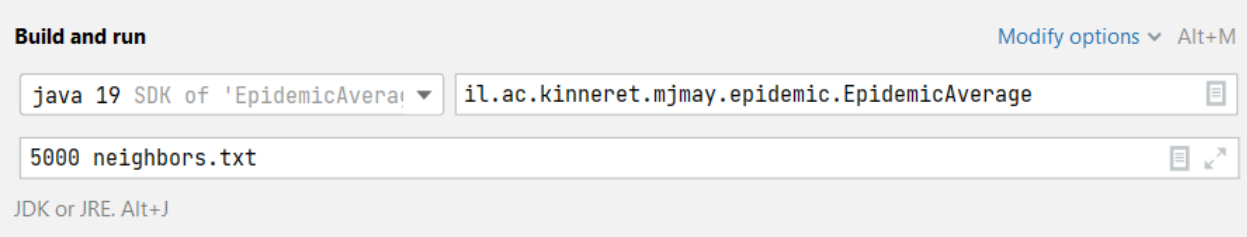
```
EpidemicAverage 4 [Java Application] C:\Program Files\Java\jdk1.8.0_73\bin\javaw.exe
Loaded 4 neighbors
Enter an initial value for the node: 100
How often to synchronize? (s): 10
Enter 's' to stop.
Listening on /0.0.0.0:5003
```

1.1 Command line parameters

The tool will have a command line interface in Java. The tool will accept two parameters:

port The port to listen on

neighborsFilePath The path to the neighbors file



1.2 Runtime parameters

Once started up, the tool will ask for two additional values:

Initial Value The initial value for the node

Sync Interval At what interval to synchronize with a random neighbor (in seconds)

2 Neighbors

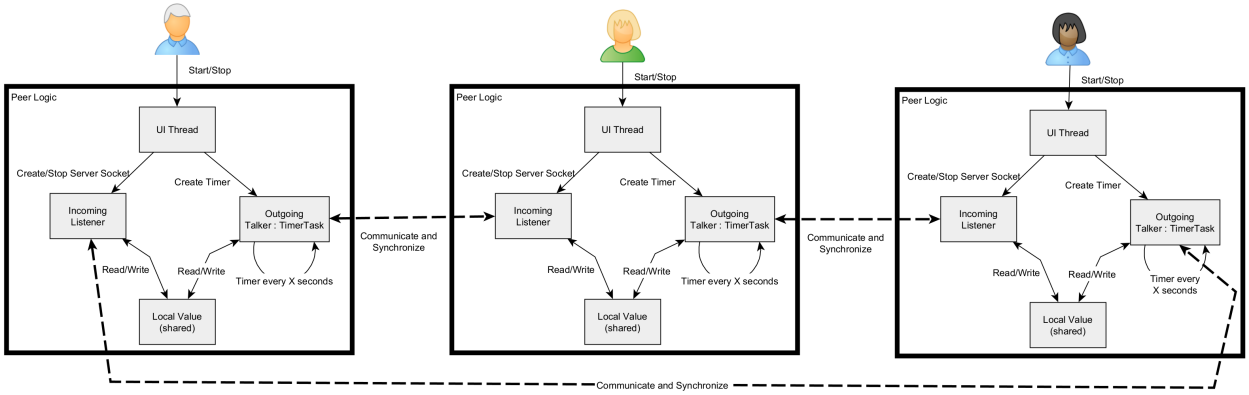
On startup, the tool automatically parses the neighbors file provided and loads the neighbors into a `Vector`. The neighbors file is assumed to have the following format:

```
127.0.0.1:5000
127.0.0.1:5001
127.0.0.1:5002
127.0.0.1:5003
```

Each line contains an IP address and port divided by a colon (:). There shouldn't be any duplicates in the file, but the tool can ignore any such errors.

3 Tool Peer to Peer Architecture

To make the tool work, you'll need at least 3 threads in each node. We'll build the tool using a peer to peer architecture, each node will both initiate outgoing communications. and respond to incoming communications. The basic architecture of the tool is summarized in the following figure:



The tool contains the following threads:

UI thread The thread that parses the neighbors file, allows the user to configure the tool, start, and stop listening

Incoming Listener A Thread that listens on the ServerSocket for incoming conversations. When conversations come in, it acts as the server (passive) side.

Outgoing Talker A Timer Task that initiates outgoing conversations with other nodes on a regular basis. The task is run by a Timer object that configures how often it runs. The Timer Task randomly chooses a neighbor from the neighbors list and acts the client (active) side.

You will also need a place to keep the local value. It will be shared between the two threads, so you'll need to use mutual exclusion to keep it from getting corrupted.

4 Talking and Listening

The tool will automatically listen on the "any" IP address (0.0.0.0) on the port provided in the program parameters. It will start a background thread which listens on the 0.0.0.0 address and port and accepts incoming conversations.

At the same time, the tool will use a `Timer` to initiate an outgoing conversation to a random neighbor. The periodicity of the `Timer` is defined by the interval value the user provided above.

When the user types 's', the tool should shut down the listening and outgoing conversation `Timer` and quit.

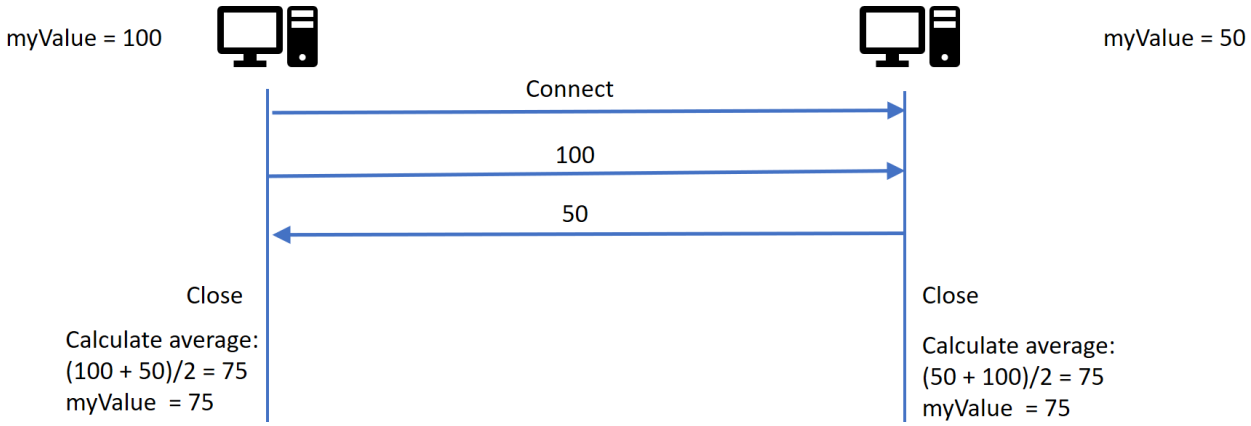
5 Communication Protocol

Ensure that the tool is multi-threaded so that it can both listen and initiate communication at the same time. The listener should be in a different thread to ensure that multiple incoming conversations can happen at once.

The communication protocol is simple. Upon establishing a new session:

1. The initiator sends a single double value (x) (its *current value*).
2. The listener sends a single double value (y) (its *current value*).
3. Both initiator and listener calculate $\frac{x+y}{2}$ and store the result in their respective *current values*.

The protocol is shown graphically in the following figure:



6 Experiments and Modifications

A sample trace of the tool at work is shown below:

```
<terminated> EpidemicAverage 4 [Java Application] C:\Program Files\Java\jdk1.8.0_73\bin\javaw.exe
Loaded 4 neighbors
Enter an initial value for the node: 100
How often to synchronize? (s): 10
Enter 's' to stop.
Listening on /0.0.0.0:5003
Outgoing connection to: /127.0.0.1:5002
127.0.0.1:5002: Received : 525.0 Updated to 312.5
Outgoing connection to: /127.0.0.1:5001
127.0.0.1:5001: Received : 418.75 Updated to 365.625
Outgoing connection to: /127.0.0.1:5002
127.0.0.1:5002: Received : 418.75 Updated to 392.1875
Outgoing connection to: /127.0.0.1:5001
127.0.0.1:5001: Received : 3515.8125 Updated to 1954.0
Outgoing connection to: /127.0.0.1:5002
127.0.0.1:5002: Received : 1954.0 Updated to 1954.0
Outgoing connection to: /127.0.0.1:5000
127.0.0.1:5000: Received : 1954.0 Updated to 1954.0
Outgoing connection to: /127.0.0.1:5000
127.0.0.1:5000: Received : 1954.0 Updated to 1954.0
Outgoing connection to: /127.0.0.1:5000
127.0.0.1:5000: Received : 1954.0 Updated to 1954.0
Outgoing connection to: /127.0.0.1:5002
127.0.0.1:5002: Received : 1954.0 Updated to 1954.0
s
Stopped everything.  Final value is: 1954.0
Goodbye.
```

Use the tool to conduct the following experiments:

1. How is the convergence of the average affected by the interval rate (the wait time) used by the client? Change the interval wait time up and down to see how it affects the convergence. Try having two nodes with a very small time interval (< 1 sec) and the rest with relatively long ones (> 3 sec).

2. How does the topology of the network affect the convergence? Try changing the topology of the neighbors and see how it affects the convergence time.
3. Modify the tool to calculate a distributed maximum in addition to the distributed average.

7 Auto-Pause

Under the tool's current configuration, it will never stop chatting, even when all of the nodes have reached a steady state. To enable the tool to stop performing outgoing conversations when we have reached a steady state, we will implement an "Auto-Pause" feature. Using it will give a gossip-like behavior in which we stop updating others when there is nothing more interesting to talk about.

The auto-pause feature should work with the following behavior:

1. If the most recent value learned via an outgoing conversation is closer to the current value than the difference threshold, outgoing conversations should be paused with probability $\frac{1}{k}$
2. When paused, incoming conversations should still be accepted.
3. If during an incoming conversation received while paused, we receive a value that is more than (difference threshold) units from the current value, the outgoing conversations should be resumed based on the timer.

This requires us to accept another two parameters from the user at start up:

k The value k for determining probabilistically to stop

Threshold The threshold for differences not mattering (keep in mind that double precision division is not precise).

After you implement the auto-pause functionality, the tool's output should look like the following:

```
EpidemicAverage 5000 [Java Application] C:\Program Files\Java\jre1.8.0_231\bin\javaw.exe
Loaded 7 neighbors
Enter an initial value for the node: 100
How often to synchronize? (s): 10
What stopping probability? (<= 1): 0.01
What stopping threshold? (>= 0): 0.001
Listening on /0.0.0.0:5000
Enter 's' to stop.
Outgoing connection to: /127.0.0.1:5000
Value received is within stopping threshold
127.0.0.1:53492: Received : 100.0 Updated to 100.0
127.0.0.1:5000: Received : 100.0 Updated to 100.0
127.0.0.1:53496: Received : 150.0 Updated to 125.0
Outgoing connection to: /127.0.0.1:5000
127.0.0.1:53498: Received : 125.0 Updated to 125.0
Value received is within stopping threshold
127.0.0.1:5000: Received : 125.0 Updated to 125.0
127.0.0.1:53500: Received : 75.0 Updated to 100.0
Outgoing connection to: /127.0.0.1:5001
127.0.0.1:5001: Received : 125.0 Updated to 112.5
Outgoing connection to: /127.0.0.1:5002
Value received is within stopping threshold
127.0.0.1:5002: Received : 100.0 Updated to 106.25
Outgoing connection to: /127.0.0.1:5002
Value received is within stopping threshold
127.0.0.1:5002: Received : 106.25 Updated to 106.25
```

7.1 Experiments

1. Configure the tool to stop with probability 1 and threshold 1. Can you get 5 nodes to enter a situation where they are different by more than 1, but no progress is made?
2. Use a 4 or 5 node group and set each node with a different difference threshold. What is the outcome?

3. Configure a 4 node group with value $k = 2$ and run an experiment until all the nodes are paused. Now run the experiment again with value $k = 10$. What is the difference between the two runs?

