Programming Homework 1

1. Introduction

This project designs and implements a simplified storage management system. The system consists a number of agents and a manager. An agent is a piece of software that runs on a computer and manages the storage device such as redundant array of independent disk (RAID), which is connected to the computer. The manager is another piece of software that runs by a system administrator who manages the storage devices across the enterprise. Each agent periodically reports the system health information to the manager and executes the commands from the manager.

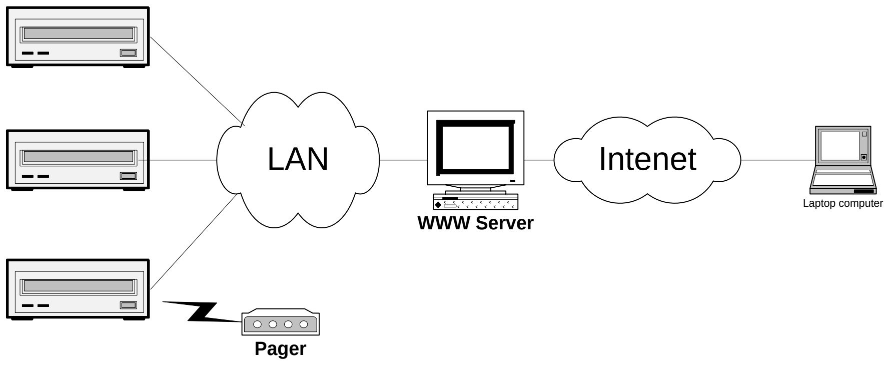

Figure 1. Centralized Storage Management

2. System design

2.1. Agent

The agent starts by launching two threads, BeaconSender and CmdAgent.

BeaconSender sends a UDP packet, referred to as a beacon, periodically (e.g., every 1 minute) to the manager. Each beacon has the following information:

```
struct BEACON
{
        int     ID;             // randomly generated during startup
        int     startUpTime; // the time when the client starts
        int     timeInterval; // the time period that this beacon will be repeated
        char    IP[4];          // the IP address of this client
        int     cmdPort;        // the client listens to this port for manager commands
}
```

The ID is randomly generated when the agent starts. The same ID is used during the life time of the agent. When the agent restarts (e.g., crashes), a new ID is generated. The startUpTime is the time when the agent starts. The timeInterval is the time period that this client will send a beacon. The IP is the IP address of the computer on which the agent is running, and the cmdPort is the port for the manager to send command. A computer may have multiple network interface cards (NICs) and thus have multiple IP addresses. The IP reported in the beacon must be the one by which the manager can communicate with.

CmdAgent listens to a port that the manager sends commands through TCP. When receiving a command, CmdAgent executes the command and returns the result to the manager. In this project, CmdAgent is expected to receive and execute the following commands:

1. void GetLocalOS(char OS[16], int *valid): Here OS[16] contains the name of the operating system where the agent is running, and the integer pointed by valid indicates the execution result. If it is 1, the data in OS is valid.

2. void GetLocalTime(int *time, int *valid): Here the integer pointed by time represents the current system clock, and the integer pointed by valid indicates the execution result. If it is 1, the data pointed by time is valid.

2.2. Manager

The manager starts by launching two threads: AgentMonitor and BeaconListener.

AgentMonitor maintains a list of active agents. Periodically it checks each agent for its heath status. If the time since it receives the last beacon from the agent exceeds 2 times of the timeInterval that it specifies in its beacon, the agent is considered dead. In this case, AgentMonitor prints out an alert message to inform the system administrator.

BeaconListener listens to a UDP port for beacons from all agents. Upon receiving a beacon, it checks if this is from a new agent, in which case it prints a message to inform the system administrator. The beacon is from a new agent if its ID cannot be found in the active agent list, or the ID can be found, but the startup time is different (which indicates the agent dies and restarts somehow).

When a new agent is detected, a new thread CmdAgent is launched to execute two commands, GetLocalOS() and GetLocalTime(), and print out the execution result.

3. Implementation

Agent shall be implemented in C/C++, which allows to access lower-level device information. Manager shall be implemented in Java, which is simple but sufficient to manage the information for each agent.

Important:

The project description (e.g., system functionality, design and implementation) is by no means to be complete. Indeed, you need to think A LOT MORE details in design and implementation. For examples, what ports to be used for UDP and TCP communication, should a new thread need to be launched to process each beacon, and so on. You will also need to think about data format. For example, your OS may be big endian but java is little endian, int in C may be 64 bits but java is 32 bits.

Ask teaching staffs if you have questions.

4. Code Submission and Testing

Please develop your code on pyrite (i.e., using C and Java compilers on pyrite), which runs Linux. Here are the instructions to logon on

1. http://www.cs.iastate.edu/how-use-ssh-windows-users
2. http://www.cs.iastate.edu/how-use-ssh-os-x-users

You can use vim there to edit file. Or edit the file locally, then upload to pyrite using filezilla (make sure to set sftp://pyrite.cs.iastate.edu in Host field).

Submit your design document (e.g., data format, pseudo code) and actual code through your Canvas account by the deadline.

You may be asked to run your code to show it works properly. For examples:

- Run the manager
- Open several consoles
  - Run an agent on each console
  - The manager should be able to detect when a new client shows up by printing out the client's OS and startup time
- Terminate one agent
  - The manager should be able to detect the agent dies by printing out a message
- Terminate one agent and run the agent again immediately
  - The manager should be able to detect the agent dies and resurrects