Database based Email System

Edara Pavan   Ahmed Shareef   Vaibhav Selot

November 26, 2004

Outline

Outline

1. Background
    - Motivation
    - System Architecture
    - IMAP server
2. LDAP Authentication
    - Directory structure
    - Directory entries
    - Authentication
3. Data Design
    - Database schema
4. Detailed design
    - Additions to IMAP
    - Mail Searching
    - Misc.
    - Squirrelmail Interface

Motivation

- To provide a means of email access to end-users through a command line interface or an RFC-compliant client.
- Email systems store emails on a flat file. They use operating system support for file access to service requests for messages by clients.

Why Database?

Database offers superior techniques for search than normal file system.

Motivation

- To provide a means of email access to end-users through a command line interface or an RFC-compliant client.
- Email systems store emails on a flat file. They use operating system support for file access to service requests for messages by clients.

Why Database?

Database offers superior techniques for search than normal file system.

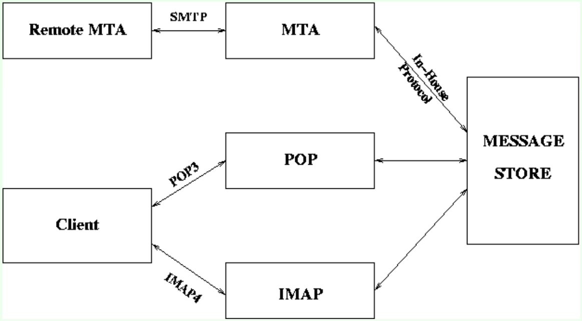
Motivation

- To provide a means of email access to end-users through a command line interface or an RFC-compliant client.
- Email systems store emails on a flat file. They use operating system support for file access to service requests for messages by clients.

Why Database?

Database offers superior techniques for search than normal file system.

System Architecture



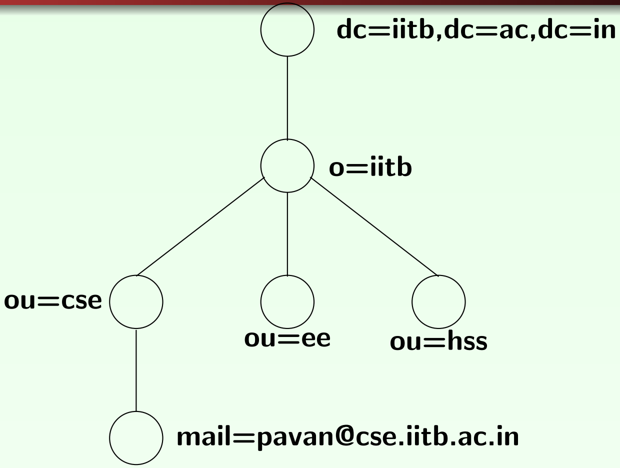
IMAP server

- Access to e-mail message stores as if they were local.
- User works on the message store itself, so requires persistent connection to the server.
- Each user can have multiple folders (INBOX, Trash, Sentmail, Drafts)
- Simultaneous access to the mailbox from two different locations (laptop, desktop etc.)
- Typically listens on port 143.

Outline

1. Background
    - Motivation
    - System Architecture
    - IMAP server
2. LDAP Authentication
    - Directory structure
    - Directory entries
    - Authentication
3. Data Design
    - Database schema
4. Detailed design
    - Additions to IMAP
    - Mail Searching
    - Misc.
    - Squirrelmail Interface

Directory structure



DIT for Email application

Directory entries

Sample ldif entry for a user:

dn: mail=pavan@cse.iitb.ac.in, ou=cse, o=iit, dc=iitb, dc=ac, dc=in
objectclass: account
objectclass: mailAccount
objectclass: top
userpassword: {md5}aknbKIfeaxs
uid: pavan
mail: pavan@cse.iitb.ac.in
mboxid: 20001

Authentication

- User requests a connection to his mailbox by sending login and password
- IMAP server initializes a handle to the ldap server - **ldap_init**
- IMAP server binds the connection with ldap user name and password - **ldap_bind**.
- Call the synchronous search, **ldap_search_s** with the email username to get the password and password type.
- Convert the entered password to the password type and authenticate if the user is legitimate (if it exists at all).
- Search also returns mailbox id for use by IMAP server to connect to the message store database.

Authentication

- User requests a connection to his mailbox by sending login and password
- IMAP server initializes a handle to the ldap server - **ldap_init**
- IMAP server binds the connection with ldap user name and password - **ldap_bind**.
- Call the synchronous search, **ldap_search_s** with the email username to get the password and password type.
- Convert the entered password to the password type and authenticate if the user is legitimate (if it exists at all).
- Search also returns mailbox id for use by IMAP server to connect to the message store database.

Authentication

- User requests a connection to his mailbox by sending login and password
- IMAP server initializes a handle to the ldap server - **ldap_init**
- IMAP server binds the connection with ldap user name and password - **ldap_bind**.
- Call the synchronous search, **ldap_search_s** with the email username to get the password and password type.
- Convert the entered password to the password type and authenticate if the user is legitimate (if it exists at all).
- Search also returns mailbox id for use by IMAP server to connect to the message store database.

Authentication

- User requests a connection to his mailbox by sending login and password
- IMAP server initializes a handle to the ldap server - **ldap_init**
- IMAP server binds the connection with ldap user name and password - **ldap_bind**.
- Call the synchronous search, **ldap_search_s** with the email username to get the password and password type.
- Convert the entered password to the password type and authenticate if the user is legitimate (if it exists at all).
- Search also returns mailbox id for use by IMAP server to connect to the message store database.

Authentication

- User requests a connection to his mailbox by sending login and password
- IMAP server initializes a handle to the ldap server - **ldap_init**
- IMAP server binds the connection with ldap user name and password - **ldap_bind**.
- Call the synchronous search, **ldap_search_s** with the email username to get the password and password type.
- Convert the entered password to the password type and authenticate if the user is legitimate (if it exists at all).
- Search also returns mailbox id for use by IMAP server to connect to the message store database.

Outline

1 Background
- Motivation
- System Architecture
- IMAP server

2 LDAP Authentication
- Directory structure
- Directory entries
- Authentication

3 Data Design
- Database schema

4 Detailed design
- Additions to IMAP
- Mail Searching
- Misc.
- Squirrelmail Interface

Database tables

- Mail-Message Table.
  Contains an entry for each message received by the SMTP server.
- Mail-Flag Table.
  Flags of each message in user's mailbox.
- User-Table.
  Contains for each mailbox, the number of messages it contains.
- CC Table
- To Table
- From Table
  Contain for each message, the list of users whose address is in To, Cc or From Field.

Database tables

- Mail-Message Table.
  Contains an entry for each message received by the SMTP server.
- Mail-Flag Table.
  Flags of each message in user's mailbox.
- User-Table.
  Contains for each mailbox, the number of messages it contains.
- CC Table
- To Table
- From Table
  Contain for each message, the list of users whose address is in To, Cc or From Field.

Database tables (Contd.)

- User-Label Table.
  For each mailbox, contains the set of defined labels.
- Message-Label Table.
  For each message, contains the set of applied labels.
- Attachment Table.
  For each message the list of attachments.
- Filter Table.
  For each mailbox, the set of defined per-user filters.

Database tables (Contd.)

- User-Label Table.
  For each mailbox, contains the set of defined labels.
- Message-Label Table.
  For each message, contains the set of applied labels.
- Attachment Table.
  For each message the list of attachments.
- Filter Table.
  For each mailbox, the set of defined per-user filters.

Database tables (Contd.)

- User-Label Table.
  For each mailbox, contains the set of defined labels.
- Message-Label Table.
  For each message, contains the set of applied labels.
- Attachment Table.
  For each message the list of attachments.
- Filter Table.
  For each mailbox, the set of defined per-user filters.

Structure of the tables

Mail-Message	
FieldName	Type
Message-ID	String
Subject	String
Date	Date
Blob path	String
Count	Int
Size	Int

User-Label	
FieldName	Type
Mail-Box-ID	Int
Label-ID	Sequence
Label	String

Message-Label	
FieldName	Type
Mail-Box-ID	Int
Message-ID	String
Label-ID	Sequence

Structure of the tables

To	
FieldName	Type
Message-ID	String
To	String

From	
FieldName	Type
Message-ID	String
From	String

Cc	
FieldName	Type
Message-ID	String
CC	String

Attachments	
FieldName	Type
Attachment-ID	Int
Message-ID	Int
Size	Int
Attachment-Name	Int
Pointer	Int

Structure of the tables (Contd.)

User-Table	
FieldName	Type
Mail-Box-ID	String
User-ID	Int
Count	Int

Filter	
FieldName	Type
Mail-Box-ID	Int
Filter-ID	Int
Seive-Rule	String

Outline

1. Background
    - Motivation
    - System Architecture
    - IMAP server
2. LDAP Authentication
    - Directory structure
    - Directory entries
    - Authentication
3. Data Design
    - Database schema
4. Detailed design
    - Additions to IMAP
    - Mail Searching
    - Misc.
    - Squirrelmail Interface

Support for Labels

DEFINELABEL $< label >$
: Defines a label for the user. An entry is added to the User-Label table if this is not a duplicate.

UNDEFLABEL $< label >$
: Removes a label defined for the user. Deletes the entry from the User-Label table.

ADDLABEL $< msg - num > \quad < label >$
: Defines a label for the message in the particular user's mailbox. Gets the corresponding message-id and adds an entry to Message Label table.

REMLABEL $< msg - num > \quad < label >$
: Removes a label from the message. Deletes the corresponding entry from the Message Label table.

Support for Labels

DEFINELABEL $< label >$
: Defines a label for the user. An entry is added to the User-Label table if this is not a duplicate.

UNDEFLABEL $< label >$
: Removes a label defined for the user. Deletes the entry from the User-Label table.

ADDLABEL $< msg - num >$ $< label >$
: Defines a label for the message in the particular user's mailbox. Gets the corresponding message-id and adds an entry to Message Label table.

REMLABEL $< msg - num >$ $< label >$
: Removes a label from the message. Deletes the corresponding entry from the Message Label table.

Support for Labels

DEFINELABEL $< label >$
: Defines a label for the user. An entry is added to the User-Label table if this is not a duplicate.

UNDEFLABEL $< label >$
: Removes a label defined for the user. Deletes the entry from the User-Label table.

ADDLABEL $< msg - num > \quad < label >$
: Defines a label for the message in the particular user's mailbox. Gets the corresponding message-id and adds an entry to Message Label table.

REMLABEL $< msg - num > \quad < label >$
: Removes a label from the message. Deletes the corresponding entry from the Message Label table.

Support for Labels

DEFINELABEL $< label >$
: Defines a label for the user. An entry is added to the User-Label table if this is not a duplicate.

UNDEFLABEL $< label >$
: Removes a label defined for the user. Deletes the entry from the User-Label table.

ADDLABEL $< msg - num >$ $< label >$
: Defines a label for the message in the particular user's mailbox. Gets the corresponding message-id and adds an entry to Message Label table.

REMLABEL $< msg - num >$ $< label >$
: Removes a label from the message. Deletes the corresponding entry from the Message Label table.

Support for Labels (Contd.)

SEARCH LABEL "String"
: Returns all message numbers with the given label in the user's mailbox

SEARCH ATTACHNAME attch-name
: Returns message numbers of all messages in the user's mailbox having an attachment with the given name

SEARCH INATTACH attch-name(optional) string
: Returns message numbers of all messages in the mailbox with the given word(s) in the attachment.

SEARCH FROM “cs701@cse.iitb.ac.in”

```
SELECT COUNT(*)+1 FROM
(SELECT * FROM Mail_Message mm,Mail-Flag mft
    WHERE mm.Message-ID=mft.Message-ID
    AND mft.Mail-Box-ID=mailboxid)m,
(SELECT msg.date
    FROM From f,Mail_Message msg,Mail-Flag mf
    WHERE f.From="cs701@cse.iitb.ac.in"
    AND f.Message-ID=msg.Message-ID
    AND msg.Message-ID=mf.Message-ID
    AND mf.Mail-Box-ID=mail-box-id)t
WHERE m.Mail-Box-ID=mailboxid
    AND convert(m.date)<convert(t.date)
    group by t.date;
```

Search by Keywords

SEARCH INATTACH “AttachName” “string”

```
SELECT line_number(msg.DATE)
FROM Mail-Message msg,Attachments a,
      Mail-Flag mf where (natural join)
AND a.AttachName=AttchName
AND search(a.AttachName,"string")==1;
```

- SEARCH BODY “string” uses similar query on the body.

Searching Mail

SEARCH BEFORE date	msg.Date<date
SEARCH AFTER date	msg.Date>date
SEARCH SMALLER size	msg.Size<size
SEARCH LARGER size	msg.Size>size
SEARCH DELETED(or any other flag)	Flags&&128 == 1

Search by Labels

SEARCH LABEL “LabelName”

```
SELECT line_number(msg.date)
FROM Mail-Message msg,User-Label u,Message-Label m
WHERE (natural join)
      AND u.Mail-Box-ID=mailboxid
      AND u.Label="LabelName";
```

LIST * Lists all the Labels of a User.

GETLABELS “Msg-ID” Lists the labels attached to a Msg-ID

Other Commands

- STORE "Msg-ID" +FLAGS (\DELETED)
  Message will be set for deletion.
- EXPUNGE
  - Look for all the messages in the mailbox with the deleted flag set.
  - For each message, decrement the count in the corresponding entry in the Mail-Message table by one.
  - To reduce the **expunge** time, entry in the database is not deleted immediately when count becomes zero.
  - Instead a **Garbage Collector** cleans up the orphaned blobs at some specified intervals or when the system load is low.
- LOGOUT
  - Expunge messages.
  - Logs out.

Displaying MailBox

- SELECT INBOX
- FETCH 1:* (FLAGS BODY[HEADER.FIELDS (FROM SUBJECT DATE SIZE)])

Labels

Client Side Action	Server Side Command
Create Label	DEFINE LABEL
Remove Label	UNDEF LABEL
Rename Label	REM LABEL
Add Label To a msg	ADDLABEL "String"
Remove Label from a msg	REMLABEL <Msg-ID> "String"

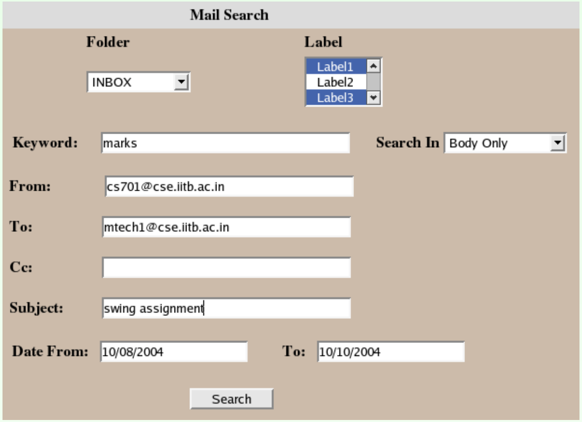
What are the available Labels?

LIST *

What are the labels associated with a message?

GETLABEL <Msg-ID>

Searching



Mail Search

Folder: INBOX

Label: Label1 [✓], Label2, Label3 [✓]

Keyword: marks

Search In: Body Only

From: cs701@cse.iitb.ac.in

To: mtech1@cse.iitb.ac.in

Cc:

Subject: swing assignment

Date From: 10/08/2004 To: 10/10/2004

Searching

- Complex Query

Get msgs from cs701@cse.iitb.ac.in having label "Swing"

From "cs701@cse.iitb.ac.in"

Client: SEARCH FROM "cs701@cse.iitb.ac.in"
Server: OK 12 16 17 19 24

Label "Swing"

Client: SEARCH LABEL "Swing"
Server: OK 11 16 19 25

- Messages that match - 16, 19

Searching

- Complex Query

Get msgs from cs701@cse.iitb.ac.in having label "Swing"

From "cs701@cse.iitb.ac.in"

Client: SEARCH FROM "cs701@cse.iitb.ac.in"
Server: OK 12 16 17 19 24

Label "Swing"

Client: SEARCH LABEL "Swing"
Server: OK 11 16 19 25

- Messages that match - 16, 19

Searching

- Complex Query

Get msgs from cs701@cse.iitb.ac.in having label "Swing"

From "cs701@cse.iitb.ac.in"

Client: SEARCH FROM "cs701@cse.iitb.ac.in"
Server: OK 12 16 17 19 24

Label "Swing"

Client: SEARCH LABEL "Swing"
Server: OK 11 16 19 25

- Messages that match - 16, 19

Searching

- Complex Query

Get msgs from cs701@cse.iitb.ac.in having label "Swing"

From "cs701@cse.iitb.ac.in"

Client: SEARCH FROM "cs701@cse.iitb.ac.in"
Server: OK 12 16 17 19 24

Label "Swing"

Client: SEARCH LABEL "Swing"
Server: OK 11 16 19 25

- Messages that match - 16, 19

Per-user Sieve filters

- Client converts filter definition to the seive rule in RFC 3028 format.
- Sends it to the server.
- Server checks the syntax
- Server stores rule in the database
- SMTP server applies the filter on incoming messages

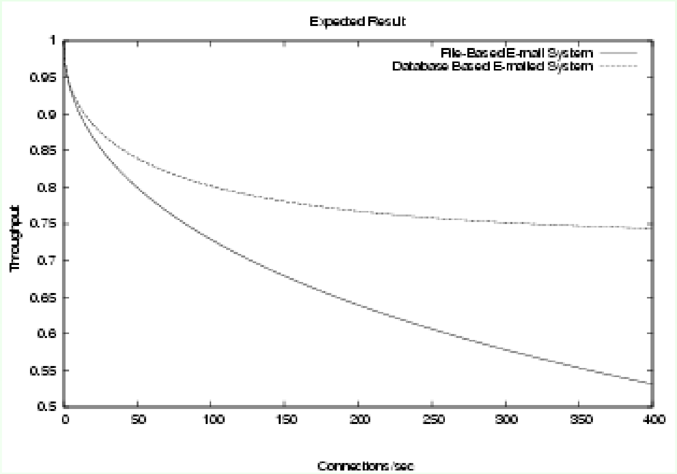
Per-user Sieve filters

- Client converts filter definition to the seive rule in RFC 3028 format.
- Sends it to the server.
- Server checks the syntax
- Server stores rule in the database
- SMTP server applies the filter on incoming messages

Per-user Sieve filters

- Client converts filter definition to the seive rule in RFC 3028 format.
- Sends it to the server.
- Server checks the syntax
- Server stores rule in the database
- SMTP server applies the filter on incoming messages

Performance graph



References

- http://www.ietf.org/rfc/rfc3501.txt
- http://www.ietf.org/rfc/rfc3028.txt
- http://www.openldap.org
- http://www.squirrelmail.org