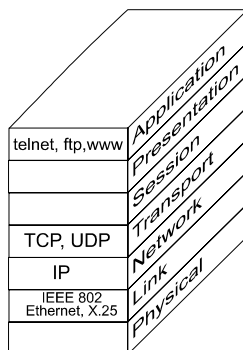


Servizi di UNIX (livello applicazione)



Servizi di tre tipi fondamentali

TERMINALE REMOTO

accesso a nodi remoti

FILE TRANSFER

possibilità di trasferire file tra nodi diversi

COMANDI REMOTI (applicazioni)

esecuzione di comandi remoti, anche specializzati, e riferimenti a servizi remoti

NEWS, MAIL, gopher, WWW (Trasparenza allocazione)

Alcuni sono solo per sistemi UNIX ⇒ **rlogin, rwho, rsh, rup, ...**

Altri più generali ⇒ **ftp, telnet, mail, ...**

Proprietà fondamentali di implementazione

Trasparenza allocazione (o meno)

Modelli Cliente/Server (senza stato con evoluzioni)

Standardizzazione

Servizi APPLICATIVI di un SO

a livello di applicazione per sistemi UNIX: protocolli

Virtual Terminal Protocol: **telnet**

File Transfer Protocol: **ftp**

Trivial File Transfer Protocol: **tftp**

Simple Mail Transfer Protocol: **smtp**

Network News System Transfer Protocol: **nntp**

Line Printer Daemon Protocol: **lpd**

Domain Name System: **dns**

Diffusione conoscenza (più o meno con trasparenza): **nntp, gopher, http, ...**

servizi remoti (UNIX BSD): **rsh, rwho, rlogin, ...**

telnet / rlogin (Virtual Terminal)

Implementazioni di virtual terminal

Per gestire l'eterogeneità di sistemi operativi ed hardware:

Il terminale locale diventa un terminale del sistema remoto

rlogin per i sistemi UNIX BSD (remote login)

telnet standard per sistemi con TCP/IP

Protocollo telnet eterogeneo

telnet costruito su TCP/IP

connessione TCP con **server** per accesso remoto

possibilità di aggancio a server qualunque

Caratteristiche

- **gestione eterogeneità** tramite interfaccia di terminale virtuale
- **Client** e **Server** negoziano le opzioni del collegamento (es., ASCII a 7 bit o a 8 bit)
- comunicazione **simmetrica ma funzionalità differenziate**

Due attività:

Client stabilisce una connessione TCP con **Server**

Client accetta i caratteri dall'utente e li manda al Server

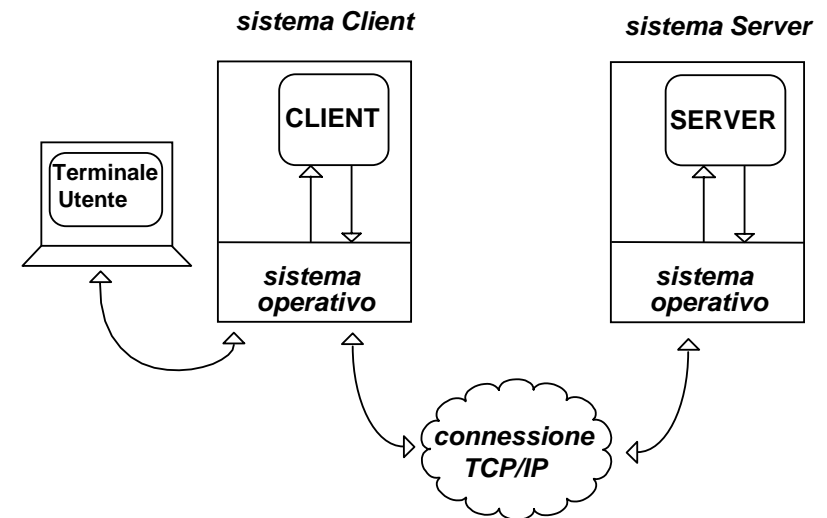
contemporaneamente

accetta i caratteri del server

e li visualizza sul terminale d'utente

Server accetta la richiesta di connessione del Client e inoltra i dati dalla connessione TCP al sistema locale

Esempio di connessione telnet



Telnet in due parti: cliente e servitore

telnet con nome logico host o indirizzo fisico IP

anche il *numero di porta*

telnet [host [port]]

Esempio:

```
telnet 137.204.57.33 (telnet deis33.deis.unibo.it)
```

```
username:antonio
```

```
password:*****
```

Il controllo del flusso viene fatto dal servitore (a default)

TERMINALE VIRTUALE

esigenza sentita in generale nelle reti e in particolare nel internetworking

problema: eterogeneità dei terminali

I terminali possono differire gli uni dagli altri per:

- il *set* di caratteri
- diversa *codifica* dei caratteri
- la *lunghezza* della linea e della pagina
- i *tasti funzione* individuati da diverse sequenza di caratteri (escape sequence)

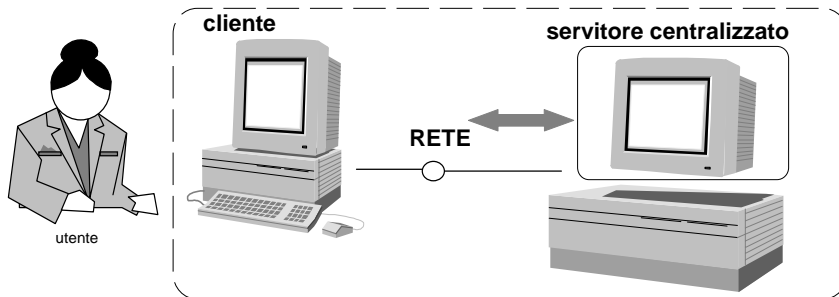
soluzione:

definizione di un **terminale virtuale**

Sulla rete si considera un **unico terminale standard**

In corrispondenza di ogni stazione di lavoro, si effettuano la conversione da terminale locale a terminale virtuale e viceversa.

telnet, rlogin sono basati su questo modello detto **standard NVT** (ossia **Network Virtual Terminal**)

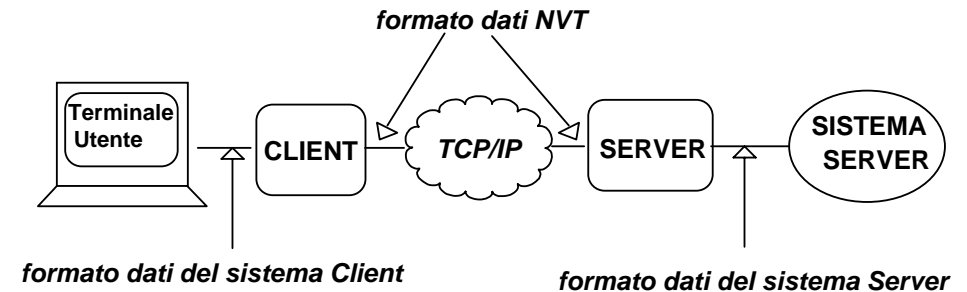


Implementazione

Uso di formato **NVT (Network Virtual Terminal)**

All'inizio NVT prevede uso di caratteri con rappresentazione 7 bit USASCII

(*caratteri normali* in byte con bit alto a 0, 8° bit resettato)



Client trasla caratteri utente nel formato NVT prima di inviarli al server

Server trasformazione dal formato NVT nel formato **locale** viceversa al ritorno

NEGOZIAZIONE

Possibilità di **negoziare** la connessione, sia alla *inizializzazione* sia *successivamente* per selezionare le opzioni del telnet (comunicazione half- full- duplex, determinare il tipo di terminale, codifica 7-8 bit)

Protocollo per negoziare le opzioni **simmetrico**, con messaggi:

will X will you agree to let me use option **X**

do X I do agree to let you use option **X**

don't X I don't agree to let you use option **X**

won't X I won't start using option **X**

NVT definisce un tasto di interruzione concettuale per richiedere la terminazione della applicazione

Caratteri di controllo NVT, USASCII

CODICE DI CONTROLLO ASCII	VALORE	SIGNIFICATO ASSEGNATO DA NTV
NUL	0	NESSUNA OPERAZIONE
BEL	7	SUONO UDIBILE/SEGNALE VISIBILE
BS	8	SPOSTAMENTO A SINISTRA DI UNA POSIZIONE
HT	9	SPOSTAMENTO A DESTRA DI UNA TABULAZIONE
LF	10	SPOSTAMENTO IN BASSO ALLA LINEA SUCCESSIVA
VT	11	SPOSTAMENTO IN BASSO DI UNA TABULAZIONE
FF	12	SPOSTAMENTO ALL'INIZIO DELLA PROSSIMA PAGINA
CR	13	SPOSTAMENTO SUL MARGINE SINISTRO DELLA RIGA
altri codici	—	NESSUNA OPERAZIONE

UNICA CONNESSIONE

Invio di caratteri di controllo e funzioni di controllo insieme con i dati normali

Funzioni di controllo NVT

(codificati con bit più significativo a 1)

SEGNALE	SIGNIFICATO
IP	INTERRUZIONE DEL PROCESSO
AO	ABORT IN USCITA (SCARTA I CONTENUTI DEI BUFFER)
AYT	CI SEI? (TEST DELLA PRESENZA DEL SERVER)
EC	CANCELLA IL PRECEDENTE CARATTERE
EL	CANCELLA LA CORRENTE LINEA
SYNC	SINCRONIZZAZIONE
BRK	SOSPENSIONE TEMPORANEA (ATTESA DI UN SEGNALE)
WILL	SI PROPONE AL PARI DI ATTUARE UNA AZIONE
WONT	SI PROPONE AL PARI DI NON ATTUARE UNA AZIONE
DO	SI ACCETTA DI ATTUARE UNA AZIONE
DONT	NON SI ACCETTA DI ATTUARE UNA AZIONE

NEGOZIAZIONE

chi ha la iniziativa determina l'accordo necessario a lavorare sulla connessione stabilita

WILL ECHO

DO ECHO

DO "terminal type" XYZ

WILL "terminal type" XYZ

e si va avanti... in modo sorgente

Le richieste negative **DONT** e **WONT** sono sempre accettate

MODI di LAVORO

half-duplex

one char at a time (echo fatto sempre dal server)

problemi di overhead

one line at a time

problemi di ritardo nei dati

linemode

Uso di dati urgenti

Lo stream impiega i dati fuori banda (out-of-band signal)

per ovviare al caso di stream di dati pieno

Riconosciuto

flush

scarta tutto

no client flow control

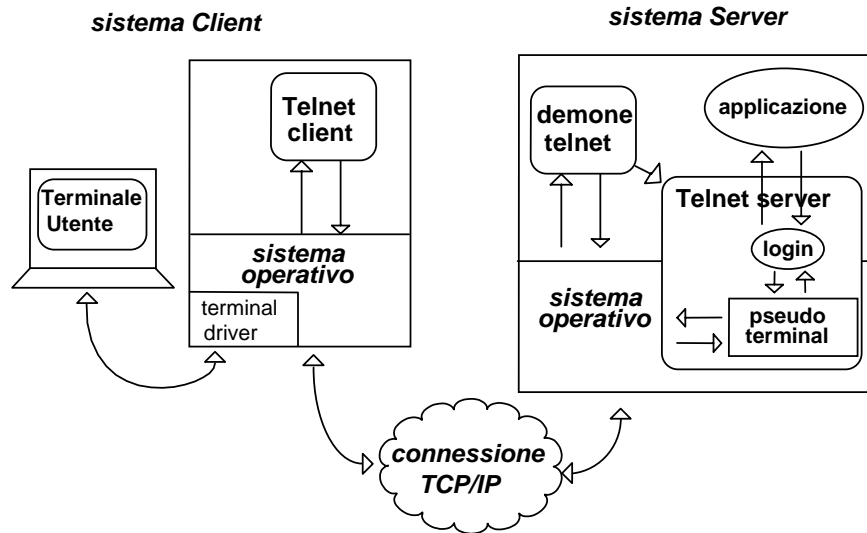
il cliente non fa più controllo di flusso

client flow control

sliding window

cambiamento della dimensione e controllo del cliente

Implementazione



Pseudo-terminal può essere una funzione del sistema operativo

Se il sistema operativo ha una *astrazione di pseudoterminale* telnet ==> programma applicativo

vantaggio --> modifiche e controllo facili
svantaggio --> inefficienza

RLOGIN

Servizio di login remoto => login su un'altra macchina UNIX

```
rlogin lia02.deis.unibo.it
username: antonio
password: *****
```

Se l'utente ha una **home directory** in remoto
 accede a quel direttorio

Altrimenti,
 l'utente entra nella **radice** della macchina remota

Il servizio di rlogin UNIX supporta il concetto di trusted hosts.
 Utilizzando i file

.rhosts

/etc/hosts.equiv

per garantire corrispondenze tra utenti.

(uso senza password)

In genere, il **superutente** non può passare da una macchina ad un'altra

Problemi di **sicurezza** rlogin:

- nell'uso di .rhosts ed hosts.equiv
- password in chiaro

Caratteristiche RLOGIN

- conosce l'ambiente di partenza e quello di arrivo, ha nozione di stdin, stdout e stderr (collegati al client mediante TCP)
- esporta l'ambiente del client (es. il tipo di terminale) verso il server
- utilizzo di **una connessione TCP/IP e più processi (due per parte)**

Si lavora solo a un carattere alla volta (**Nagle on**)

- **flow control**: il client rlogin tratta **localmente** i caratteri di controllo del terminale (<Ctrl><S> e <Ctrl><Q> fermano e fanno ripartire l'output del terminale) (in modo simile il <Ctrl><C>)

out-of-band signalling per i comandi dal server al client (es. flush output per scartare dei dati, comandi per il resize della finestra e per la gestione del flow control)

in-band signalling per i comandi dal client al server (spedizione dimensione finestra)

- rlogin molto più snello di telnet ma minori prestazioni
migliaia di linee vs. decine di migliaia

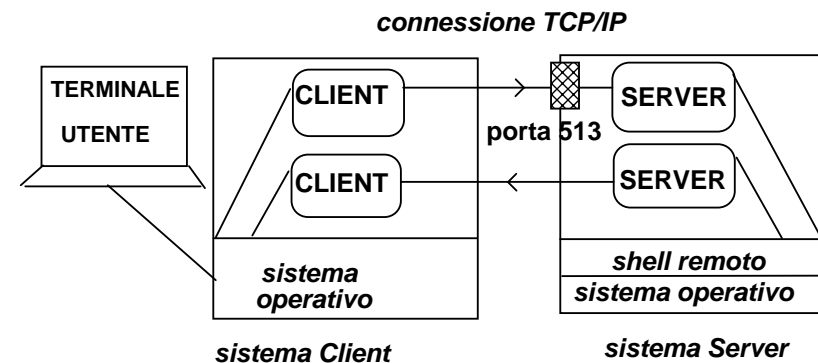
Implementazione

Client rlogin e Server remoto (server rlogind)

Il **client** crea una **connessione TCP** al server rlogind

Il **client** rlogin spezza le funzioni di ingresso/uscita
il genitore gestisce i caratteri che vanno allo shell remoto
il figlio gestisce i caratteri in arrivo dallo shell remoto

Il server si collega ad uno shell remoto
con coppia master-slave di uno pseudoterminale



Invocazione remota

Anche *rsh*

invoca l'interprete (shell) remoto UNIX
con gli argomenti della linea di comando
`rsh nodo_remoto comando`

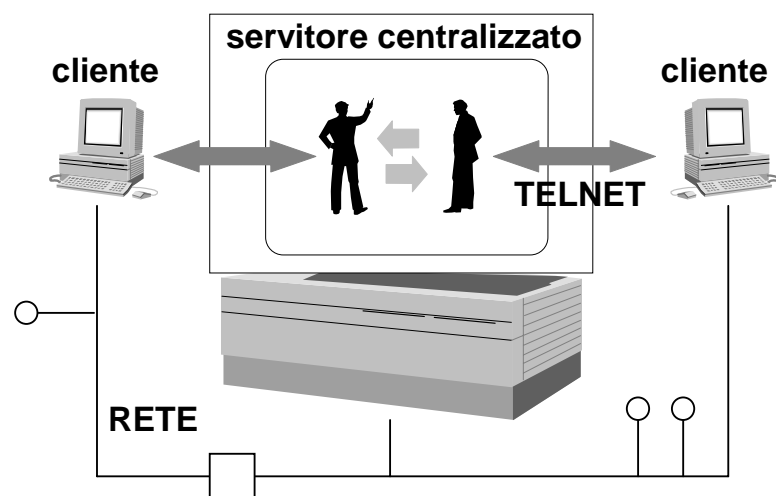
EVOLUZIONI possibili

Modello di comunicazione

client / server evoluto

- parallelo
- stateful
- sul server stato condiviso

Uso del server come framework di interazione



Sicurezza

- autenticazione utenti tramite password
- liste d'accesso sugli oggetti
- domini di protezione tramite ruoli standard e relazioni mutue

ACCESSO E TRASFERIMENTO FILE

Uso di TCP (affidabile ed orientato alla connessione)

ftp (file transfer protocol)

tftp (trivial file transfer protocol) basato su **UDP**

Permettono la copia di file nei due sensi

Proprietà ulteriori:

- Eseguito dai programmi applicativi o con accesso **interattivo** (si può richiedere la lista dei file di un direttorio remoto, o creare un direttorio remoto, etc.)
- Specifica del **formato** dei dati (rappresentazione): file di tipo testo o binario
- Controllo **Identità** (login e password)

Comandi di trasferimento file

```
put local-file [remote-file]
```

memorizza un file locale sulla macchina remota

```
get remote-file [local-file]
```

trasferisce un file remoto sul disco locale

mget e **mput** utilizzano metacaratteri nei nomi dei file
altri comandi:

```
help, dir, ls, cd, lcd, ...
```

tftp più semplice e con meno possibilità (uso di UDP)

Esistono nodi server di ftp che sono contenitori di informazioni a cui si può accedere "liberamente"

Uso di ftp anonymous verso i server

Tutte le informazioni viaggiano in chiaro

Codifica numerica

La prima cifra codifica le **interazioni**

- 1xx Risposta positiva preliminare
- 2xx Risposta positiva completa
- 3xx Risposta positiva intermedia
- 4xx Risposta negativa transitoria
il comando può essere ripetuto
- 5xx Risposta negativa permanente

La seconda cifra codifica le **risposte**

- x0x Errore di Sintassi
- x1x Informazione
- x2x Connessione
- x3x e x4x Codici non specificati
- x5x Filesystem status

La terza cifra specifica più precisamente

- 150 risposta preparatoria per filelist
- 200 OK
- 226 trasferimento completo
- 331 username OK, serve la password

Per il **formato** delle **informazioni di controllo**? =>
uso di NVT

Esempio di ftp anonymous

```
antonio deis33 ~ 7 > ftp didahp1.deis.unibo.it
Connected to didahp1.
220 didahp1 FTP server (Version 1.7.109.2 Tue Jul 28
23:32:34 GMT 1992) ready.
Name (didahp1.deis.unibo.it:antonio): anonymous
331 Guest login ok, send ident as password.
Password:XXXXXXXXXXXX
230 Guest login ok, access restrictions apply.
ftp> ls
200 PORT command successful.
150 Opening ASCII mode data connection for file list.
bin
etc
pub
RFC
incoming
prova.txt
226 Transfer complete.
37 bytes received in 0.0035 seconds (10 Kbytes/s)
ftp> ascii
200 Type set to A.
ftp> get prova.txt
200 PORT command successful.
150 Opening ASCII mode data connection for prova.txt
(8718 bytes).
226 Transfer complete.
local: prova.txt remote: prova.txt
8719 bytes received in 0.025 seconds
(3.3e+02 Kbytes/s)
ftp> get pippo.txt -
// si mostra il contenuto direttamente a console
ftp> get pippo.txt "| more"
// si passa il contenuto a more
```

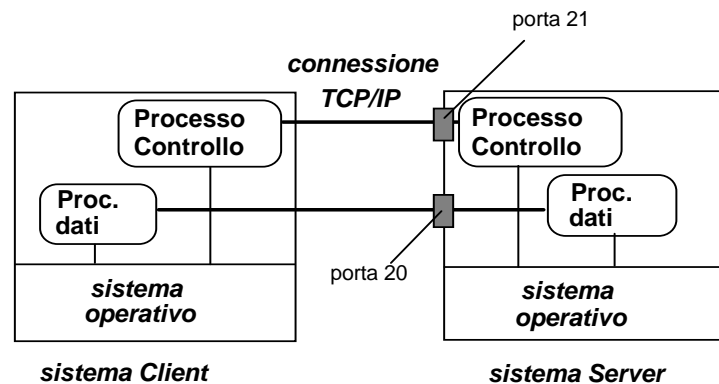

Implementazione FTP

Vari tipi di file riconosciuti

filetype ASCII, EBCDIC, binary, local
format Nonprint, telnet fmt, Fortran fmt
structure stream, record, page
TX mode stream, block, compressed

Accesso concorrente da parte di più client ad un unico server
uso di TCP per le connessioni al server

Almeno due collegamenti per ogni client e per ogni server:
UNA CONNESSIONE DI CONTROLLO e **UNA DI DATI**



Dettagli della implementazione:

Un processo **master** del server attende connessioni (processo **ftpd**, demone di ftp) e crea uno **slave** per ciascuna richiesta

Lo slave è composto da:

- un processo per il **collegamento di controllo** con il client (persiste per tutta la durata del collegamento)
- un processo per il **trasferimento dati** (possono essere molti nello stesso collegamento)

Anche il client usa processi separati per la parte di controllo e di trasferimento dati

Uso di numeri di porta tcp

TCP: entrambi gli estremi individuano una connessione

FTP prevede almeno **due connessioni**:
una **controllo persistente** e
una **dati** per ogni trasferimento

Collegamento controllo

la porta di trasferimento lato server è fissa (**21**)
specifica della porta da parte del cliente (**xxx**)

Collegamento dati

la porta di trasferimento lato server è fissa (**20**)
porta da parte del cliente (**xxx/yyy**)

Client **collegamento iniziale** con server una propria porta (**xxx**)

Servizio sequenziale, la stessa porta **xxx** del cliente può essere usata per connessione dati

Il valore di porta passato al server rappresentano una forma di coordinamento senza cui il servizio non può funzionare

Servizi paralleli per lo stesso cliente o porte multiple

Il cliente può indicare una porta addizionale (**yyy**)
normalmente **porte successive (per connessioni diverse)**

Possibilità di stato della connessione

In caso di trasferimento di grandi moli di dati, se ci si blocca, non si deve ripartire dall'inizio, ma dall'ultima posizione trasferita

Per quanto tempo si tiene lo stato? E dove lo si mantiene?

Confronto telnet ftp

Servizio	FILE TRANSFER ftp ftp	VIRTUAL TERMINAL telnet rlogin
Oggetto	file	caratteri
Distribuzione Informazioni	punto a punto	punto a punto
Protocollo	NVT	NVT

SERVIZI SINCRONI

vs.

SERVIZI ASINCRONI

La posta elettronica

La posta elettronica (**e-mail**) permette lo scambio di messaggi tra utenti, in modo simile al servizio postale

Caratteristica fondamentale: servizio **asincrono**
(a differenza di telnet ed ftp)

il mittente non aspetta il destinatario

spooling

I messaggi possono essere dei semplici testi oppure degli interi file (uso alternativo ad ftp)

Mail Esempio di uso

```
antonio deis33 ~ 12 >Mail beppe@ing.unibo.it
Subject: Prova di mail
```

```
testo del mail
ctrl-D
```

```
-----
beppe ingbo ~ 10 > Mail
Mail version SMI 4.1-OWV3 Mon Sep 23 07:17:24 PDT
1991 Type ? for help.
"/usr/spool/mail/beppe": 1 message 1 unread
>U 1 antonio Fri May 22 16:48
14/296 Prova di mail
```

& **return**

```
Message 1:
From antonio Fri, 22 May 16:48:38 1999
Received: by ing.unibo.it (4.1/4.7); Fri, 22 May 99
16:48:37 +0200
From: antonio (Antonio Corradi)
Subject: Prova di mail
To: Beppe
Date: Fri, 22 May 99 16:48:39 MET DST
X-Mailer: ELM [version 2.3 PL11]
Status: RO
Testo del mail
```

Indirizzi di mail

destinatario come

```
{  identificatore IP nodo di destinazione  
  mailbox sul nodo (nome login)  
}
```

Indirizzi di posta elettronica

varie possibili forme

From:acorradi@deis.unibo.it (Antonio Corradi)
postmaster mailbox del postmaster in ogni dominio
MAILER-DAEMON segnalazioni di problemi

Altri indirizzi

mappaggio identificatori distinti in nomi di sistema
anche **pseudonimi (aliases)** e **mail forwarding**

un utente può avere **più identificatori di mail**
e anche **unico identificatore per un gruppo di destinatari**

NOMI MULTIPLI

più sottodomini e nomi multipli

```
acorradi@deis.unibo.it  
acorradi@deis33.deis.unibo.it  
antonio@deis33.deis.unibo.it
```

electronic mailing list anche con destinatari non locali

Esempio

nella mailing list di A, x mappato in y di B
nella mailing list di B, y mappato in x di A

Problema: *Ciclo senza fine*

Formato dei messaggi

Header

```
From:  
To:  
Date:  
Subject:  
Corpo:
```

From: indirizzo del **mittente**
To: mailbox cui il messaggio va recapitato
anche **più indirizzi** di destinazione
Date: la **data** di spedizione
Subject: il **soggetto** del messaggio

opzionali

Cc: **copia** ai destinatari
Bcc: **copia nascosta** ai destinatari
Reply-To: **indirizzo** per la risposta
Message-Id: **Identificatore unico** del messaggio

Corpo

il testo dei messaggi è in formato ASCII

Per estendere il formato del corpo due vie:

- prevedere la codifica ascii dei binari
- estensione ex novo (con introduzione di nuovi tipi riconosciuti associati ad una parte del messaggio)

MIME (Multipurpose Interchange Mail Extension)

possibilità di inserimento di messaggi con formati diversi
in un unico corpo di un messaggio che il protocollo
riconosce automaticamente

MIME (Multipurpose Interchange Mail Extension)

messaggi con formati diversi nel corpo ASCII text

Mime-Version:
Content-Type:
Content-Transfer-Encoding:
Content-ID:
Content-Description:

Content-type Subtype

text	<i>plain</i> <i>richtext</i> <i>enriched</i>	testo non formattato testo con semplice formattazione (bold, ecc) raffinamento del richtext
multipart	<i>mixed</i> <i>parallel</i> <i>digest</i> <i>alternative</i>	parti da processare seq. parti su cui lav. in parallelo message digest molte copie con la stessa semantica (in alternativa)
message	<i>rfc822</i> <i>partial</i> <i>external-body</i>	un altro messaggio di mail frammento di un messaggio puntatore al messaggio
application	<i>octet-stream</i> <i>postscript</i>	dati arbitrari file postscript
image	<i>jpeg</i> <i>gif</i>	file ISO 10981 immagine Graphic Interchange Format
video	<i>mpeg</i>	file ISO 11172 stream
audio	<i>basic</i>	formato audio 8-bit

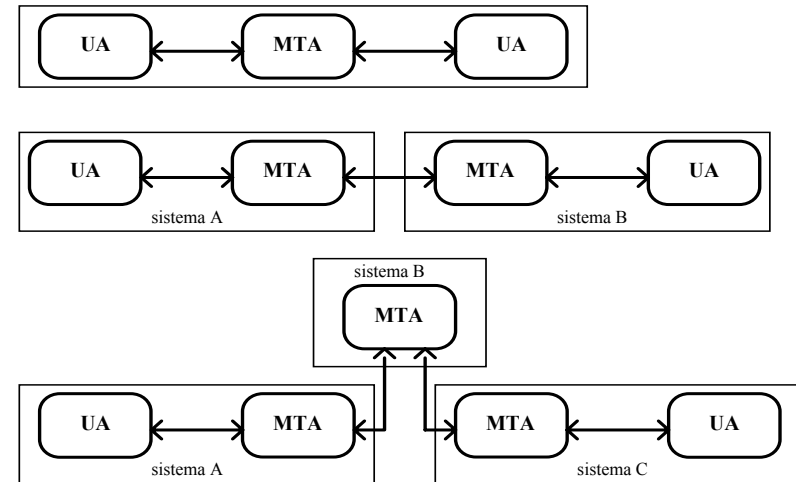
Content-Transfer-Encoding:
7bit (NVT ASCII), 8-bit, binary-encoding, ...

Architettura del servizio di mail

Uso di comunicazioni **punto a punto**
attraverso una rete di
User Agent (UA) e
Mail Transfer Agent (MTA)

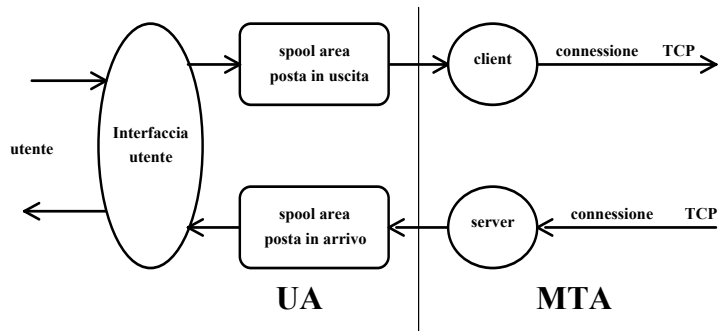
Mail Transport Agent (MTA) trasferisce mail dal user agent (UA) sorgente a quello di destinazione

Diversi metodi di collegamento sistemi di posta elettronica



Uso di mailbox come area riservata ad un **solo utente**

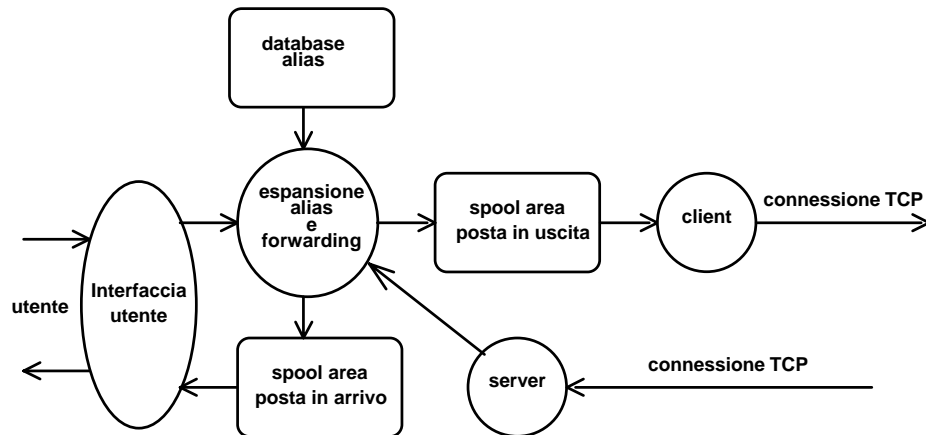
Componenti del servizio di posta elettronica



Il processo in background UA diventa il cliente di un MTA

- mappa il nome della destinazione in indirizzo IP
- tenta la connessione TCP con il mail server destinazione
- Se OK, copia del messaggio alla mailbox remota

Trattamento dei nomi multipli di utente e liste



NOMI di mail e Domain Name System (DNS)

ROUTING tra MTA

Il sistema di nomi della posta elettronica può

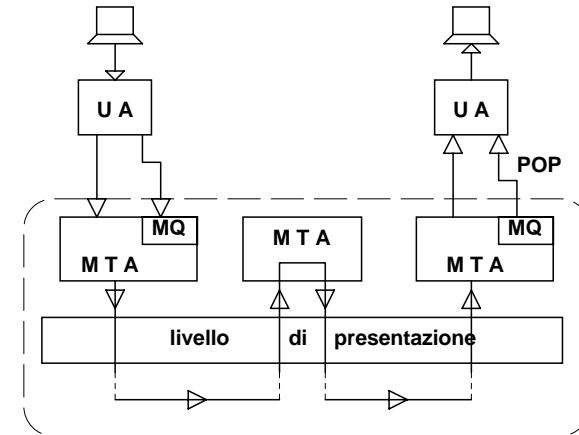
- basarsi sui nomi di DNS
- basarsi su altri cammini e percorsi

I diversi MTA possono organizzarsi anche in modo del tutto indipendente dalle normali forme di routing di IP
diverso dal sistema di corrispondenze di IP

Il sistema di nomi standard DNS può definire percorsi dedicati di mail distinti e trattati a parte
vedi record DNS tipo MX

Connessione di tipo **end-to-end diretto** (TCP-IP)

Uso di **mail gateway** (macchine intermedie)



La porta TCP per gli scambi tra MTA è la **25**

e i singoli messaggi sono regolati

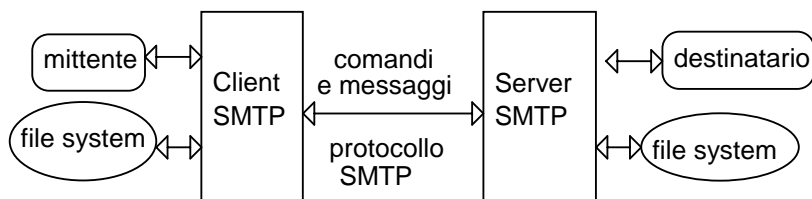
POP Post Office Protocol

Molti lettori diversi di posta elettronica: Mail, mail, elm, eudora, anche **sicuri**, IMAP. ecc.,

protocollo SMTP

standard per il trasferimento della mail
Simple Mail Transfer Protocol RFC 821

Scambi di messaggi codificati tra un client ed un server



PROTOCOLLO SMTP

Comandi cliente ----- **Risposte server**

Esempio

sender 'MAIL FROM:' nome del mittente

receiver 'OK'

sender 'RCPT TO:' nome del destinatario

receiver 'OK' abilitato

sender 'DATA'
linee di testo del messaggio

sender '< cr-lf> < cr-lf>' fine messaggio

receiver 'OK'

I ruoli tra sender e receiver (o client e server) possono essere invertiti per trasmettere la posta diretta nel verso opposto.

COMANDI: parole composte di caratteri ascii:

RISPOSTE: composte di codice numerico di 3 cifre e testo

Codifica

La prima cifra codifica le interazioni

1xx Comando accettato

2xx Risposta positiva completa

3xx Risposta positiva intermedia

4xx Risposta negativa transitoria

il comando può essere ripetuto

5xx Risposta negativa permanente

La seconda cifra codifica le risposte

x0x Sintassi

x1x Informazione

x2x Connessione

x3x e x4x Codici non specificati

x5x Mail system (stato del receiver)

La terza cifra specifica più precisamente

Procedure di SMTP

Procedura di invio come *mail transaction*

MAIL TRANSACTION

fatta in modo da completare la trasmissione

Se tutto va bene OK

Se problemi

messaggi disordinati e ripetuti

(azioni di posta idempotenti ?)

S: MAIL FROM:<Smith@Alpha.ARPA>
R: 250 OK
S: RCPT TO:<Jones@Beta.ARPA>
R: 250 OK
S: RCPT TO:<Green@Beta.ARPA>
R: 550 No such user here
S: RCPT TO:<Brown@Beta.ARPA>
R: 250 OK
S: DATA
R: 354 Start mail input; end with <CRLF>.<CRLF>
S: Blah blah blah...
S: ...etc. etc. etc.
S: <CRLF>.<CRLF>
R: 250 OK

Return-Path:

<@GHI.ARPA, @DEF.ARPA, @ABC.ARPA:JOE@ABC.ARPA>
Received: from GHI.ARPA by JKL.ARPA ; 27 Oct 81 15:27:39 PST
Received: from DEF.ARPA by GHI.ARPA ; 27 Oct 81 15:15:13 PST
Received: from ABC.ARPA by DEF.ARPA ; 27 Oct 81 15:01:59 PST
Date: 27 Oct 81 15:01:01 PST
From: JOE@ABC.ARPA
Subject: Improved Mailing System Installed
To: SAM@JKL.ARPA
This is to inform you that ...

MAIL FORWARDING

forward-path non corretto

251 User not local; will forward to forward-path

551 User not local; please try forward-path

VERIFYING AND EXPANDING

verificare di user name (VRFY)

espansione di mailing list (EXPAN)

VRFY user-name

- i) 250 'username completo' <indirizzo>
- ii) 251 User not local; will forward to <indirizzo>
- iii) 551 User not local; please try <indirizzo>
- iv) 550 That is a mailing list, not a user
550 String does not match anything
- v) 553 User ambiguous.

EXPAN <mailing-list>

S: EXPAN Example-People
R: 250-Jon Postel <Postel@USC-ISIF.ARPA>
R: 250-Fred Fonebone <Fonebone@USC-ISIQ.ARPA>
R: 250-Sam Q. Smith <SQSmith@USC-ISIQ.ARPA>
R: 250-Quincy Smith <@USC-ISIF.ARPA:Q-Smith@ISI-VAXA.ARPA>
R: 250-<joe@foo-unix.ARPA>
R: 250 <xyz@bar-unix.ARPA>

OPENING E CLOSING

HELO <domain> <CR-LF>

QUIT <CR-LF>

RESET (RSET)

abort della transazione corrente; receiver deve inviare OK

TURN (TURN)

intenzione di scambio dei ruoli

USENET News

Un insieme di **gruppi** di discussione

Ogni gruppo riguarda un particolare argomento e permette di partecipare a una discussione su tale argomento, scambiando informazioni e facendo domande.

Insiemi aperti di interessi pubblici

GERARCHIE PRINCIPALI DI NEWS

comp	(COMPUTER)
misc	(MISCELLANEOUS)
news	(NEWS)
rec	(RECREATIVE)
soc	(SOCIETY)
sci	(SCIENCE)
talk	(TALK)
alt	(ALTERNATIVE)
bit	(BITNET)
biz	(BUSINESS)

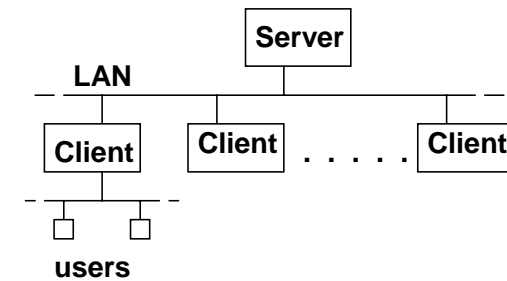
SOTTOGERACHIE DI 'comp.unix'

admin, aix, amiga, aux, internals, large, misc, programmer, question, etc ...

STORIA

1979	3 macchine uucp
1980	anews con due soli gruppi
1982	bnews

Architettura del servizio di NEWS



Nodo **client**:

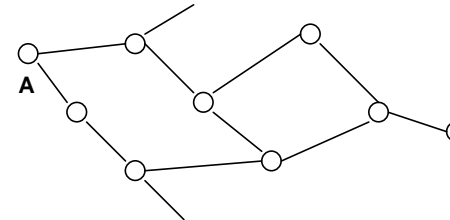
- un **client** di news mantiene le news
- presenza di **lettori** di news

Il **client** si coordina con il/i **server** per ottenere le news

I client sono *strumenti per l'accesso applicativo alle news e consentono anche di inviare news ai gruppi di interesse.*

Uso di agenti con **TCP/IP** di connessione

News: uso di **database coordinati** per le informazioni



Protocollo news:

Il protocollo è **NNTP (USENET)**

Comandi cliente ----- Risposte server

COMANDI: parole composte di caratteri ascii:

RISPOSTE: composte di codice numerico di 3 cifre e testo

In genere gli agenti si coordinano usando la **porta 119**

Protocollo NNTP

Protocolli a negoziazione

(connessione con well known porta 119)

Come smtp, così nntp (USENET)

comandi e risposte

il server restituisce una risposta al comando del client con il risultato dell'azione chiesta

comandi sono una parola di comando (ASCII)

più parametri separati e fine con carattere <CR> <LF>

risposte di testo e di stato

le risposte di testo

linee successive con un <CR> <LF>

le risposte di stato

stato dal server per l'ultimo comando

codice numerico di tre cifre

prima cifra successo o meno

1xx - messaggio informativo

2xx - comando ok

3xx - comando non ancora ok, richiesta del resto

4xx - comando corretto ma non eseguito

5xx - comando non implementato, o scorretto, o errore

seconda cifra categoria della risposta

x0x - messaggi di connessione, setup, e vari

x1x - selezione newsgroup

x2x - selezione articoli

x3x - funzioni di distribuzione

x4x - posting

x8x - estensioni non standard

x9x - debugging output

100 help text

190 through

199 debug output

200 server ready - posting allowed

201 server ready - posting not allowed

400 service discontinued

500 command not recognized

501 command syntax error

502 access restriction or permission denied

503 program fault - command not performed

PROTOCOLLO NNTP

Comandi cliente ----- Risposte server

COMANDI: parole composte di caratteri ascii:

RISPOSTE: composte di codice numerico di 3 cifre e testo

In genere gli agenti si coordinano usando la port 119

CODICI NUMERICI DI RISPOSTA

Utilizzati per la gestione automatica delle risposte.

C: GROUP msgs

S: 211 103 402 504 msgs Your new group is msgs

C: ARTICLE 401

S: 423 No such article in this newsgroup

C: ARTICLE 402

S: 220 402 4105@xyz-vax.ARPA

Article retrieved, text follows

S: (invio del testo da parte del server)

S: 205 XYZ-VAX news server closing connection. Goodbye

SERVIZI SINCRONI e ASINCRONI

Servizio	FILE TRANSFER	VIRTUAL TERMINAL
Oggetto	ftp tftp file	telnet rlogin caratteri
Distribuzione Informazioni	punto a punto	punto a punto
Protocollo	NVT	NVT

Servizio	POSTA ELETTRONICA	NEWS
Oggetto	messaggi	messaggi
Distribuzione	mailbox	database centralizzati che sono distribuiti
Protocollo	SMTP	NNTP

Per **USENET**

- la **dimensione globale** anche delle informazioni
- la distribuzione anche a **flooding** e a **gruppi**
- ⊗ nessuna sicurezza

SI COMINCIA A DELINEARE LA IDEA DI UNA INFRASTRUTTURA DI SUPPORTO