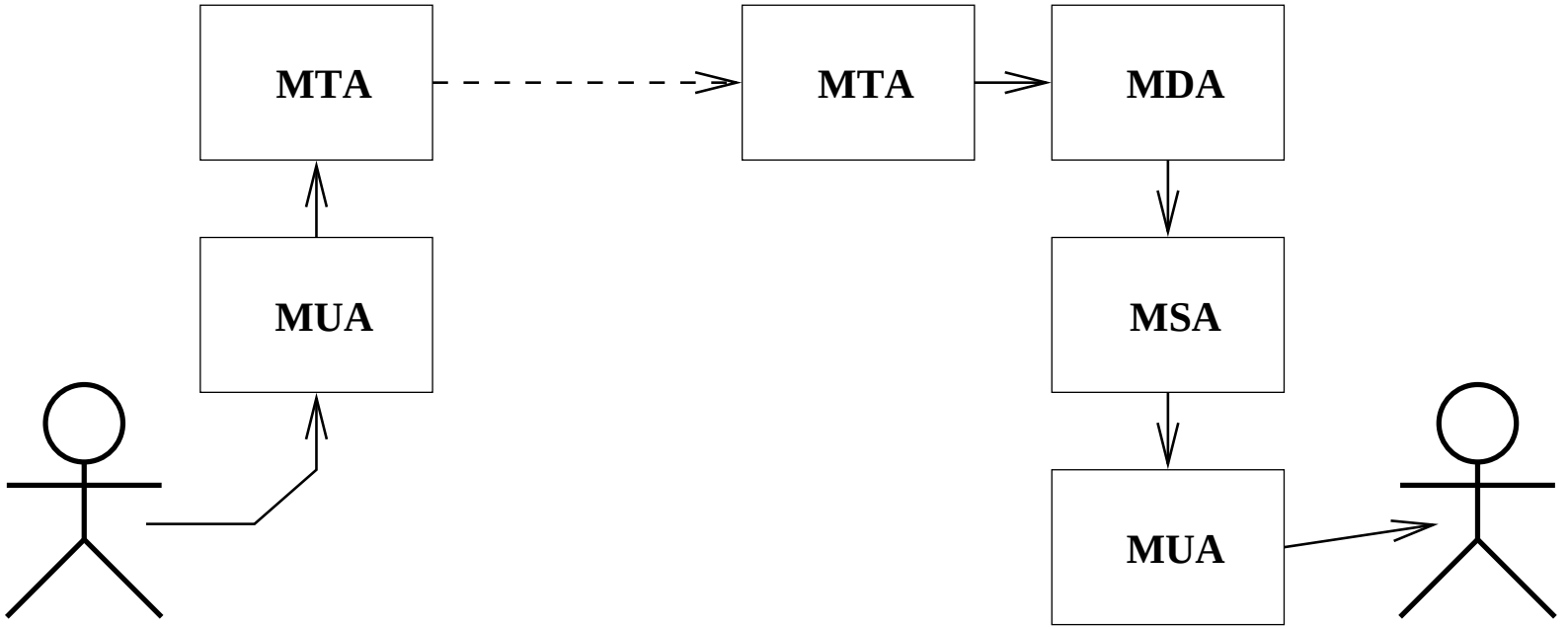
5.3 Mail Agenten



MUA Mail User Agent, Benutzerschnittstelle, Erstellen von Mails (mail, pine, mh, outlook express, nsmail, ...)

↓Kommunikation: Programmaufruf, SMTP

MTA Mail Transfer Agent, Routing von Mails (sendmail, qmail, postfix, ...)

↓Kommunikation: Programmaufruf, SMTP

MDA Mail Delivery Agent, Zustellung der Mail in die Mailbox (mail, deliver, qmail, procmail, ...)

↓Kommunikation: Programmaufruf, IMAP, Dateizugriff

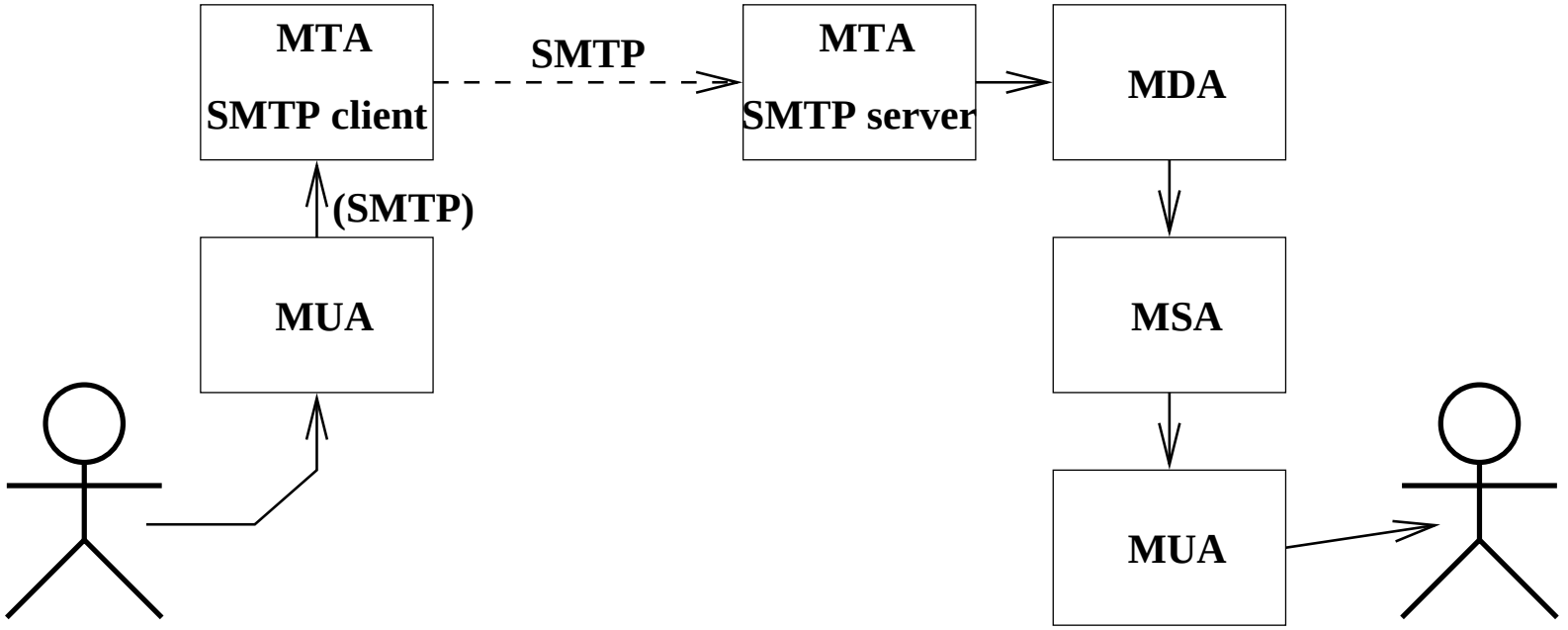
MSA Mail Storage Agent, "Mailspeicher" (imap)

↓Kommunikation: IMAP, POP, Dateizugriff

MUA Mail User Agent, Lesen von Mails

5.4 Simple Mail Transfer Protocol (SMTP)

- RFC 2821 (Revision)
- Stream Protokoll, port 25 “smtp”
- Mail Transport, Delivery, Submission



“Belauschte” Konversation

```
> telnet localhost smtp
220 kailua.informatik.uni-freiburg.de ESMTP Sendmail 8.12.2/8.12.2/SuSE Linux 0.6;
    Tue, 1 Jul 2003 14:24:18 -0700
HELO kailua
250 kailua.informatik.uni-freiburg.de Hello localhost [127.0.0.1], pleased to meet you
MAIL FROM: thiemann
250 2.1.0 thiemann... Sender ok
RCPT TO: president@whitehouse.gov
250 2.1.5 president@whitehouse.gov... Recipient ok
DATA
354 Enter mail, end with "." on a line by itself
Hello, just a test missile.
.
250 2.0.0 h61LOIlx008730 Message accepted for delivery
QUIT
221 2.0.0 kailua.informatik.uni-freiburg.de closing connection
Connection closed by foreign host.
```

Erweiterungen

- Ursprünglich: 7bit Transportprotokoll
- ESMTP definiert Rahmen für Erweiterungen
- Beispiel:

```
220 kailua.informatik.uni-freiburg.de ESMTP Sendmail 8.12.2/8.12.2/SuSE Linux 0.6;
    Tue, 1 Jul 2003 14:29:02 -0700
EHLO localhost
250-kailua.informatik.uni-freiburg.de Hello localhost [127.0.0.1], pleased to meet you
250-ENHANCEDSTATUSCODES
250-PIPELINING
250-8BITMIME
250-SIZE
250-DSN
250-ETRN
250-AUTH GSSAPI
250-DELIVERBY
250 HELP
```

Spezifikation der Erweiterungen

Parameter der verschickten Mail (Größe, 8bit?, Zeitraum)

`MAIL FROM:`⟨*reverse-path*⟩ `[SP` ⟨*mail-parameters*⟩ `]` ⟨*CRLF*⟩

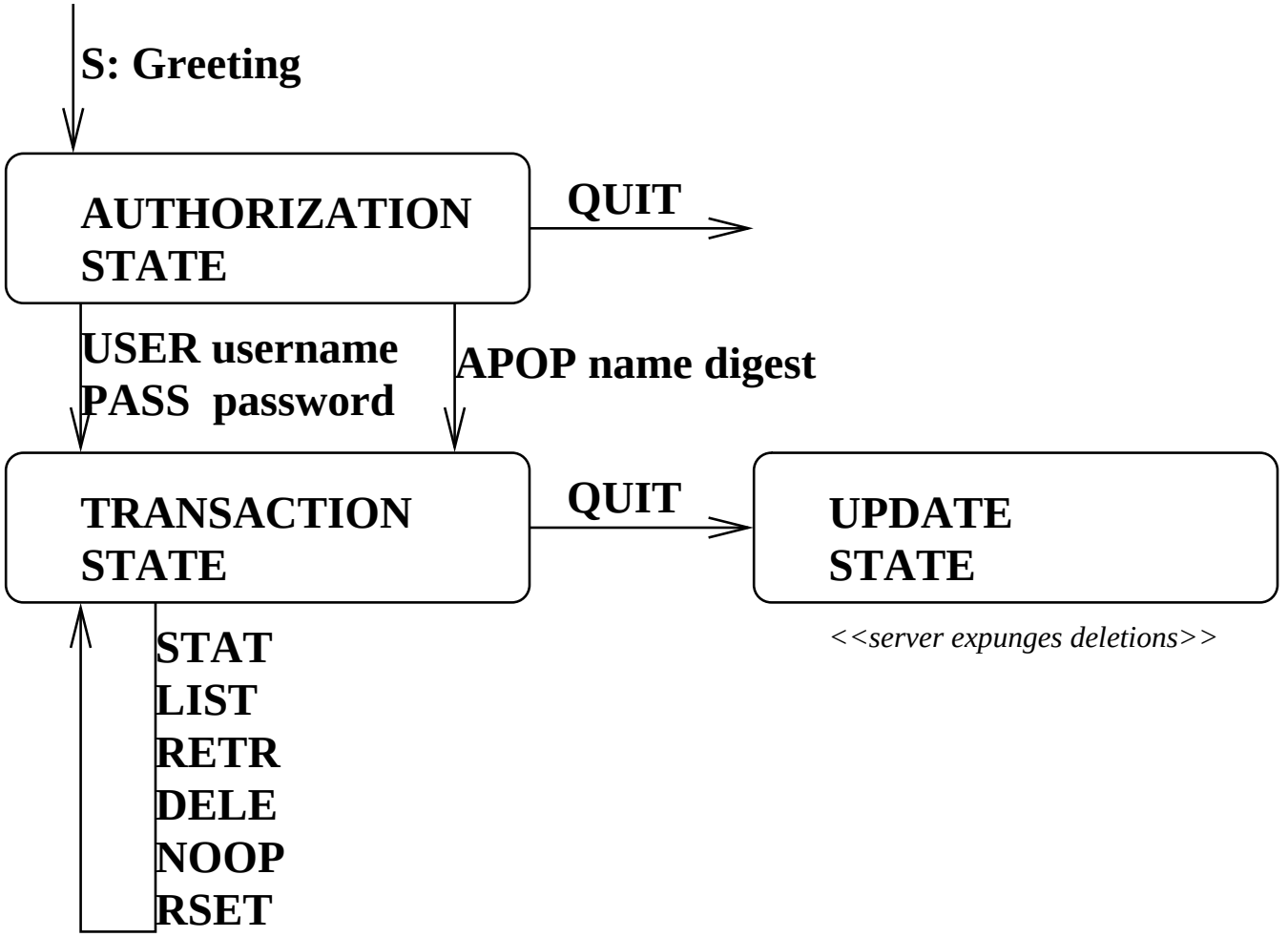
Empfängerparameter

`RCPT TO:`⟨*forward-path*⟩ `[SP` ⟨*rcpt-parameters*⟩ `]` ⟨*CRLF*⟩

5.5 Post Office Protocol (POP3)

- RFC 1939
- Mail **Abholung** für PC-artige Computer
  - nicht ständig eingeschaltet
  - nicht genug Ressourcen für MTS (message transfer system) d.h. MTA und MDA
- Zustellung auf zentralem Server
- geringer Funktionsumfang
- Stream Protokoll, Port 110 “pop3”

Übersicht POP3



Authentisierung

- **USER** ⟨*name*⟩, **PASS** ⟨*word*⟩
  überträgt Information im Klartext
- **APOP** ⟨*name*⟩ ⟨*digest*⟩
  berechnet digest aus Nonce (in der Begrüßung) und Geheimnis (Passwort)
  (nur möglich, falls Nonce vorhanden)

```
S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>
C: APOP mrose c4c9334bac560ecc979e58001b3e22fb
S: +OK maildrop has 1 message (369 octets)
```

- Nonce: `<1896.697170952@dbc.mtview.ca.us>`
- Geheimnis: `tanstaaf`
- Digest=MD5(Nonce+Geheimnis)
  =MD5(`<1896.697170952@dbc.mtview.ca.us>tanstaaf`)
  =c4c9334bac560ecc979e58001b3e22fb

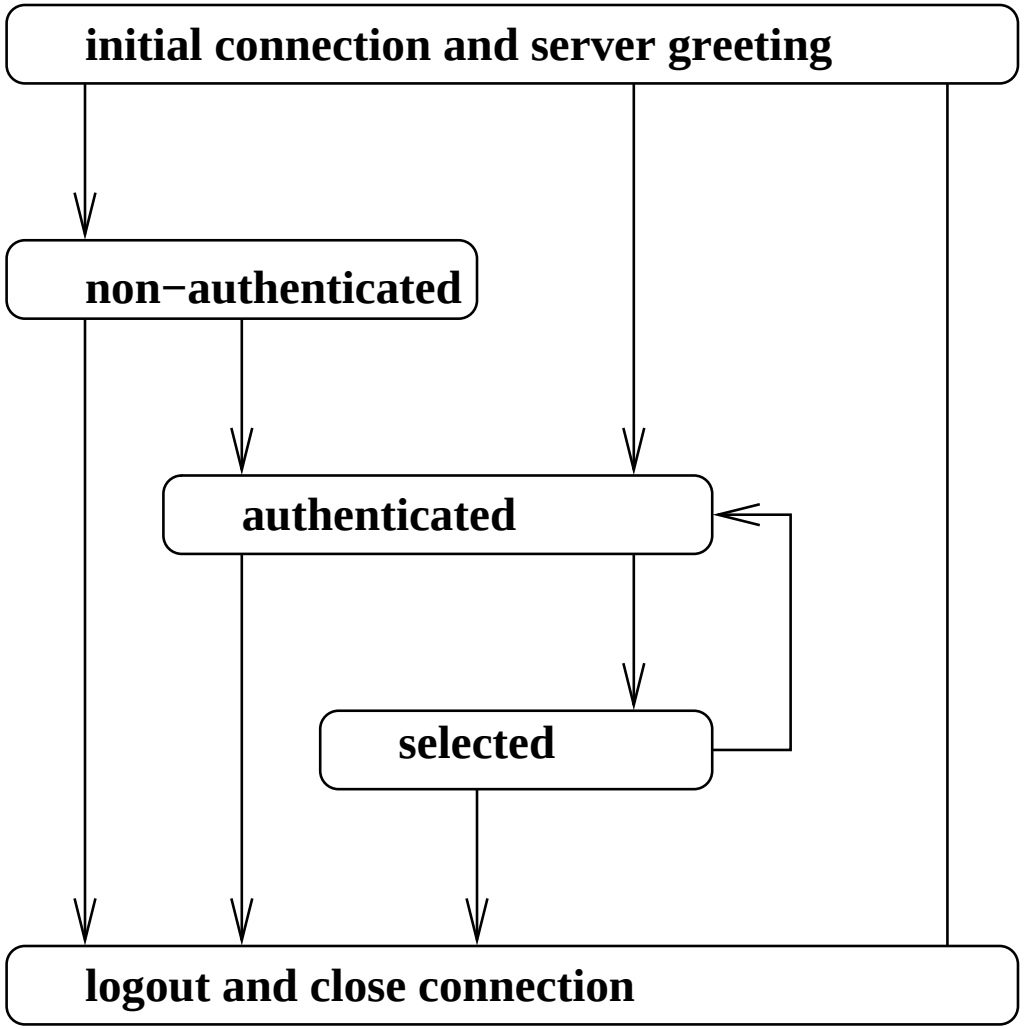
Diskussion

- Vorteile: einfach und klein, Grundfunktionalität
- Nachteile:
  - Mailgefängnis PC
  - Zusammenarbeit?
  - Datensicherheit?
  - keine Suche und Bearbeitung
  - keine Offline Verarbeitung

5.6 Internet Message Access Protocol (IMAP4)

- RFC 1730
- Zweck: "... access and manipulate electronic mail messages on a [centralized] server."
- "manipulation of remote message folders"
- "operations for creating, deleting, and renaming mailboxes; checking for new messages; permanently removing messages; setting and clearing flags; RFC 822 and MIME parsing; searching; and selective fetching of message attributes, texts, and portions thereof."

Übersicht IMAP4



Typischer Ablauf

- Authentisierung
  - `LOGIN` ⟨*Name*⟩ ⟨*Passwort*⟩
    (Klartext OK, falls über SSL/TLS oder direkte Einwahl)
  - `AUTHENTICATE ...`
- (authenticated) Kommandos auf Mailboxen:
  - Erzeugung, Umbenennung, Aktivieren, ...
  - Auswahl einer Mailbox → ...
- (selected) Kommandos auf Messages:
  Suche, Zugriff, Abspeichern, Kopieren

Zugriff auf Nachrichten

- sequentielle Numerierung innerhalb einer Mailbox
  kurzlebig, da ständige Änderung
- UID/UIDVALUE (je 32 Bit)
  - Mailbox identifiziert durch Name und UIDVALUE
  - UID = eindeutiger Bezeichner einer Nachricht *in einer Mailbox*
  - Vergabe der UIDs erfolgt aufsteigend, aber nicht notwendig zusammenhängend
  - Änderung des Mailboxnamens $\Rightarrow$ UIDs werden ungültig
- UID/UIDVALUE verwendet für Synchronisation zwischen Client und Server
- Probleme
  - Eindeutigkeit nur innerhalb einer Mailbox / eines UIDVALUE
  - Mehrere Server nicht vorgesehen

Beispielsitzung

```
S:   * OK IMAP4 Service Ready
C:   a001 login mrc secret
S:   a001 OK LOGIN completed
C:   a002 select inbox
S:   * 18 EXISTS
S:   * FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
S:   * 2 RECENT
S:   * OK [UNSEEN 17] Message 17 is the first unseen message
S:   * OK [UIDVALIDITY 3857529045] UIDs valid
S:   a002 OK [READ-WRITE] SELECT completed
C:   a003 fetch 12 full
S:   * 12 FETCH (FLAGS (\Seen) INTERNALDATE "14-Jul-1993 02:44:25 -0700"
      RFC822.SIZE 4282 ENVELOPE ("Wed, 14 Jul 1993 02:23:25 -0700 (PDT)"
      "IMAP4 WG mtg summary and minutes"
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      (("Terry Gray" NIL "gray" "cac.washington.edu"))
      ((NIL NIL "imap" "cac.washington.edu"))
      ((NIL NIL "minutes" "CNRI.Reston.VA.US")
      ("John Klensin" NIL "KLENSIN" "INFOODS.MIT.EDU")) NIL NIL
      "<B27397-0100000@cac.washington.edu>")
       BODY ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 3028 92))
S:    a003 OK FETCH completed
```

```
C:    a004 fetch 12 rfc822.header
S:    * 12 FETCH (RFC822.HEADER {346}
S:    Date: Wed, 14 Jul 1993 02:23:25 -0700 (PDT)
S:    From: Terry Gray <gray@cac.washington.edu>
S:    Subject: IMAP4 WG mtg summary and minutes
S:    To: imap@cac.washington.edu
S:    cc: minutes@CNRI.Reston.VA.US, John Klensin <KLENSIN@INFOODS.MIT.EDU>
S:    Message-Id: <B27397-0100000@cac.washington.edu>
S:    MIME-Version: 1.0
S:    Content-Type: TEXT/PLAIN; CHARSET=US-ASCII
S:
S:    )
S:    a004 OK FETCH completed
C:    a005 store 12 +flags \deleted
S:    * 12 FETCH (FLAGS (\Seen \Deleted))
S:    a005 OK +FLAGS completed
C:    a006 logout
S:    * BYE IMAP4 server terminating connection
S:    a006 OK LOGOUT completed
```


5.7 JavaMail

- Sun: The JavaMail API provides a platform-independent and protocol-independent framework to build mail and messaging applications.
- Teil von J2EE
- Packages `javax.mail`, `javax.mail.internet`
- Funktionen
  - Mail versenden
  - Mail empfangen (IMAP, POP3)
  - Mail verarbeiten (multi part)
  - Mail durchsuchen (IMAP)

Hauptklassen

Session Sitzung mit ein- und ausgehendem Mailserver

Message abstrakt: `javax.mail.internet.MimeMessage`

Address Sender- und Empfängeradressen

Authenticator Loginmanagement für Mailserver

Transport Versenden einer Nachricht (SMTP oder SMTPS)

Store IMAP oder POP Server (Empfang)

Folder (Eingangs-) Ordner

Beispiel A: Mail senden (Rahmen)

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

class SendJavaMail {
    public static void main (String[] arg) {
        String outgoing = "smtp.informatik.uni-freiburg.de";
        String from = "president@whitehouse.gov";
        String to = "info@deep-purple.com";
        try {
            // A1: obtain session
            // A2: create message & attach addresses
            // A3: send message
        } catch (Exception e) {
        }
    }
}
```

Beispiel/A1: Session

```
// obtain a session for the outgoing SMTP server
Properties props = System.getProperties();
        // props = new Properties();
props.put ("mail.smtp.host", outgoing);
Session session = Session.getDefaultInstance (props);
```

Beispiel/A2: Message, Address

```
// create a new message object
MimeMessage msg = new MimeMessage (session);
// headers and contents are properties
msg.setText("Nobody overtook my car...");
msg.setSubject ("Highway Star");

// must attach addresses
Address fromaddr = new InternetAddress (from, "Mr. President");
msg.setFrom (fromaddr);

Address toaddr = new InternetAddress (from, "Jon Lord");
msg.addRecipient (Message.RecipientType.TO, toaddr);
```

Beispiel/A3: Transport

```
// send it off via SMTP
Transport transport = session.getTransport("smtp");
transport.connect();
transport.sendMessage (msg, msg.getAllRecipients());

transport.close();
```

Beispiel B: Mail lesen (Rahmen)

```
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;

class ReadJavaMail {
    public static void main (String[] arg) {
        String incoming = arg[0];
        String username = arg[1];
        String password = arg[2];

        try {
            // B1: obtain session, store, folder, and messages
            // B2: application code
            // B3: cleanup
        } catch (Exception e) {
        }
    }
}
```

Beispiel/B1: Session, Store, Folder, Message

```
// obtain a session
Properties props = new Properties();
Session session = Session.getDefaultInstance (props);

// get a store
Store store = session.getStore ("pop3");
store.connect (incoming, username, password);

// get INBOX folder and messages
Folder folder = store.getFolder ("INBOX");
folder.open (Folder.READ_ONLY);
Message[] msgs = folder.getMessages();
```

Beispiel/B2: Anwendungskode

```
System.out.println ("Statistics for INBOX@" + incoming);
System.out.println (msgs.length + " messages");
for (int i=0; i<msgs.length; i++) {
    System.out.println ("Message " + i + ":");
    System.out.println (((MimeMessage)msgs[i]).getContent());
    System.out.println ("-----------");
}
```

Beispiel/B3: Aufräumen

```
folder.close (false); // keep all messages
store.close ();
```

5.7.1 Authenticator

- Realistische Eingabe von Name/Passwort
- `Authenticator`-Objekt mit Methode `getPasswordAuthentication`
- abgelegt in `Session`-Objekt
- Authentisierung wird automatisch gestartet, wenn notwendig

Einbindung des Authenticator

```
// Setze Rechnername für POP3 Dienst
Properties props = System.getProperties();
props.put("mail.pop3.host", host);

// Definiere Authentisierung für neue Session
Authenticator auth = new MyAuthenticator();
Session session = Session.getDefaultInstance(props, auth);

Store store = session.getStore("pop3");
store.connect();
// fragt nach Name/Passwort via MyAuthenticator
```

MyAuthenticator

```
import java.io.*;
import java.util.*;
import javax.mail.*;

class MyAuthenticator extends Authenticator {
    public PasswordAuthentication getPasswordAuthentication() {
        String username = null, password = null;
    BufferedReader reader = new BufferedReader (new InputStreamReader(System.in));
    try {
        System.out.print ("Username: ");
        username = reader.readLine();
        System.out.print ("Password: ");
        password = reader.readLine();
    } catch (IOException e) {
    }
    return new PasswordAuthentication(username, password);
  }
}
```

5.7.2 Multipart Nachrichten

- Erzeugen von Attachments, alternativen Teilen, etc
- Teile spezifiziert mit JAF (Java Activation Framework)

  `DataSource` Interface für Daten mit MIME Typ, können von/auf Stream gelesen/geschrieben werden `FileDataSource`, `URLDataSource`

  `DataHandler` Manipulation von Daten in mehreren unterschiedlichen Formaten; Stream $\Rightarrow$ String Umwandlung

Beispiel/C1: Multipart Vorbereitung

```
// SendFile.java
// keine Neuigkeit --- nur der Vollständigkeit halber

// obtain a session for the outgoing SMTP server
Properties props = new Properties();
props.put ("mail.host", outgoing);
Session session = Session.getDefaultInstance (props);

// create a new message object
MimeMessage msg = new MimeMessage (session);
msg.setSubject ("File: " + filename);
msg.setFrom (new InternetAddress (from));
msg.addRecipient (Message.RecipientType.TO, new InternetAddress (to));
```

Beispiel/C2: Zusammenbau des Multipart

```
Multipart mp = new MimeMultipart ();

// Part 1: some text
BodyPart bp = new MimeBodyPart ();
bp.setText("Please find enclosed the file.");
mp.addBodyPart (bp);

// Part 2: the file content
bp = new MimeBodyPart ();
DataSource src = new FileDataSource (filename);
bp.setDataHandler (new DataHandler (src));
bp.setFileName (filename);
mp.addBodyPart (bp);

msg.setContent (mp);
```

Beispiel/C3: Absenden mit Kontrollnachricht

```
// send it off via SMTP
Transport transport = session.getTransport("smtp");
transport.addTransportListener (new SendFileTL ());
transport.connect();
transport.sendMessage (msg, msg.getAllRecipients());

transport.close();
```

Beispiel/C4: Erzeugen der Kontrollnachricht

```
import javax.mail.event.*;

class SendFileTL extends TransportAdapter {
    public void messageDelivered (TransportEvent e) {
System.out.println ("Message delivered.");
    }
    public void messageNotDelivered (TransportEvent e) {
System.out.println ("Message not delivered.");
    }
    public void messagePartiallyDelivered (TransportEvent e) {
System.out.println ("Message partially delivered.");
    }
}
```