CIS 505: Software Systems

Project: MMS Mail System (MMSys)

Design Document: Due on 28th March 2007 before class
Source Code: Due on 20th April 2007 by 5 PM

1 Outline

For this project, you are required to implement an email client-server system called *mms mail system*. This system comprises of multiple email clients and servers. Multiple servers are required to support a distributed email system, wherein client accounts are stored in a distributed manner among the various servers. However, for any particular client, all its information (username, password, mailbox) is assumed to be stored on a single server.

Initially, each client connects to one of the email servers and opens a new account. Thereafter, it can send and receive emails to/from other clients in the system. For this purpose, clients must use the SMTP (RFC2821 http://www.ietf.org/rfc/rfc2821.txt) protocol to send emails and the POP3 (RFC1939 http://www.ietf.org/rfc/rfc1939.txt) protocol to receive emails. You are not required to implement all the commands in these protocols. Only those commands that ensure the basic functionality (see Section 2) must be implemented. Since these protocols do not support new client creation, you are free to design your own protocol for the same. Note that to send and receive emails in this system, a client may connect to a server different from the one on which its account was created.

Each email server must satisfy all the requests that a client makes. In addition, it must support multiple client connections at the same time. In our system, we assume that each email server is replicated once to provide fault tolerance. One of them will be the **primary** server and the other will be the **backup** server. You are required to implement a synchronization protocol for the user accounts in the replicated servers. **It is very important to ensure that this protocol is transparent to clients**. Clients are aware of the presence of replicas (this is required so that a client can connect to a backup in case a primary does not respond), but they are not responsible to ensure synchronization between the replicas.

The *mms mail system* must also support distributed servers. Here, we assume that each server (primary or backup) is aware of all the other servers in the system. If a client request cannot be satisfied by a server, then it must communicate with the other servers in the system and satisfy the client request. Note that this communication can be done using SMTP and POP3 protocols.

For example, Figure 1 shows a system comprising of three primary servers S_1, S_2 and S_3 and two clients C_1 and C_2. Assume that client C_1 has an account on server S_1 and client C_2 has an account on server S_2. Now let C_1 send an email to C_2 by connecting to server S_3. For this purpose, C_1 will first authenticate itself. S_3 will connect to server S_1 and verify the authenticity of client C_1. Then, S_3 will receive the email from C_1 and send it to S_2 so that it can be queued in C_2's mailbox. Similarly, when C_2 sends a request to server S_3 to read emails, S_3 obtains C_2's emails from S_2 and sends them to C_2. Although not shown in the figure, each of the primary servers also has a backup which must be synchronized with the primary.

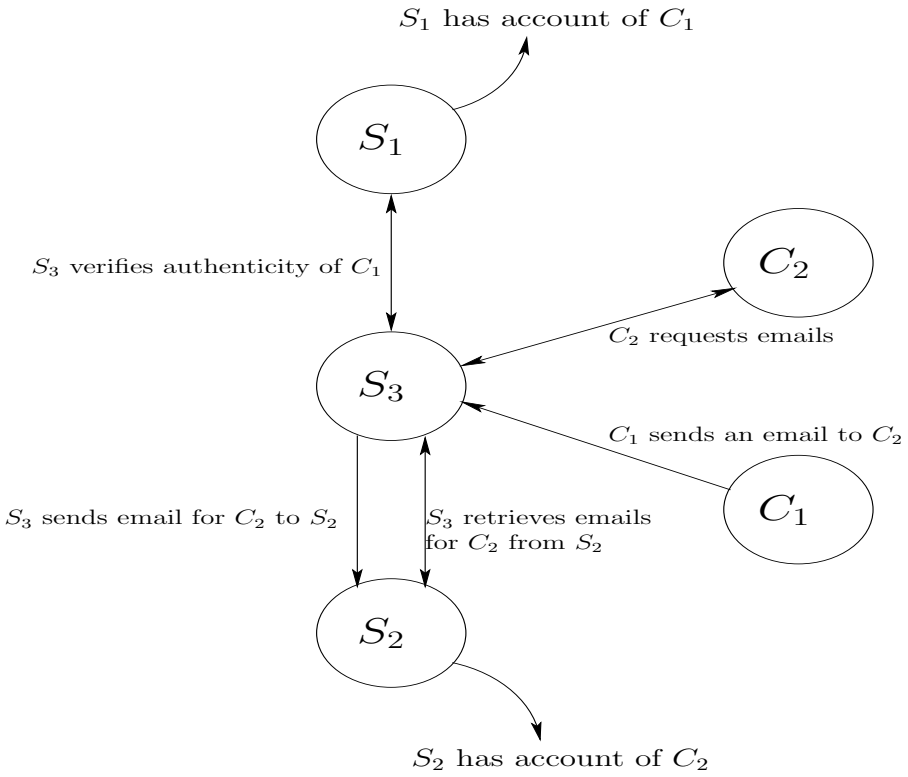

Figure 1: MMSys

2 Snapshot of Project Requirements

2.1 Design Document

- Overall software design of the *mms mail system*
  - Describe the various components and their interactions using figures.
  - Justify your design choices
- List and justify all the assumptions that you make
- Give a CSP Model of your synchronization protocol for replicated servers. You can use abstractions to make it readable
- Describe the advantages and disadvantages of your synchronization protocol

2.2 Email Client

Every client must provide the following functionalities.

- Sending a message to an account supported by the email servers
- Listing messages in inbox or mail-folders
- Reading a message (from inbox or folder)

- Deleting a message (from inbox or folder)
- Creating a folder
- Moving a message to a folder
- Sending a request to create a new account

Client can be a simple command line GUI or open source software. Clients must use SMTP and POP3 protocols for sending and receiving emails, respectively. You are free to design your own protocol for creation of new client accounts.

2.3 Email Server

Servers must support the following functionalities.

- Fulfillment of all client requests
- Management of client accounts including creation of new accounts
- Client username verification prior to sending emails from the client
- Client username and password verification prior to transferring emails from the client's mailbox to the client
- Support for multiple active clients (clients of different users)
- Support for distributed servers

Servers can manage mailboxes using either the native file system or any database system.

2.4 Server Fault Tolerance

You must provide fault tolerance for servers using replication. Assume every server has **one** replica. One of them will be the **primary** server and the other will be the **backup** server. You must implement a synchronization protocol to ensure that the user accounts in the primary and backup are synchronized.

2.5 Distributed Servers

You must implement a distributed server system. Client mailboxes and their account information must be distributed over multiple servers. Assume that for a particular client, its mailbox and account information are located on the same server.

The distributed server system must be transparent to clients. In other words, a client can send and receive emails to/from any of the servers, even to/from ones which do not have its account information. The servers must communicate with each other using SMTP and POP3 protocols to satisfy client requests. For this purpose, you can assume that servers are aware of all the other servers in the network. Your implementation must be correct under all of the following situations.

- Client C1 sends an email to client C2 by connecting to server S1 and,

  - S1 has account information for both C1 and C2
  - S1 has account information for C1 but not for C2
  - S1 has account information for C2 but not for C1
  - S1 does not have account information for either C1 or C2

- Client C1 receives emails by connecting to server S1 and,
  - S1 has account information for C1
  - S1 does not have account information for C1

3 Guidelines

3.1 Program Inputs

Your email server must be started with at least the following command line inputs.

- IP addresses and port numbers of all the other primary and backup servers in the system

Your email client must be started with at least the following command line inputs

- IP addresses and port numbers of the primary and backup servers to which this client should connect to create a new account.
- IP addresses and port numbers of the primary and backup servers to which this client should connect to send and receive emails.

You are free to have other command line inputs in addition to the ones mentioned above.

3.2 Assignment Requirements

The project consists of the following three parts:

- **Design Document:** This document is **due on 28th March 2007 before class**.
- **Source Code:** The program may be written in C, C# or Java. Make sure to include a README that explains how to run your implementation. This portion of the project is **due on 20th April 2007 by 5 PM**.
- **Demo:** Each project group is required to give a demonstration of their mail system. Details regarding date and time will follow.

3.3 Project Groups

You can form a group of 2 or 3 people for this project. You can decide on the groups amongst yourselves and inform the TA's through email (*arney@cis.upenn.edu, arvinde@cis.upenn.edu*). Please send only one email per group. In the email include the names and eniac login ids of all your group members. If anyone is not able to form a group then they can either use the newsgroup or email the TA's for help. This process should be completed on or before **19th March 2007**. The TA's will then assign a number to each group.

3.4 Testing/Evaluation

Every project group must use SMTP and POP3 protocols for sending and receiving emails, respectively. Furthermore, client authentication must also be based on these protocols. However, each group is free to implement their own protocols for new client creation and for ensuring synchronization between server replicas.

We will be cross testing your implementation between groups. In other words, we will test the client code of one group with the server code of another and vice versa. For this purpose, your email servers must have certain client accounts by default, i.e., they must already be present in one of your servers. These accounts are **instructor@mmsys.upenn.edu, ta@mmsys.upenn.edu and cis505@mmsys.upenn.edu**. Also set their passwords to **CIS505**. This is required for cross testing because we expect each group's new client creation protocol to be different.

3.5 Turning in

The assignment files (design document and source code) must be submitted as a zipped file to blackboard. You must submit to the digital drop box, under tools. Make sure to **SEND** the file to the dropbox. Merely adding it to the dropbox **DOES NOT** submit the files. Please make sure to name your design document and source code submissions as groupX_design and groupX_source, respectively, where X denotes your group number.