Implementation of Simple Mail Transfer Protocol (SMTP) for Mail System in OceanStore

Khanh Chau  
khanh@seas.ucla.edu  
Computer Science Division, U.C.Berkeley

Abstract:

OceanStore is currently being developed at the U.C. Berkeley Computer Science Division. OceanStore is a global persistent data storage designed to scale to billions of users and stored objects with high availability and is peer-to-peer technology based. Applications need to be built in OceanStore to test the OceanStore Client API(Application Programming Interface) and also to examine the conflict resolution strategies. In this paper, we build an e-mail infrastructure using OceanStore. This e-mail application enables delivery and retrieval for billions of clients. This report describes the implementation of the SMTP front-end, one of the three components of the mail infrastructure. SMTP allows clients to send mail messages to multiple recipients.

1. Introduction:

Nowadays, Internet services are in great demand. One of the most popular Internet services is electronic mail (e-mail). There are many e-mail services that are freely available to Internet users such as Yahoo, Hotmail, Juno, etc. However, not all of these services provide efficient availability or scalability, which are the two most important features for any good Internet services. Scalability is the ability to incrementally support larger user populations by adding more computers. For example, it should not be surprising when one tries to log in to the Yahoo!Mail account, one gets denied and is asked to try again later. It happens because there might be too many users trying to connect to Yahoo!Mail server at once. Since Yahoo does not adequately support scalability, the account access is denied.

Availability is also as important. It is based on resource sharing technique. That is a good Internet service has to be designed such that users/customers can ALWAYS have access to their data. Suppose a user A wants to retrieve his/her mail in the network, and
his/her mail is currently stored in server B. Even if server B dies or fails to response, user A should still be able to send or receive mail sufficiently from some other servers in the network. Another major concept of availability is access time. Good Internet services should route users/clients to the shortest location of their data. This guarantees time efficiency for both clients and server, and further avoids network traffic congestion. In addition, users should be able to access their mail from anywhere: watches, palms, laptops, cellular phones.

The e-mail infrastructure proposed in this paper is built on top of OceanStore because by doing so, it gets all the benefits of high availability of data, automatic archiving, and global scalability that are built into OceanStore

1.1 Introduction to OceanStore - a true data utility:
OceanStore was initiated at the UC Berkeley Computer Science Division and is currently in the process of developing and prototyping architecture to construct Internet services.

OceanStore is a global persistent data store [1] designed to scale to billions of users. It provides a consistent, highly available, and durable storage utility. OceanStore also maintains its users’ privacy through end-to-end encryption.

The OceanStore infrastructure ensures that data remains uncorrupted and widely available; and the system replicates the data, rendering manual backup unnecessary in order to provide high availability.

If part of the infrastructure were compromised or retired, the system would restore replication levels automatically. The system requests data regardless of its physical location and move copies near the user. With this design, not only can OceanStore guarantee data availability, but it also can route users to the nearest file location whenever users request. Furthermore, if users lose their local copy due to administration or hardware failure, the system can immediately supply a new copy.

OceanStore relies on peer-to-peer (P2P) technology. P2P is [3] a class of applications that takes advantage of resources -- storage, cycles, content, human presence -- available at the edges of the Internet. P2P networks essentially eliminate the need for servers and allow all computers to communicate and share resources as peers. This compelling technology is on the rise today largely because of the advantages that peer
networks offer in terms of collaborating on specific tasks. There are a great number of applications relying on P2P such as: Napster, Popular Power, Freenet, AIMster, and Groove.

OceanStore assumes that the infrastructure is fundamentally un-trusted. That is administration might fail or servers might crash without acknowledgement, or in the worst case, there could be malicious users eavesdropping and modifying data. To prevent those problems from happening as well as to protect users’ privacy, all the information entering the infrastructure must be encrypted in OceanStore system. The encryption key is assigned only to the user who has read permissions.

OceanStore's Tapestry [4] is a self-organizing routing and location subsystem. Tapestry works as an overlay network on top of Internet Protocol (IP), using distributed data structure to track file locations. When one requests one's data, Tapestry helps to route that person to the nearest file location. This routing ability reduces network congestions as well as connection failures due to travel distance.

Figure 1.1: The OceanStore systems. The core is composed of potentially thousands or millions of highly connected “pools” or storage domains, among which data freely flows. Clients connect to one or more pools, perhaps intermittently.
As Figure 1 shows, OceanStore consists of millions of individual servers, each cooperating to provide the service. A group of such servers is a pool. Data flows freely between these pools, allowing replicas of a data object to exist anywhere, at any time, to protect data in case of administration failure or some corrupted servers.

An OceanStore file is stored as an object, which composed of blocks having version numbers and can be modified only at the block level. Every object has a unique AGUID (Archival Globally Unique IDentifier). In addition, every object has VGUID (Version Globally Unique IDentifier) An update in OceanStore is an ordered list of predicate-action tuples. A predicate is a boolean expression of clauses that check for object’s version number, file size, etc. Action is a list of modification such as appending, deletion, insertion, and replacement. To apply an update, a replica evaluates each predicate in turn and applies the action when the predicate is satisfied and the update is said to “commit”. If no predicates are satisfied, the update is said to “abort”.

1.2 SMTP in OceanStore:

OceanStore can be used as a toolkit to create highly scalable, and available Internet services. Now the question comes: Is OceanStore and its API’s general enough to build any Internet application? The only way we can answer this question correctly and efficiently is to actually build applications on top of OceanStore. As a result, my project, with collaboration of Jeff Pang and Steven Czrenwinski, is to build an e-mail infrastructure in OceanStore.

2. Related Work

There have been numerous research projects on Internet-accessible mail applications (or set of applications), which are scalable (able to support many thousands of concurrent users) and highly available (resilient to network and hardware outages).

For instance, NINJA was an Internet service toolkit that provided availability and scalability mechanisms. The NINJA project [5] was developed at University of California, Berkeley in Computer Science Department. Ninja Mail was built using Ninja to provide both scalability and availability allowing Internet users access their mail from
anywhere. For example, users/clients can check on their e-mail simply by calling a special number from a cell phone, or equivalently by sitting down at any Internet-connected PC in the world. NINJA supports availability and scalability, but it was not built based on P2P technology. As a result, NINJA had limited availability.

There are developed projects that use peer-to-peer technology. One of them is CHORD project [6], successfully developed at MIT. It aims to build scalable, robust distributed systems using peer-to-peer concepts. CHORD has many similarities with OceanStore. It distributes the load of serving data widely. It replicates all data, and maintains that replication as nodes fail and re-join the system.

3. SMTP Architecture:

E-mail infrastructure contains three components: IMAP front-end, SMTP front-end, and OceanStore Mail Layer. The SMTP server handles mail delivery. The IMAP server allows users to access e-mails, and lastly Mail Layer is the bridge to transform mail objects and requests to their underlying native OceanStore Object representations.

Figure 2.1 shows an overview of the system’s architecture.

---

**Figure 2.1:** Mail System Application in OceanStore. It consists of three components:

IMAP front-end, SMTP front-end, Mail Layer
3.1 SMTP - What is it?

Simple Mail Transfer Protocol (SMTP) [7] is an application-level protocol enabling clients to send e-mail messages. SMTP is a server-client connection based. SMTP is implemented over TCP/IP (Transmission Control Protocol/Internet Protocol).

In this project, I built SMTP server using Sandstorm, current prototype of a SEDA-based services platform, which uses NBIO( Non-Blocking Input Output), which is integrated into OceanStore. SEDA [8] is an acronym for staged event-driven architecture. All events are queued and passed through a set of stages to completion. This design avoids the high overhead associated with thread-based concurrency models, and decouples event and thread scheduling from application logic.

3.2 How does SMTP work?

SMTP servers and clients basically communicate via certain fixed non-case-sensitive commands. SMTP clients send commands, and the servers responses accordingly.

When a client requests an SMTP session, SMTP server will open a TCP/IP connection. Once the transmission channel is established, SMTP service can then be divided into 5 main stages:

1. Clients send a HELO/EHLO command to the server to politely instantiate mail transaction.

2. Clients send a MAIL command indicating the sender of the mail. SMTP server checks if this sender in deed exists. In case of sender's non-existance, an error message will be generated to the client.

3. Clients send the RCPT command identifying a recipient of the e-mail. Multiple recipients can be specified by sending multiple RCPT commands.

4. DATA command is sent from clients to signify the server content of the email will be sent next. The end of mail data indicator, which is a dot on a new line and is followed by a return, confirms the mail transaction and tells the receiver-SMTP to now process the stored recipients and mail data.

5. a QUIT command stops SMTP service. Server-client connection ends.
SMTP server always responds with an “OK” reply for syntactically supported request or it will send an error message accordingly.

The following table demonstrates a typical SMTP session. S and C denote server and client respectively.

| S: 220 portland.cs.berkeley.edu SMTP Services ready |
| C: HELLO portland.cs.berkeley.edu |
| S: 250 Hello seas.ucla.edu, please to meet you |
| C: MAIL FROM: khanhlc@eecs.berkeley.edu |
| S: 250 khanhlc@eecs.berkeley.edu... Sender ok |
| C: RCPT TO: oceanstore-developers@cs.berkeley.edu |
| S: 250 oceanstore-developers@cs.berkeley.edu... Recipient ok |
| C: DATA |
| S: 354 Enter mail end with '.' on a line by itself |
| C: <enter data> |
| S: 250 NAA21547 Message accepted for delivery |
| C: QUIT |
| S: 221 closing connection.... |

**SMTP Design:**

I implemented a front-end to parse and to response to the SMTP requests.

I also have the OceanStore tie-ins at the Rcpt command and at the mail transfer process. After getting a “rcpt to:” command, the SMTP will check with OceanStore to see if this recipient is a valid OceanStore customer.

During the mail transfer, SMTP interacts with OceanStore. Once SMTP has the complete message it will send requests to Mail Layer for creating new header and body content objects for this message. An additional append request to Mail Layer is needed to append the pointers to the recipient’s mailbox.

User’s mailbox is an OceanStore object. An e-mail message consists of two parts: a header and a body content. A mailbox is simply a list of pointers to headers and body contents of mail messages.

Sending a new e-mail can be achieved through three steps:

1. Create a new header object with the associated information
2. Create a new body content with the message attached
3. Create two pointers to point to these header and body content and then append the pointers to the user’s mailbox.
To send a new mail
1. Create new header object
2. Create new body-content object
3. Append their pointers to Mailbox

**Figure**: mail sending procedure

---

**To send a new mail**

1. Create new header object
   - From: john
   - To: oceanstore
   - Subj: question

2. Create new body-content object
   - Hello oceanstore developers
   - This is a test

3. Append their pointers to Mailbox

**Figure**: Diagram of the implementation of SMTP command stages

---

To send a new mail
1. Create new header object

**New header**
- From: john
- To: oceanstore
- Subj: question

**New Body-content**
- Hello oceanstore developers
- This is a test

**Append**
- Header1.0
- Body 1.0
- Header2.0
- Body 2.0
- Header3.0
- Body3.0

---

**Figure**: Diagram of the implementation of SMTP command stages

---

**To send a new mail**
1. Create new header object
2. Create new body-content object
3. Append their pointers to Mailbox

**END**

QUIT cmd /221

**S0**: setup aTCP conn in sandstorm stage

**S1**: ready for MAIL command

**S2**: ready for RCPT cmd

**S3**: ready for DATA cmd

**S4**: ready to accept msg

**S5**: quit/send mail again

---

**Figure**: Diagram of the implementation of SMTP command stages
3.3 Handling Conflicts:
Conflicts may occur while sending or receiving e-mail messages whenever two different parties try to update the same object.

Consider this example: Currently my mailbox has a version of 1.5. Suppose now multiple senders A, B, and C all send a piece of mail to my mailbox at the same time specifying append action needed to version 1.5. Of course, all the messages will not arrive at the same time. Say, message from sender A arrives first and it will be accepted, and my mailbox updates its version to 1.6. Now conflict occurs. Messages from senders B and C will be rejected since they specify the wrong version number.

This type of conflict can be handled by resubmitting the message again with the right predicate specification.
To resolve these conflicts, the Mail Layer will first requests a new version number of object M from OS, which is 1.6. Then it appends this new version 1.6 to update B. Finally, Mail Layer resubmits update B. By doing this, now update B has the correct version number so it can be committed.

4. Experiments and Results:
I have built a client running on this SMTP in order to measure its performance.

Chart 4.1: Performance of each stage in SMTP Server
This chart shows the performance in each stage. As you can see, the helo cmd takes a bit of time, around 10% of the entire message transaction. Except the transfer process, the rest of stage has very little affect to the transaction. The transfer dominates the time to
completely send a message from the beginning of the connection to the end. This amount of time is reasonable since in the transfer process, we have to send three requests to Mail Layer and have to wait until all results of these requests come back.

I also measure the time spent to send messages with different sizes.

![Graph 4.2: Time Spent for sending messages with different sizes]

As the graph indicates, it takes in the range of 300-400 milliseconds to send messages with sizes less than 1 MB. Then the time starts increase obviously when we get to the message size of 1 MB and more.

The reason for this increase is as followed. We know that transfer process dominates the transaction. In the transfer process, there are many overheads that OS have for update and append requests for the header and body contents. The time spending on these overheads is relatively large when message size is small. However, when a message gets to 1 MB or more, handling the message itself takes time and it makes the overheads’ time relatively small in the transaction. When it is above 1 MB, the message size has important role in time-spent measurement.

**5. Conclusions and Future Directions**

This paper has described the overall design of OceanStore and the implementation of SMTP front-end for an e-mail infrastructure using OceanStore. I have shown that the
SMTP server is possible to build. This SMTP module will be integrated with the other two components, the Mail Layer and IMAP Front-End to complete the e-mail application in OceanStore. In future work, this SMTP server’s design can be improved by increasing security level allowing only authorized users to access. Besides, e-mail messages should be encrypted to ensure the security and privacy. Spam prevention help recipients to prevent a large amount of unwanted e-mail. Furthermore, batched updates will significantly enhance the performance of SMTP server. Instead of sending the header and body content of a message separately, it is more efficient to batch them and send to them OceanStore once.

6. Acknowledgements

I would like to thank all the people, especially the OceanStore group, who made my summer experience at Berkeley educational and enjoyable. I would like to give a special thank to Professor Joseph, my faculty mentor, for giving me valuable advise and the opportunity to work in his lab. I sincerely appreciate Professor Kubiatowicz’s guidance to have a successful research. A warm thank goes to Steve Czerwinski, my graduate mentor, for his enthusiasm to share knowledge with me. With his support, I was able to get through my project design.

I would like to thank all the SUPERB staff, especially Dr. Sheila Humpheys, Marie Mayne, Erika Tate, and Jeannie Lee, for making the program an enjoyable and unforgettable experience.

Lastly, thanks to my SUPERB group. You all are great and fun to be around with. I hope to see you all again possibly in graduate school.
References


[5]. The NINJA project. http://ninja.cs.berkeley.edu

[6]. The CHORD project. http://www.pdos.lcs.mit.edu/chord
