Digitale Kommunikation und Internetdienste 1

Wintersemester 2004/2005 – Teil 13

Belegnummer Vorlesung: 39 30 02

Übungen: 39 30 05

Jan E. Hennig

AG Rechnernetze und Verteilte Systeme (RVS)
Technische Fakultät
Universität Bielefeld

`jhennig@rvs.uni-bielefeld.de`

basierend auf den Arbeiten von Michael Blume, Heiko Holtkamp, Marcel Holtmann und I Made Wiryana

Fahrplan

- 13.01.2005: Vorlesung 13
- 13./14.01.2005: Übung 6
- 20.01.2005: Klausur, Übungszettel 7
- 27.01.2005: Vorlesung 14
- 27./28.01.2005: Übung 7
- 03.02.2005: Vorlesung 15

Klausur (1)

- die Klausur findet am 20.01.2005 von 12s.t. bis 14 Uhr in H11 statt:
  - es wird einige Fragen geben, die ähnlich der der Übungszettel aufgebaut sind
  - diese umfassen die Themengebiete aus Vorlesung und Übungen
  - es wird eine besonders gekennzeichnete zusätzliche Frage auf der Klausur angeboten
  - wenn passend beantwortet zeigt diese, daß man zusätzliche 15 Stunden über die Veranstaltung hinaus dem Thema gewidmet hat
  - diese 15h bedeuten 0,5 LP und damit könnte man als MGSler 5 LP statt 4,5 LP für diese Veranstaltung bekommen
  - für MIGler gibt es gesonderte (benotete) Klausuraufgaben

Klausur (2)

- erlaubtes Material:
  - leere Seiten Papier (für eigene Entwürfe, Skizzen, . . .)
  - dokumentenechter Stift
  - nichtprogrammierbarer Taschenrechner

Übersicht

- IMAP
- FTP

Internet Message Access Protocol (1)

Datum	Titel	Nummer	Status
1988, Jul	Interactive Mail Access Protocol: Version 2	RFC 1064	Unknown
1990, Aug	Interactive Mail Access Protocol: Version 2	RFC 1176	Experimental
1991, Feb	Interactive Mail Access Protocol: Version 3	RFC 1203	Historic
1994, Dez	Internet Message Access Protocol - Version 4	RFC 1730	Proposed Standard
1994, Dez	IMAP4 Authentication Mechanisms	RFC 1731	Proposed Standard
1994, Dez	IMAP4 Compatibility with IMAP2 and IMAP2bis	RFC 1732	Informational
1994, Dez	Distributed Electronic Mail Models in IMAP4	RFC 1733	Informational
1996, Dez	Internet Message Access Protocol - Version 4rev1	RFC 2060	Proposed Standard
1996, Dez	IMAP4 Compatibility with IMAP2bis	RFC 2061	Informational
1996, Dez	Internet Message Access Protocol - Obsolete Syntax	RFC 2062	Informational
1997, Jan	IMAP4 ACL extension	RFC 2086	Proposed Standard
1997, Jan	IMAP4 QUOTA extension	RFC 2087	Proposed Standard
1997, Jan	IMAP4 non-synchronizing literals	RFC 2088	Proposed Standard

Internet Message Access Protocol (2)

Datum	Titel	Nummer	Status
1997, Jan	IMAP/POP AUTHorize Extension for Simple Challenge/Response	RFC 2095	Proposed Standard
1997, Jun	IMAP4 IDLE command	RFC 2177	Proposed Standard
1997, Jul	IMAP4 Multi-Accessed Mailbox Practice	RFC 2180	Informational
1997, Sep	IMAP URL Scheme	RFC 2192	Proposed Standard
1997, Sep	IMAP4 Mailbox Referrals	RFC 2193	Proposed Standard
1997, Sep	IMAP/POP AUTHorize Extension for Simple Challenge/Response	RFC 2195	Proposed Standard
1997, Okt	IMAP4 Login Referrals	RFC 2221	Proposed Standard
1997, Nov	ACAP -- Application Configuration Access Protocol	RFC 2244	Proposed Standard
1998, Mai	IMAP4 Namespace	RFC 2342	Proposed Standard
1998, Jun	IMAP4 UIDPLUS extension	RFC 2359	Proposed Standard
1999, Jun	Using TLS with IMAP, POP3 and ACAP	RFC 2595	Proposed Standard
1999, Sep	IMAP4 Implementation Recommendations	RFC 2683	Informational
2000, Okt	IMAP4 ID extension	RFC 2971	Proposed Standard

Distributed Electronic Mail Models

- es gibt drei unterschiedliche E-Mail-Modelle:
  - offline
  - online
  - disconnected
- definiert in RFC 1733
- IMAP4 unterstützt alle drei Modelle
- POP3 ist ein Offline-Modell (download–delete)

Vor- und Nachteile der Modelle

Feature	Offline	Online	Disconnected
Can use multiple clients	✗	✓	✓
Minimum use of server connect time	✓	✗	✓
Minimum use of server resources	✓	✗	✗
Minimum use of client disk resources	✗	✓	✗
Multiple remote mailboxes	✗	✓	✓
Fast startup	✗	✓	✗
Mail processing when not online	✓	✗	✓

Ideen von IMAP

- Protokoll zum Zugriff und Bearbeiten von Nachrichten auf dem Mail Server
- Unterstützung von Ordnern auf dem Server
- ein Ordner wird *Mailbox* genannt
- Möglichkeit der Re-Synchronisation
- jede Nachricht hat eine eindeutige Nummer auf dem IMAP Server (UID bzw. MSN)

Weitere Vorteile von IMAP

- der Server hat einen *Autologout-Timer*
- es können neue Befehle abgesendet werden, obwohl andere noch in Bearbeitung sind
- ein Server kann auch von sich aus Antworten an den Client übertragen
- das Herunterladen eines *Attachments* geschieht erst, wenn es benötigt wird

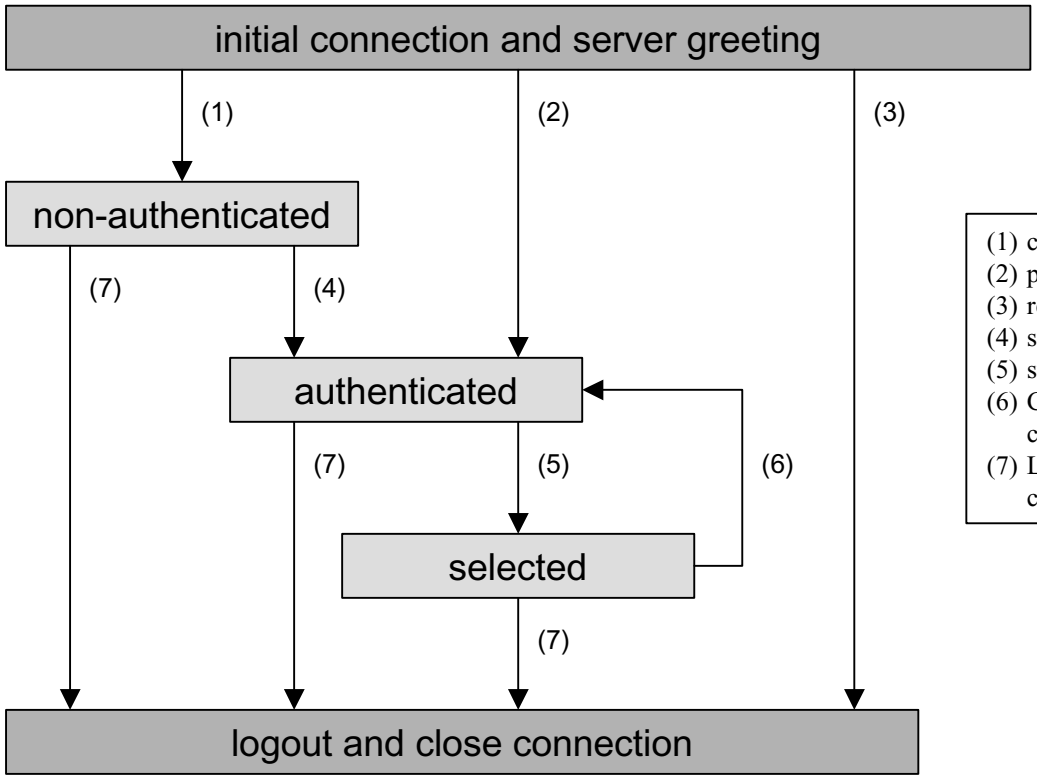
Operationen von IMAP

- Erstellen, Löschen und Umbenennen von *Mailboxen*
- Kontrolle ob neue Nachrichten vorhanden sind
- endgültiges Löschen von Nachrichten
- Setzen und Entfernen von *Flags*
- Auswerten von Nachrichten im RFC822- oder MIME-Format für mehr Kontrolle und Informationen
- selektives und partielles Herunterladen von Nachrichten oder auch Header-Einträgen
- Durchsuchen des Servers

Was kann IMAP nicht?

- es wird kein Mechanismus angegeben um auf Konfigurationen für mehrere IMAP-Server zuzugreifen
- mit IMAP kann man keine Nachrichten versenden
  (Hierfür ist das SMTP-Protokoll zuständig)
- IMAP4 ist nicht kompatibel zu IMAP3
  (Kompatibilität ist zu IMAP2 und IMAP2bis spezifiziert)

Vier-Status-Protokoll IMAP



(1) connection without pre-authentication (OK greeting)
(2) pre-authenticated connection (PREAUTH greeting)
(3) rejected connection (BYE greeting)
(4) successful LOGIN or AUTHENTICATE command
(5) successful SELECT or EXAMINE command
(6) CLOSE command, or failed SELECT or EXAMINE command
(7) LOGOUT command, server shutdown, or connection closed

Kommandos und Responses

- jedem Kommando wird ein ASCII-String (ein *Tag*) vorangestellt (z.B. `A001, A002, ...`)
- die Antwort enthält auch den entsprechenden Tag gefolgt von
  - einem Server-Status,
  - einem optionalen Response-Code und
  - einem Text zur Weitergabe an den MUA
- Ein *-Tag kennzeichnet eine *untagged response*
- Mit dem Tag + signalisiert der Server, daß weitere Eingaben für das Kommando notwendig sind

Ein Beispiel

*	OK IMAPrev1 Service ready
A001	LOGIN "holtmann" "sagichnicht"
A001	OK LOGIN completed
A002	SELECT "INBOX"
*	18 EXISTS
*	FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
*	2 RECENT
*	OK [UNSEEN 17] Message 17 is the first unseen message
*	OK [UIDVALIDITY 3857529045] UIDs valid
A002	OK [READ-WRITE] SELECT completed
A003	LOGOUT
*	BYE IMAP4rev1 server terminating connection
A003	OK LOGOUT completed

Server Status-Responses

Server Status Response	Tagged Meaning	Untagged Meaning
OK	Erfolgreiche Beendigung eines Befehls	Zusätzliche Informationen
NO	Der Befehl wurde nicht erfolgreich ausgeführt oder abgeschlossen	Ausgabe einer Warnung
BAD	Ein Protokollfehler ist aufgetreten	Ein Protokollfehler auf dem Server dem kein Kommando zugeordnet werden kann
PREAUTH	N/A	Wird beim Start einer Verbindung benutzt um anzuzeigen, das eine Verbindung mit einem externen Mechanismus authentisiert wurde
BYE	N/A	Der Server wird die Verbindung schließen

Response-Code

Response Code	Beschreibung
ALERT	Nachricht in menschenlesbarer Form
NEWNAME	Meldung beim Umbenennen einer Mailbox
PARSE	Es wird angezeigt das ein Header nicht untersucht werden kann
PERMANENTFLAGS	Liste der Flags, die ein Client dauerhaft ändern kann
READ-ONLY	Eine Mailbox wurde nur zum lesen geöffnet
READ-WRITE	Die Mailbox wurde für Lese- und Schreibzugriffe geöffnet
TRYCREATE	Die Zielmailbox existiert nicht – kann aber erstellt werden
UIDVALIDITY	Der „UID Validity“ Wert
UNSEEN	Die Message Nummer der ersten ungelesenen Nachricht

Der Response Code wird in eckigen Klammern hinter dem Status angegeben (z.B. [UNSEEN 51])

Additional Untagged Responses

Status Response	Beschreibung
FLAGS	Auflistung der Flags, die für eine Mailbox definiert sind
<n> EXISTS	Anzahl der Nachrichten in einer Mailbox
<n> RECENT	Die Anzahl der Nachrichten in einer Mailbox mit dem „\Recent“ Flag
<n> EXPUNGE	Es wurde die Nachricht mir der Nummer „<n>“ vom System entfernt
<n> FETCH	Die Nachricht (oder Teile) werden ausgegeben

Command Response	Beschreibung
LIST / LSUB	Ausgabe von Mailboxlisten (pro Mailbox eine Response)
SEARCH	Durch Leerzeichen getrennte Liste von Message Nummern
STATUS	Rückgabe des Status einer Mailbox

Message-Nummern

- es gibt zwei Arten von Message-Nummern:
- UID: Unique Identifier (32-Bit-Wert)
- MSN: Message Sequence Number
- eine UID darf während einer Sitzung nicht verändert werden und sollte sich nie ändern
- die MSN ist eine Nummerierung von $1 \ldots n$, die für jede Mailbox individuell ist

Message-Flags

\Seen	Die Nachricht wurden gelesen
\Answered	Die Nachricht wurde beantwortet
\Flagged	Die Nachricht ist „markiert“ für eine notwendige oder spezielle Aufmerksamkeit
\Deleted	Die Nachricht ist „gelöscht“ und wird bei einem „EXPUNGE“ entgültig entfernt
\Draft	Die Nachricht wurde nicht vollständig fertiggestellt und wird als „Vorabversion“ markiert
\Recent	Diese Nachricht ist gerade angekommen Bei einer weiteren Session wird das Flag nicht mehr vorhanden sein

Datenformate

atom	Ein oder mehrere Standardzeichen
number	Ein oder mehrere Ziffern die einen numerischen Eintrag bilden
string	Eine „literal“ oder „quoted“ Zeichenfolge z.B. {8}holtmann und "holtmann" Binäre Strings müssen MIME benutzen
parenthesized list	Eine Liste von Einträge, die durch Leerzeichen getrennt und mit runden Klammern gebunden ist
NIL	Ein nicht Vorhandensein von einem Dateneintrag

Message-Sets

Ein Message-Set ist eine Menge von Message-Nummern, die nicht leer sein darf

- es dürfen einzelne Nummern angegeben werden
- es darf eine willkürliche Liste von Nummern angegeben werden, die durch Kommata abgetrennt sind
- ein Bereich von Nummern wird durch die Trennung mit einem Doppelpunkt angegeben (z.B. 1:5)
- mehrere Bereiche werden durch Kommata getrennt
- Der * wird als Platzhalter benutzt (z.B. 1:*)

Mailbox-Namen

- die Mailbox *INBOX* (case-insensitive) definiert die primäre Mailbox für den aktuellen Benutzer
- es wird eine *left-to-right-Hierarchie* benutzt
- es gibt ein Zeichen, das durchgehend als *hierarchy separator* benutzt wird (z.B. /)
- Ein # am Anfang einer Mailbox definiert einen Namensraum (z.B. `#news.comp.mail.misc`)

Namensräume

- Es wird zwischen drei elementaren Klassen von Namenräumen unterschieden
  - personal namespaces (z.B. INBOX.)
  - other users' namespaces (oder user.)
  - shared namespaces (#news. oder #shared/)
- in jeder Klasse kann es mehrere Namensräume geben
- jeder Namensraum kann einen unterschiedlichen Hierarchy-Separator benutzen

Kommandos in jedem Status

CAPABILITY	Es wird eine Liste von Fähigkeiten angefordert. Die Rückgabe ist in jedem Status identisch.
NOOP	Der Befehl setzt den „auto-logout“ Zählers zurück und kann den „message status“ liefern. Hierdurch kann man nach neuen Nachrichten auf dem Server Ausschau halten.
LOGOUT	Der Server wechselt in den Logout Status und beendet die Verbindung.

```
abcd CAPABILITY
* CAPABILITY IMAP4rev1 AUTH=KERBEROS_V4 AUTH=CRAM-MD5
abcd OK CAPABILITY completed
```

```
a002 NOOP
a002 OK NOOP completed

a047 NOOP
* 22 EXPUNGE
* 23 EXISTS
* 3 RECENT
* 14 FETCH (FLAGS (\Seen \Deleted))
a047 OK NOOP completed
```

```
A023 LOGOUT
* BYE IMAP4rev1 Server logging out
A023 OK LOGOUT completed
(Server and client then close the connection)
```

Kommandos in Non-Authenticated

AUTHENTICATE	Es wird eine Authentifizierung über eine angegebenen Mechanismus durchgeführt. Der Server fordert weitere Eingaben mit dem „+“ Tag. Ein Abbruch erfolgt durch senden eines „*“.
LOGIN	Eine einfache Anmeldung mit Passwort im Klartext.

```
A001 AUTHENTICATE KERBEROS_V4
+ AmFYig==
BAcAQU5EUkVXLkNNVS5FRFUAOCAsho84kLN3/IJmrMG+25a4DT+nZI
mJjnTNHJUtxAA+oOKPKfHEcAFs9a3CL5Oebe/ydHJUwYFdWwuQ1M
Wiy6lesKvjL5rL9WjXUb9MwT9bpObYLGOKi1Qh
+ or//EoAADZI=
DiAF5A4gA+oOIALuBkAAmw==
A001 OK Kerberos V4 authentication successful
```

```
a001 LOGIN smith sesame
a001 OK LOGIN completed
```

Weiterhin sind die Kommandos CAPABILITY, NOOP und LOGOUT gültig

Kommandos in Authenticated (1)

SELECT	Es wird eine Mailbox ausgewählt und bei Erfolg wechselt der Server in den „Selected Status“. Als Antwort wird eine Übersicht über die Mailbox ausgegeben.
EXAMINE	Die Mailbox wird ausgewählt, aber als „read-only“ identifiziert. So kann eine Mailbox untersucht werden, ohne den Status von Nachrichten zu verändern.
CREATE	Eine neue Mailbox wird angelegt. Für das Anlegen muss der passende „hierarchy separator“ benutzt werden.

```
A142 SELECT INBOX
* 172 EXISTS
* 1 RECENT
* OK [UNSEEN 12] Message 12 is first unseen
* OK [UIDVALIDITY 3857529045] UIDs valid
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS (\Deleted \Seen \*)] Limited
A142 OK [READ-WRITE] SELECT completed
```

```
A932 EXAMINE blurdybloop
* 17 EXISTS
* 2 RECENT
* OK [UNSEEN 8] Message 8 is first unseen
* OK [UIDVALIDITY 3857529045] UIDs valid
* FLAGS (\Answered \Flagged \Deleted \Seen \Draft)
* OK [PERMANENTFLAGS ()] No permanent flags permitted
A932 OK [READ-ONLY] EXAMINE completed
```

```
A003 CREATE owatagusiam/
A003 OK CREATE completed
A004 CREATE owatagusiam/blurdybloop
A004 OK CREATE completed
```

Kommandos in Authenticated (2)

LIST	Es wird eine Liste mit von Mailboxen zurückgeliefert. Um das Suchergebnis zu verfeinern kann man einen Referenzordner und ein Pattern angeben. Die Suche nach einer "" Pattern liefert den „hierarchy separator“ zurück. Wird "" als Referenz benutzt so gilt die Basisebene.
LSUB	Die Liste der „subscribed“ oder „active“ Mailboxen wird zurückgegeben.
SUBSCRIBE	Mailbox zur Liste der „subscribed“ hinzufügen.
UNSUBSCRIBE	Auswahl der Mailbox rückgängig machen.

```
A101 LIST "" ""
* LIST (\Noselect) "/" ""
A101 OK LIST Completed
A102 LIST #news.comp.mail.misc ""
* LIST (\Noselect) "." #news.
A102 OK LIST Completed
A103 LIST /usr/staff/jones ""
* LIST (\Noselect) "/" /
A103 OK LIST Completed
A202 LIST ~/Mail/ %
* LIST (\Noselect) "/" ~/Mail/foo
* LIST () "/" ~/Mail/meetings
A202 OK LIST completed
```

```
A002 LSUB "#news." "comp.mail.*"
* LSUB () "." #news.comp.mail.mime
* LSUB () "." #news.comp.mail.misc
A002 OK LSUB completed
```

```
A002 SUBSCRIBE #news.comp.mail.mime
A002 OK SUBSCRIBE completed
```

```
A002 UNSUBSCRIBE #news.comp.mail.mime
A002 OK UNSUBSCRIBE completed
```

Kommandos in Authenticated (3)

DELETE	Eine angegebene Mailbox wird entgültig vom System entfernt. Das rekursive entfernen eine Mailbox mit Unter-Mailboxen ist nicht definiert und muss vom Server auch nicht unterstützt werden.
RENAME	Umbenennen einer Mailbox.

```
A682 LIST "" *
* LIST () "/" blurdybloop
* LIST (\Noselect) "/" foo
* LIST () "/" foo/bar
A682 OK LIST completed
A683 DELETE blurdybloop
A683 OK DELETE completed
A684 DELETE foo
A684 NO Name "foo" has inferior hierarchical names
A685 DELETE foo/bar
A685 OK DELETE Completed
A686 LIST "" *
* LIST (\Noselect) "/" foo
A686 OK LIST completed
A687 DELETE foo
A687 OK DELETE Completed
```

```
A682 LIST "" *
* LIST () "/" blurdybloop
* LIST (\Noselect) "/" foo
* LIST () "/" foo/bar
A682 OK LIST completed
A683 RENAME blurdybloop sarasoop
A683 OK RENAME completed
A684 RENAME foo zowie
A684 OK RENAME Completed
A685 LIST "" *
* LIST () "/" sarasoop
* LIST (\Noselect) "/" zowie
* LIST () "/" zowie/bar
A685 OK LIST completed
```

Kommandos in Authenticated (4)

STATUS	Es wird der Status einer Mailbox abgefragt.
APPEND	Es wird eine Nachricht zu einer Mailbox hinzugefügt.

```
A042 STATUS blurdybloop (UIDNEXT MESSAGES)
* STATUS blurdybloop (MESSAGES 231 UIDNEXT 44292)
A042 OK STATUS completed
```

```
A003 APPEND saved-messages (\Seen) {310}
Date: Mon, 7 Feb 1994 21:52:25 -0800 (PST)
From: Fred Foobar <foobar@Blurdybloop.COM>
Subject: afternoon meeting
To: mooch@owatagu.siam.edu
Message-Id: <B27397-0100000@Blurdybloop.COM>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; CHARSET=US-ASCII

Hello Joe, do you think we can meet at 3:30 tomorrow?

A003 OK APPEND completed
```

Weiterhin sind die Kommandos CAPABILITY, NOOP und LOGOUT gültig

Kommandos in Selected (1)

CHECK	Es wird ein Checkpoint in der aktuellen Mailbox gesetzt. Das Kommando hat keine weitere Bedeutung und ist mit dem „NOOP“ Kommando identisch.
CLOSE	Die aktuelle Mailbox wird geschlossen und der Server wechselt in den „Authenticated Status“. Alle Nachrichten mit dem Flag „\Deleted“ werden endgültig entfernt.
EXPUNGE	Es werden alle Nachrichten mit dem Flag „\Deleted“ werden vom Server entfernt. Im Beispiel haben die Nachrichten 3, 4, 7 und 11 das „\Deleted“ Flag.

```
FXXZ CHECK
FXXZ OK CHECK Completed
```

```
A341 CLOSE
A341 OK CLOSE Completed
```

```
A202 EXPUNGE
* 3 EXPUNGE
* 3 EXPUNGE
* 5 EXPUNGE
* 8 EXPUNGE
A202 OK EXPUNGE completed
```

Kommandos in Selected (2)

COPY	Es wird ein Set von Nachrichten in eine andere Mailbox kopiert.
STORE	Einer Menge von Nachrichten werden Flags hinzugefügt („+“) oder wieder entfernt („-“). Als Ergebnis werden alle modifizierten Nachrichten gelistet.
SEARCH	Es wird eine Mailbox nach einer Liste von Kriterien durchsucht. Als Ergebnis liefert das Kommando eine Liste von Message Nummern zurück.

```
A003 COPY 2:4 MEETING
A003 OK COPY completed
```

```
A003 STORE 2:4 +FLAGS (\Deleted)
* 2 FETCH FLAGS (\Deleted \Seen)
* 3 FETCH FLAGS (\Deleted)
* 4 FETCH FLAGS (\Deleted \Flagged \Seen)
A003 OK STORE completed
```

```
A282 SEARCH FLAGGED SINCE 1-Feb-1994 NOT FROM "Smith"
* SEARCH 2 84 882
A282 OK SEARCH completed
```

Kommandos in Selected (3)

FETCH	Es wird eine Nachricht vom Server geladen. Hierbei kann speziell auswählen welchen Teil der Nachricht man bekommen möchte.
UID	Bei den Kommandos „COPY“, „FETCH“ und „STORE“ wird anstelle von Message Nummern eine UID angegeben. Bei „SEARCH“ im Gegensatz erfolgt die Ausgabe als UID‘s

```
A654 FETCH 2:4 (FLAGS BODY[HEADER.FIELDS (DATE FROM)])
* 2 FETCH ....
* 3 FETCH ....
* 4 FETCH ....
A654 OK FETCH completed
```

```
A999 UID FETCH 4827313:4828442 FLAGS
* 23 FETCH (FLAGS (\Seen) UID 4827313)
* 24 FETCH (FLAGS (\Seen) UID 4827943)
* 25 FETCH (FLAGS (\Seen) UID 4828442)
A999 UID FETCH completed
```

Weiterhin sind die Kommandos CAPABILITY, NOOP, LOGOUT und SELECT, EXAMINE, CREATE, DELETE, RENAME, SUBSCRIBE, UNSUBSCRIBE, LIST, LSUB, STATUS, APPEND gültig

Experimentelle Kommandos

- experimentelle Kommandos werden mit einem vorangestellten `X<atom>` gekennzeichnet
- der CAPABILITY-Befehl muß das experimentelle Kommando auflisten
- jeder Antwort vom Server muß das X auch vorangestellt sein
- Beispiel: `X-NETSCAPE`

Das Kommando Fetch

- es werden zwei Parameter mit angegeben:
  - ein Set von Nachrichten
  - eine Liste von zu holenden Informationen
- das Resultat sind vom Server geparste Einträge bzw. Teile der Nachricht
  - Flags, Envelope, Body und Struktur der Nachricht
  - internes Datum und UID

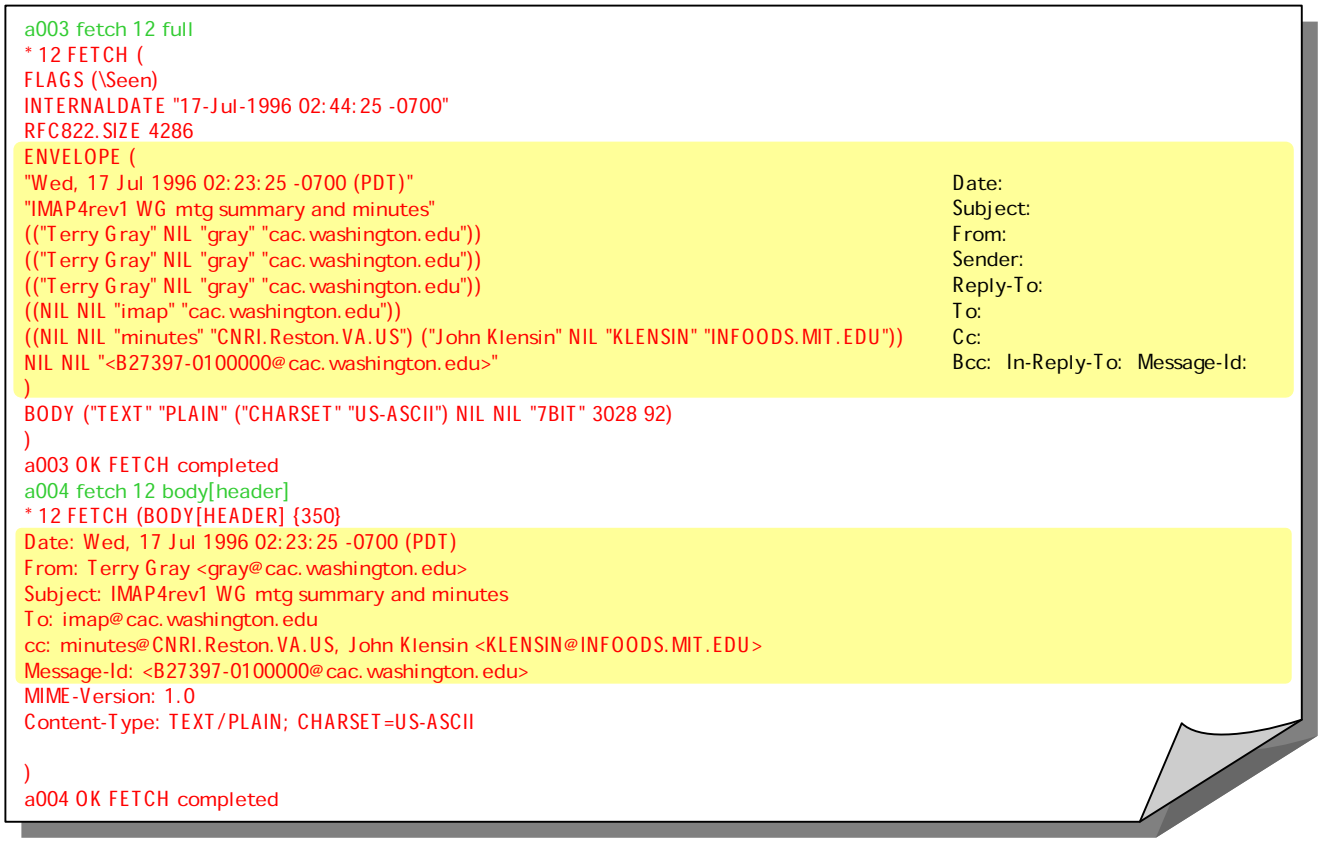
Parameter-Daten von Fetch

ALL	Entspricht „(FLAGS INTERNALDATE RFC822.SIZE ENVELOPE)“
BODY	Nicht erweiterte bzw. erweiterbare Form von „BODYSTRUCTURE“
BODY[section]<partial>	Die Nachricht in in Abhängigkeit der Parameter „HEADER“, „TEXT“, „MIME“, „HEADER.FIELDS“ und „HEADER.FIELDS.NOT“ (es wird auf jeden Fall das „\Seen“ Flag für diese Nachricht gesetzt)
BODY.PEEK[section]<partial>	Alternative Form von „BODY[section]“, die nicht das „\Seen“ Flag setzt
BODYSTRUCTURE	Die Struktur der Nachricht (vom Server geparste MIME Struktur)
ENVELOPE	Die Struktur des „Envelope“ (der vom Server geparste Header)
FAST	Entspricht „(FLAGS INTERNALDATE RFC822.SIZE)“
FLAGS	Die gesetzten Flags für diese Nachricht
FULL	Entspricht „(FLAGS INTERNALDATE RFC822.SIZE ENVELOPE BODY)“
INTERNALDATE	Das interne Datum der Nachricht
RFC822	Gleichbedeutend mit „BODY[]“ (aber die Rückgabesyntax ist anders)
RFC822.HEADER	Gleichbedeutend mit „BODY.PEEK[HEADER]“ (aber die Rückgabesyntax ist anders)
RFC822.SIZE	Die aus RFC 822 definierte Größe einer Nachricht
RFC822.TEXT	Gleichbedeutend mit „BODY[TEXT]“ (aber die Rückgabesyntax ist anders)
UID	Die UID der Nachricht

Response-Daten von Fetch

- die Rückgabe ist immer eine Liste
- der Inhalt der Liste besteht aus dem Anfrageparameter und dessen Ergebnis
- wurden mehrere Parameter abgefragt, werden die Antworten einfach hintereinander gestellt
- die wichtigsten Anfragen sind *Flags*, *Envelope* und *Body*

Ein Beispiel von Fetch



Die Struktur einer Nachricht

- eine Nachricht mit 48 Zeilen und einer Größe von 2279 Octets hat folgende Struktur:

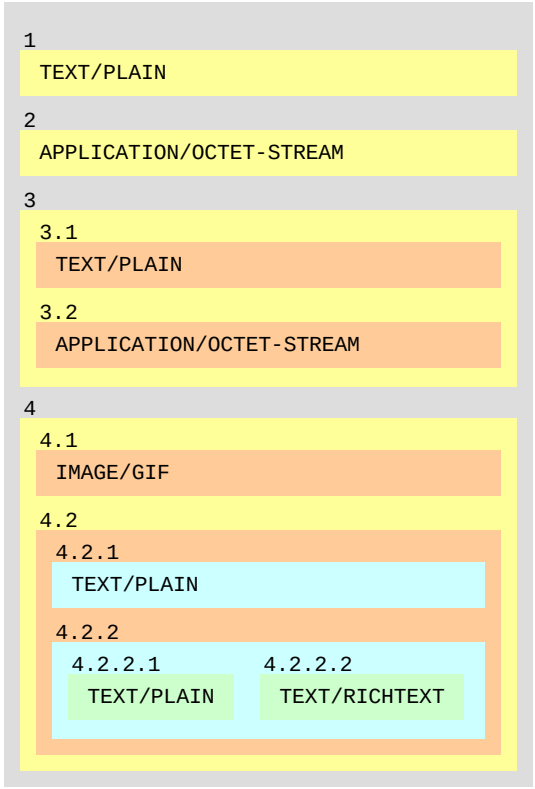
```
("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 2279 48)
```

- Teile von Multipart Nachrichten werden durch Nachstellen des Subtype gekennzeichnet:

```
( ("TEXT" "PLAIN" ("CHARSET" "US-ASCII") NIL NIL "7BIT" 1152 23)
  ("TEXT" "PLAIN" ("CHARSET" "US-ASCII" "NAME" "cc.diff")
   "<960723163407.20117h@mit.edu>" "Compiler diff" "BASE64" 4554 73)
  "MIXED" )
```

Gezielte Abfrage von Strukturen


```
HEADER     (RFC822 header of the message)
TEXT       MULTIPART/MIXED
1          TEXT/PLAIN
2          APPLICATION/OCTET-STREAM
3          MESSAGE/RFC822
3.HEADER   (RFC822 header of the message)
3.TEXT     (RFC822 text body of the message)
3.1        TEXT/PLAIN
3.2        APPLICATION/OCTET-STREAM
4          MULTIPART/MIXED
4.1        IMAGE/GIF
4.1.MIME   (MIME header for the IMAGE/GIF)
4.2        MESSAGE/RFC822
4.2.HEADER (RFC822 header of the message)
4.2.TEXT   (RFC822 text body of the message)
4.2.1      TEXT/PLAIN
4.2.2      MULTIPART/ALTERNATIVE
4.2.2.1    TEXT/PLAIN
4.2.2.2    TEXT/RICHTEXT
```



Erweiterungen von IMAP4

- mit NAMESPACE wird das Auslesen der Namensräume und deren Hierarchy-Separator möglich
- Access Control Lists stehen mit ACL zur Verfügung
- QUOTA liefert Befehle zur Speicherplatzbegrenzung
- mit der UIDPLUS Erweiterung werden die Befehle EXPUNGE , APPEND und COPY ergänzt
- der Befehl ID liefert die Möglichkeit Informationen über die Implementierung des Server auszulesen

Zusammenfassung E-Mail

- Versenden über SMTP
- Anmelden für SMTP über SMTP-after-POP oder SMTP-after-IMAP
- Abholen über POP3 oder IMAP4
- auf dem Server bearbeiten mit IMAP4

Klartextprotokolle

- Weitere *Klartextprotokolle*:
- FTP: *File Transfer Protocol*
- HTTP: *Hypertext Transfer Protocol*
- . . .

Warum Klartext?

- leicht zu debuggen
- leicht zu lernen
- gemeinsamer (alter) Standard (oft 7-Bit-ASCII)
- Umsetzung auf verschiedenen Architekturen leicht möglich

Warum nicht Klartext?

- nicht besonders effizient
- Befehle und Daten könnten Binär oft besser ausgedrückt werden
- dann aber Kommunikation nicht mehr direkt vom Menschen mitlesbar
- viele Protokolle aber auch Dokumentenformate verwenden daher wenigstens für Befehle und Metadaten Klartext

FTP (1)

- FTP bezeichnet den Dienst und das Protokoll gleichzeitig und setzt auf dem TCP-Protokoll auf
- „Reine Datenübertragung, ohne viel Schnickschnack“[1]
- Protokoll zur Übertragung von Daten zwischen zwei Computern
- FTP gehört zu den ältesten Protokollen des Internets und hat seine Ursprünge noch im ARPANet, dem militärischen Vorläufer des Internets
- wichtige RFCs: RFC 959, RFC 1579, RFC 1635

[1] `http://www.uni-erfurt.de/rechenzentrum/anleitungen/ftp.html`

FTP (2)

- FTP ist ein Client–Server-Dienst
- einer der beiden Computer fungiert immer als Server
- der andere als Client
- normalerweise: Serverseite bietet Daten an, Client fragt diese ab

FTP-Ports

- FTP-Standard benutzt zwei Ports: den Port 21/tcp und den Port 20/tcp
- Steuerverbindung:
- über Port 21 werden die Befehle an den Server übermittelt und die Antwort, die der Server generiert, empfangen
- Datenverbindung:
- die Daten (der Inhalt der Dateien) kommen über Port 20

Befehlssatz (1)

- umfangreicher Befehlssatz
- ähnlich wie auf CP/M-, Unix- oder DOS-Oberflächen
- (einige Befehle haben nahezu gleiche Funktionen)

```
ASCII  - Stellt ASCII-Übertragungsmodus ein
BINARY - Stellt Binär-Übertragungsmodus ein
BYE    - Beenden der FTP-Sitzung und des Interpreters
CD     - Wechseln des Verzeichnisses
DIR    - Verzeichnisinhalt anzeigen
GET    - Download einer Datei
LS     - Verzeichnisinhalt anzeigen
MGET   - Download mehrerer Dateien gleichzeitig
MPUT   - Upload mehrerer Dateien gleichzeitig
OPEN   - Verbindung zu einem FTP-Server aufnehmen
PUT    - Upload einer Datei
```

Befehlssatz (2)


```
!          - Betriebssystem-Fenster öffnen
?          - Lokale Hilfe
APPEND     - Verknüpfen von lokaler und Server-Datei
BELL       - Akustische Meldung, wenn Befehl ausgeführt ist
CLOSE      - Beenden der FTP-Sitzung
DELETE     - Löschen einer Datei
DEBUG      - Schaltet in den Debug-Modus
DISCONNECT - Beenden der FTP-Sitzung
GLOB       - Umschalten zwischen normaler und erweiterter Zeichenanzeige
HASH       - Grafische Anzeige der Übertragung (mit '#')
HELP       - Lokale Hilfe
LCD        - Wechseln des Verzeichnisses
LITERAL    - Senden eines willkürlichen FTP-Befehls
MDELETE    - Mehrere Dateien gleichzeitig löschen
MDIR       - Verzeichnisinhalt inklusive Unterverzeichnisse anzeigen
MKDIR      - Erstellen eines Verzeichnisses
MLS        - Verzeichnisinhalt inklusive Unterverzeichnisse anzeigen
PROMPT     - Aktivieren von interaktiven Ausgaben bei Mehrfach-Befehlen
PWD        - Anzeigen des aktuellen Verzeichnisses
QUIT       - Beenden der FTP-Sitzung und des Interpreters
QUOTE      - Senden eines willkürlichen FTP-Befehls
RECV       - Download einer Datei
REMOTEHELP - Hilfe des FTP-Servers anfordern
RENAME     - Datei umbenennen
RMDIR      - Verzeichnis entfernen
SEND       - Upload einer Datei
STATUS     - Anzeigen des aktuellen Status
TRACE      - Schaltet in den Packet-Tracing-Modus
TYPE       - Umschalten des Transfer-Typs
USER       - Senden von neuen User-Informationen
VERBOSE    - Umschalten in den ausführlichen Modus
```

FTP-Einschränkungen?

- das FTP-Protokoll stammt noch aus der Zeit, als es das Internet in der heutigen Form noch nicht gab
- zu dieser Zeit ging es um das Übertragen von Daten von einem Computer auf einen anderen
- dabei kam man noch ohne Virenscanner, NAT und Firewall aus
- das ursprüngliche FTP Protokoll weist eine Besonderheit auf, mit der einige NAT-Router und Firewalls nicht klarkommen

Active FTP (1)

- Active FTP ist das originäre FTP-Protokoll
- es bekam den Namen Active erst, nach dem das Passive FTP entwickelt wurde
- FTP ist ein auf TCP basierender Service und zeichnet sich durch eine Besonderheit aus: FTP verwendet zwei Ports
- andere Protokolle kommen fast immer mit nur einem Port je Verbindungsende aus
- der eine Port ist der Command-Port (21) und der andere der Data-Port (20)

Active FTP (2)

- Möchte sich ein Client mit einem FTP-Server verbinden, so öffnet der Client einen beliebigen unpriviligierten Port n aktiv (Ports > 1024)
- zum Command-Port 21 des Servers
- dort sendet er dem Server die Portnummer, auf dem er die Daten empfangen will
- dies ist i.d.R. die Portnummer $n + 1$

Active FTP (3)

Beispiel:

- Client öffnet Port 1230 aktiv
- sendet dem Server auf dessen Port 21 die Information
- „ich erwarte Daten auf Port 1231“
- der Server bestätigt den Emfang
- der Client öffnet den Port 1231 passiv
- nun öffnet der Server seinen Daten Port 20 und sendet Daten zum Port 1231 des Clients
- der Client bestätigt den Empfang jedes TCP Packetes (wie üblich)

Active FTP (4)

Probleme:

- für eine Firewall sieht die Sache aber sehr suspekt aus:
- Firewalls blockieren normalerweise jede Anfrage von außen
- Firewalls akzeptieren nur TCP-Pakete von außen, wenn diese vom empfangenden Port angefordert wurden
- beim NAT sieht es ähnlich aus, es gibt noch keinen Eintrag für den Verbindungsaufbau von außen
- FTP aber sendet von Port 20 Daten an Port 1231, die von diesem Port nicht angefordert wurden (die Anforderung stammte ja von Port 1230)
- gute Firewalls und NAT-Router erkennen einen FTP Transfer und behindern diesen nicht

Passive FTP (1)

- wird auch PASV-Mode genannt, weil der FTP-Befehl PASV dem Server mitteilt, das der passive Mode gewünscht ist
- im Gegensatz zum aktiven Mode leitet der Client beide Verbindungen zum Server ein
- damit wird das Firewall-Problem gelöst, da damit die Firewall über beide clientseitige Verbindungen Kenntnis erhält
- auch NAT-Router werden auf diesem Weg einen passenden Eintrag in der Übersetzungstabelle anlegen

Passive FTP (2)

- wenn der Client eine Verbindung zu einem FTP-Server wünscht,
- so öffnet er zwei unpriviligierte Ports > 1024, z.B. Port 1030 und 1031
- nun kontaktiert der erste Port den Server auf dessen Command-Port 21
- anstelle des PORT-Kommandos, wird nun PASV befohlen
- der Server weiß nun nicht auf welchem Port der Client die Daten empfangen will
- hat aber die Information, dass der Client eine passive Verbindung wünscht

Passive FTP (3)

- der Server öffnet seinerseits einen Port > 1024 und sendet die Portnummer mittels des FTP Befehls PORT an den Client
- der Client kennt nun den unpriviligierten Port, den der Server anstelle des Ports 20 für den Datenverkehr geöffnet hat
- und fordert nun von seinem Port 1031 (in unserem Beispiel) die Daten an

Aussicht

Übersicht für die kommende Woche:

- Klausur

Ende Teil 13. Danke für die Aufmerksamkeit.