Programming Assignment #1

COEN 241 Introduction to Cloud Computing
Department of Computer Engineering
Santa Clara University

Dr. Ming-Hwa Wang — Spring Quarter 2017
Phone: (408) 805-4175 — Email address: m1wang@scu.edu
Course website: — http://www.cse.scu.edu/~mwang2/cloud/
Office Hours: — Monday & Wednesday 9:00-9:30am

Due date: midnight April 16, 2017

Linda Tuple Space, Part I (200 points)

Please implement the distributed model, Linda, using C/C++ or Java. In a distributed environment, Linda provides a conceptually "global" tuple space (TS) which remote processes can access the matched tuples in TS by atomic operations (in, rd, inp, rdp, out, eval). A tuple is an ordered list of values with types (integer, float, string). A match can be exact match (value only) or variable match (in the form of ?variable_name:type) for all operations. The "out" simply put tuples in TS, and the "eval" is similar to "out" with function in tuples to be evaluated before putting the tuples in TS. The "in" will match and remove tuples from TS, but the "rd" is not destructive. If multiple tuples are available to an "in" or "rd" call, one is selected non-deterministically. If no tuple is available, the "in" or "rd" call blocks (but the "inp" and "rdp" is non-blocking). To guarantee atomicity, a tuple is store at its "home" processor, other processors can use a hash function to access a tuple efficiently (instead of using broadcast). Part I of this assignment implement the basic functions of Linda, and part II supports redundancy and fault-tolerant.

To make the distributed system transparent, you need client-server network programming. To simplify your work, both server and client run on the Linux machines, and no need to support inp, rdp, and eval. You should run P1 with a unique name first and use subcommand (add hosts, the host/server will find an available port automatically for each machine and display the corresponding port number, and in, rd, out). E.g.,

On a machine, e.g., with IP address 129.210.16.80, do:

```
$ P1 host_1
129.210.16.80 at port number: 9998
linda>
```

On another machine, e.g., with IP address 129.210.16.81, do:

```
$ P1 host_2
129.210.16.81 at port number: 3571
linda> add (host_1, 129.210.16.80, 9998)
linda> out("abc", 3)
put tuple ("abc", 3) on 129.210.16.81
linda>
```

Note that for p1, the 'add' subcommand has to be executed before other subcommands and contains a list of (<name>, <IP address>, <port>) as arguments.

On the machine with IP address 129.210.16.80, do:

```
linda> in("abc", ?i:int)
get tuple ("abc", 3) on 129.210.16.81
linda>
```

The add hosts will run the remote execute program to setup client/server program on the hosts and report available port numbers. All those information should save on the hosts' /tmp/<login>/linda/<name>/nets and all tuples should store in /tmp/<login>/linda/<name>/tuples. Make the /tmp/<login>, /tmp/<login>/linda and /tmp/<login>/linda/<name>mode 777, and make nets and tuples mode 666. Which host to store the tuples is decided by hashing, e.g., "md5sum - <<< "<string>", or echo "<string>" | md5sum.

Student Name:
ID:
Score:

Correctness and boundary condition (60%):
Error Handling (5%):
Automatic available port finding and support both host name and IP address (5%):
Display output on both server and client windows whenever there is an event happens (5%):
Compiling without warning messages (5%)
Modular design, file/directory organizing, showing input, documentation, coding standards, sympathy/typing point with README (20%):

Subtotal:

Late penalty (20% per day):
Special service penalty (5%):

Total score: