



Università degli Studi di Bologna  
Scuola di Ingegneria

# Corso di Reti di Calcolatori T

***Servizi Applicativi standard Internet  
Non solo UNIX***

**Antonio Corradi  
Luca Foschini**

Anno accademico 2020/2021

# SERVIZI APPLICATIVI INTERNET

---

**Internet si è diffusa attraverso la proposta di servizi da diffondere agli utenti già agganciati e creando una offerta che potesse invogliare nuovi altri clienti**

**Le applicazioni sono costruite sui protocolli con interfacce locali diverse**

**Applicazioni oggetto di standardizzazione**

## **Applicazioni cliente servitore**

- Stampa remota
- Terminale remoto
- Trasferimento file

## **Applicazioni di infrastruttura distribuita**

- Sistemi di nomi globali per Internet
- Mail
- News

**Le prime sono tutte applicazioni punto a punto che coinvolgono due entità alla volta,**

**Le seconde invece prevedono una infrastruttura sempre presente per fornire servizio**

# SERVIZI APPLICATIVI INTERNET

---

**Internet definisce protocolli standard alla base della realizzazione di applicazioni per comunicazione tra nodi diversi (usando il livello trasporto)**

**Applicazioni 'indipendenti' dal sistema operativo**

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, ...**

**Applicazioni possibili nel solo sistema operativo UNIX**

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

# SERVIZI DI UNIX (LIVELLO APPLICAZIONE)

---

**Servizi clienti servitore 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

## INFRASTRUTTURA DISTRIBUITA

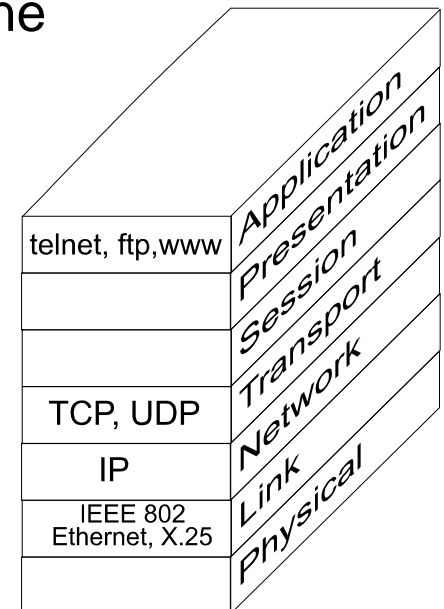
NEWS, MAIL, gopher, WWW con trasparenza allocazione

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

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

## Proprietà fondamentali delle implementazione

- **Trasparenza allocazione** (o meno)
- **Modello Cliente/Servitore** (con stato o meno)
- **Standardizzazione**



# TERMINALE REMOTO (VIRTUAL TERMINAL)

---

Per **gestire l'eterogeneità di sistemi operativi e hardware**: il terminale locale diventa un **terminale sul e del sistema (operativo) remoto**

- **telnet** standard per sistemi con TCP/IP
- **rlogin** per i sistemi UNIX BSD (remote **login**)

## Protocollo telnet eterogeneo

- **telnet** costruito su **connessioni TCP/IP**
- **connessione TCP con server** per accesso remoto
- **possibilità di aggancio C/S a server qualunque**

## Caratteristiche di telnet

- **comunicazione simmetrica con funzioni distinte, complementari e differenziate**
- **gestione eterogeneità** tramite interfaccia di **terminale virtuale**
- **negoziazione dinamica** delle **opzioni del collegamento** tra **Client e Server** (ASCII a 7 bit o a 8 bit, con bit di controllo, ...)

# TELNET: MODELLO C/S

---

**Cliente** → processo con due attività principali:

1. **Stabilisce una connessione TCP con Server, accetta i caratteri dall'utente e li manda al Server in invio**
2. **Accetta i caratteri del server in risposta e li visualizza sul terminale d'utente**

**Server** → processo che deve sia **accettare la richiesta di connessione** del Client e **inoltrare i dati** dalla connessione TCP al **sistema locale** sia **continuare a ricevere richieste**

1. **Creazione demone** sul server per la gestione del servizio
2. **Generazione di un processo figlio** (da parte del demone) per la gestione delle singole sessioni

# ESEMPIO DI CONNESSIONE TELNET

Telnet in due parti: **cliente** e **servitore** ( e demone)

Posso collegarmi ad un server telnet o *qualunque*

**telnet** con nome logico host o indirizzo fisico IP

anche il *numero di porta da contattare* (23 a default per telnet server)

```
telnet [ host [ port ] ]
```

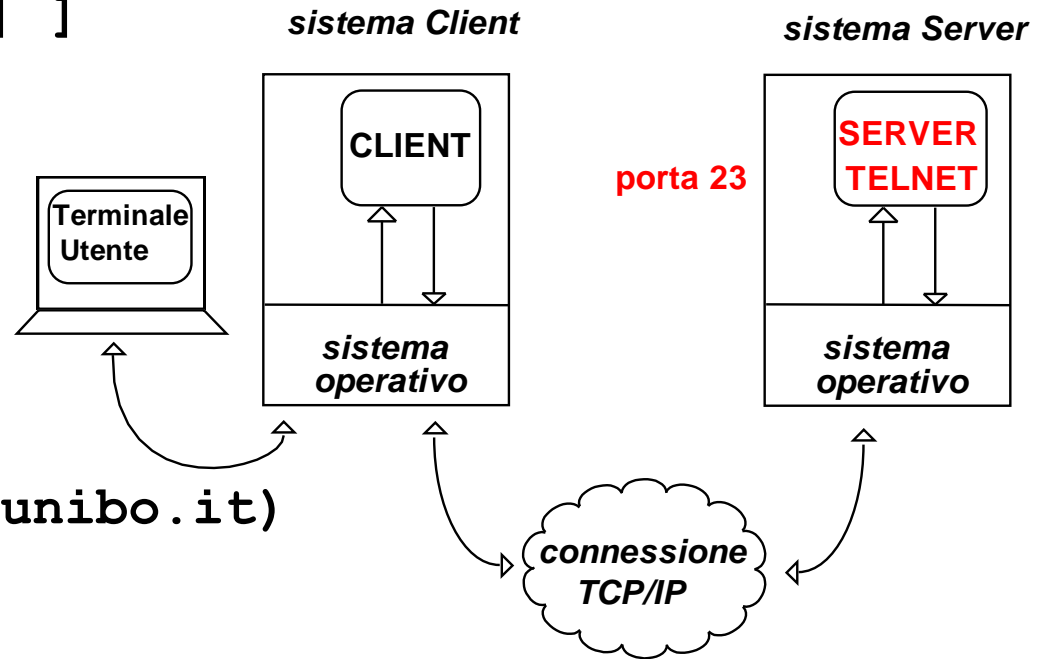
Esempio:

```
telnet 137.204.57.33
```

```
(o telnet disi33.disi.unibo.it)
```

```
username:antonio
```

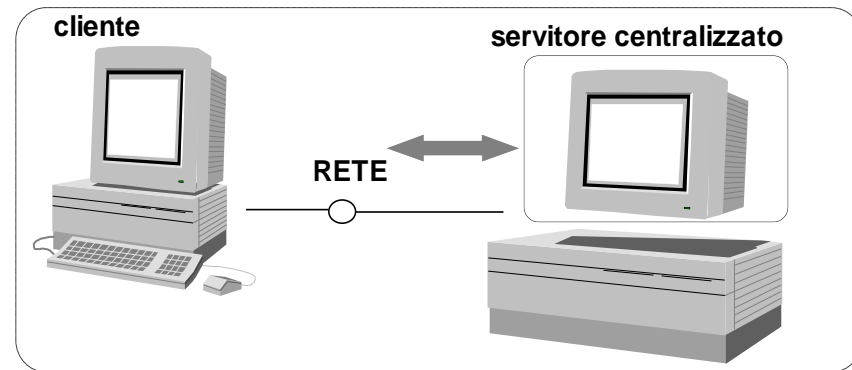
```
password:*****
```



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** e in corrispondenza di ogni stazione di lavoro, si effettuano la **conversione da terminale locale a terminale standard virtuale** e viceversa (vedi sessione OSI)

**Sia telnet, sia ftp** sono basati su questo modello di terminale detto **standard NVT** (ossia **Network Virtual Terminal**)



# IMPLEMENTAZIONE STANDARD NVT

Definizione ed uso del **formato NVT (Network Virtual Terminal)**

Allo startup, **NVT prevede uso di caratteri con rappresentazione**

**7 bit USASCII (caratteri normali 1 byte con bit alto a 0, 8° bit reset)**

**in seguito la rappresentazione della sessione NVT può anche cambiare, se negoziata opportunamente tra i pari**

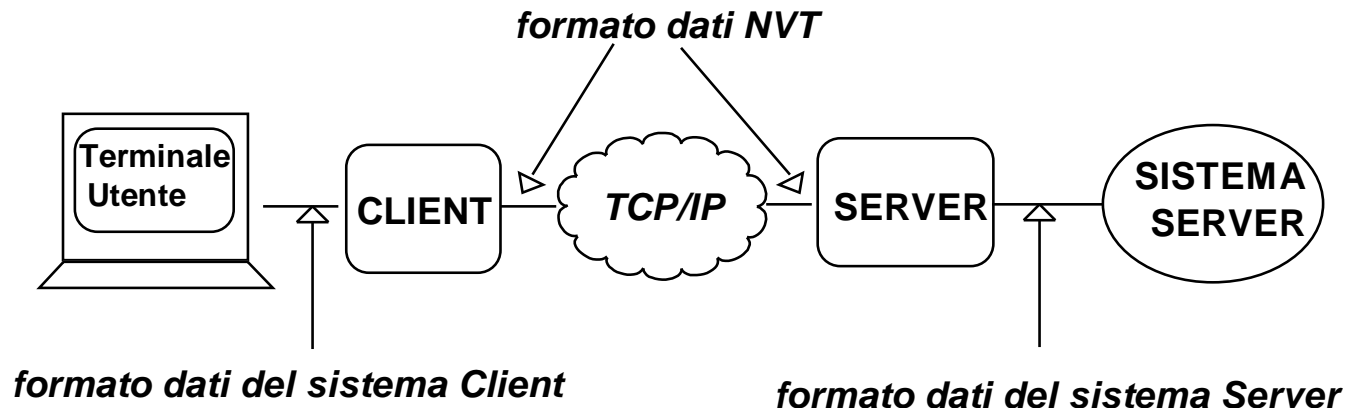
**In generale, per ogni informazione scambiata**

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

**I dati viaggiano in formato standard**

**Server** trasforma dal formato NVT al formato locale

(e viceversa al ritorno): due **trasformazioni** per ogni dato comunicato



# CARATTERI DI CONTROLLO NVT, USASCII

---

## UNICA CONNESSIONE (CONTROLLO IN BANDA)

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

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
IAC	255	CODICE DI COMANDO (INTERPRET A COMMAND)
altri codici	—	NESSUNA OPERAZIONE

# FUNZIONI DI CONTROLLO NVT

---

Codificati con bit più significativo a 1

SEGNALE	VALORE	SIGNIFICATO
IP	244	INTERRUZIONE DEL PROCESSO
AO	245	ABORT IN USCITA (SCARTA I CONTENUTI DEI BUFFER)
AYT	246	CI SEI? (TEST DELLA PRESENZA DEL SERVER)
EC	247	CANCELLA IL PRECEDENTE CARATTERE
EL	248	CANCELLA LA CORRENTE LINEA
SYNC	249	SINCRONIZZAZIONE
BRK	250	SOSPENSIONE TEMPORANEA (ATTESA DI UN SEGNALE)
WILL	251	SI PROPONE AL PARI DI ATTUARE UNA AZIONE
WONT	252	SI PROPONE AL PARI DI NON ATTUARE UNA AZIONE
DO	253	SI ACCETTA DI ATTUARE UNA AZIONE
DONT	254	NON SI ACCETTA DI ATTUARE UNA AZIONE

# NEGOZIAZIONE

---

## NEGOZIAZIONE del TERMINALE (cfr. OSI presentazione)

Possibilità di **negoziare** la connessione,

sia alla **inizializzazione**

sia **successivamente** per selezionare le opzioni telnet

**(comunicazione half- full- duplex,**

**determinare il tipo di terminale,**

**codifica 7-8 bit)**

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

# NEGOZIAZIONE

---

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

<code>will X</code>	<code>will you agree to let me use option X?</code>
<code>won't X</code>	<code>I won't start using option X</code>
<code>do X</code>	<code>I do agree to let you use option X</code>
<code>don't X</code>	<code>I don't agree to let you use option X</code>

**Chi ha la iniziativa determina l'accordo richiesto per potere lavorare correttamente sulla connessione stabilita**

**WILL ECHO**

**DO** "terminal type" XYZ

**DO ECHO**

**WILL** "terminal type" XYZ

e si va avanti... in modo sorgente

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

# OPERATIVITÀ TELNET

---

## MODI di LAVORO

### half-duplex

**one char at a time** (echo fatto sempre dal server) →  
un carattere alla volta, con problemi di overhead

**one line at a time** buffering → problemi di ritardo nei dati **linemode**

## Uso di dati urgenti

Lo stream impiega i **dati urgenti** sulla connessione stessa per ovviare al caso di stream di dati pieno

### Riconosciuti come comandi

#### flush

→ scarta tutto

#### no client flow control

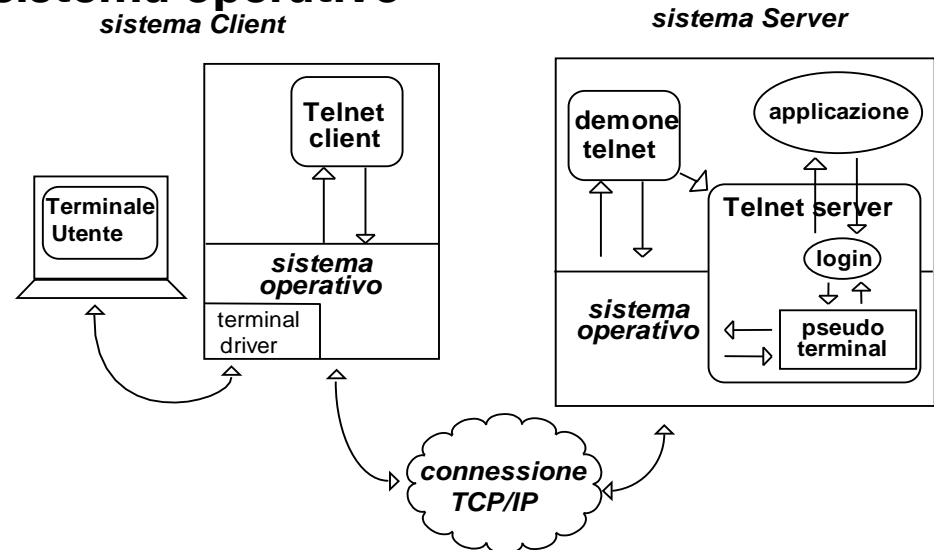
→ il cliente non fa più controllo di flusso

#### client flow control sliding window

→ cambiamento della dimensione e controllo del cliente

# IMPLEMENTAZIONE A PSEUDO-TERMINAL

## Pseudo-terminal come funzione del sistema operativo



Questa soluzione implementativa permette di avere a livello di applicazione **un dispositivo (finto ossia pseudo)** che si comporta **come un dispositivo reale** del sistema operativo ma che facilita di molto l'I/O da rete e la interazione locale

Se il sistema operativo ha **l'astrazione di pseudoterminale**

**telnet** ⇒ programma applicativo

**Vantaggio** ⇒ **modifiche e controllo facili**

**Svantaggio** ⇒ **inefficienza**

# RLOGIN

---

## Servizio di login remoto 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**)

**L'utente riconosciuto passa da una macchina ad un'altra senza qualificarsi e fornire 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**
- Utilizza **una connessione TCP/IP** e **più processi** (**due** per parte)
- Lavora solo **un carattere alla volta** (**Nagle** on e overhead)

**rlogin** molto più snello di telnet ma con prestazioni limitate e non ottimizzate

codice di **migliaia di linee vs. decine di migliaia (di telnet)**

- **Alla fine, uso di due connessioni e 4 processi**
- **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, e così il **<Ctrl><C>**)

# IMPLEMENTAZIONE

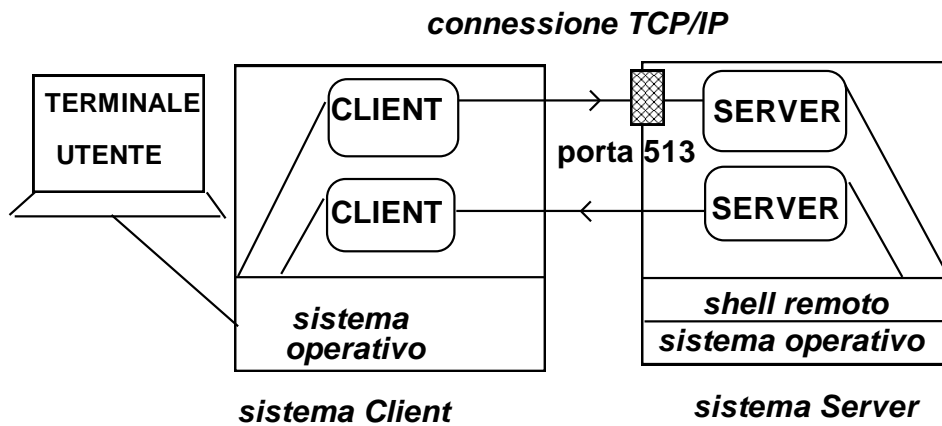
## Client rlogin e Server remoto (server rlogind)

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 dal cliente allo shell remoto
- il figlio gestisce i caratteri in arrivo sul cliente dallo shell remoto

Il server si collega ad uno shell remoto con coppia master-slave di uno pseudoterminale (dopo avere lasciato libero il **demone rlogind**)



Invocazione remota

Anche **rsh**:

- invoca l'interprete (shell) remoto UNIX
- con gli argomenti della linea di comando:

**rsh nodo\_remoto comando**

# TRASFERIMENTO FILE E ACCESSO

---

**ftp** (file transfer protocol)

**Uso di TCP** (affidabile ed orientato alla connessione)

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

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

**Entrambi  
permettono  
copia di file nei  
due versi**

## FILE TRANSFER PROTOCOL ftp

- **Controllo identità**: login e password per controllo accesso
- **Esecuzione a livello applicativo o con accesso interattivo utente** (si può richiedere la lista dei file di un direttorio remoto, o creare un direttorio remoto, etc.)
- **Specifica differenziata del formato dei dati** (rappresentazione): file di tipo testo o binario

# OPERAZIONI DI TRASFERIMENTO FILE

---

Comandi di **trasferimento file**:

1. **put local-file [remote-file]**

→ memorizza un **file locale sulla macchina remota**

2. **get remote-file [local-file]**

→ trasferisce un **file remoto sul disco locale**

3. **mget** e **mput** utilizzano metacaratteri nei nomi dei file per trasferire gruppi di file

**Altri comandi**:

- comandi di movimento nel file system remoto e locale  
**dir, ls, cd, lcd, ...**
- comandi per le proprietà della sessione  
**ascii** → trasferimento di file ascii e non binario
- comandi di supporto  
**help**

Esistono **nodi server di ftp** che sono contenitori di informazioni a cui si può accedere "liberamente" → uso di **ftp anonymous** verso i server

Operazioni libere di download di file

# ESEMPIO DI FTP ANONYMOUS

---

```
antonio disi33 ~ 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.disi.unibo.it:antonio):
anonymous
331 Guest login ok, send ident as password.
Password:XXXXXXXXXX
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
ftp> mget pip*
// trasferisce tutti i file del direttorio corrente che fanno match con la wildcard
...
```

# CODIFICA ED OTTIMIZZAZIONI

---

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**

# IMPLEMENTAZIONE FTP

---

## Vari tipi di file riconosciuti

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

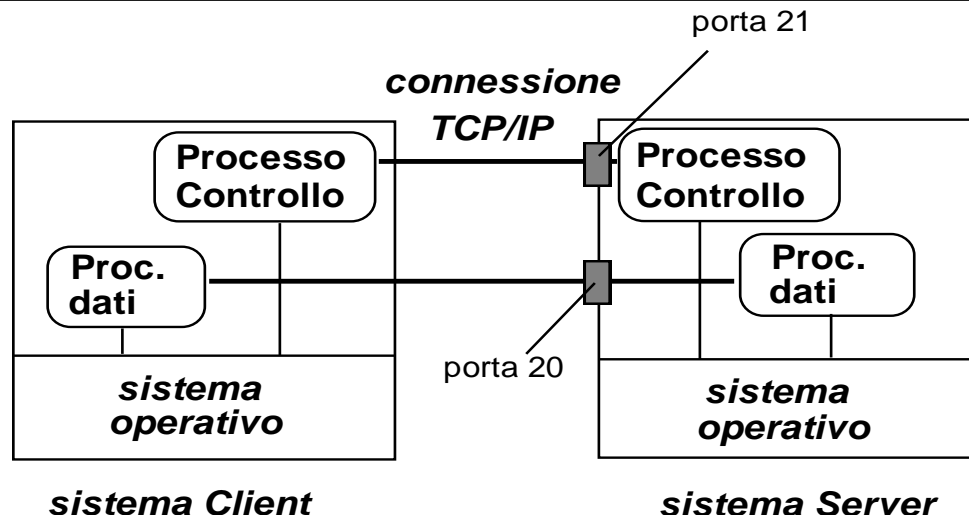
**Accesso concorrente** da parte di più client ad un unico server e uso di **TCP** per le connessioni al server

**Almeno due collegamenti** per ogni client e per ogni server:

- **una connessione di controllo (sempre viva)**
- **una (o più) di dati (ciascuna attiva solo per ogni trasferimento di file)**

Impiegando più processi (**due per nodo**) a parte il demone iniziale

# IMPLEMENTAZIONE FTP: DETTAGLI IMPLEMENTATIVI



## Dettagli implementazione server:

Un processo **master** del server attende connessioni (processo **ftpd**, demone 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 (creato al bisogno)** (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:** **gli endpoint** individuano una connessione ed è sufficiente un endpoint diverso per avere una connessione diversa (almeno un elemento della quadrupla, o porta o nodo)

FTP prevede almeno **due connessioni**:

- **una di controllo persistente** per tutta la sessione
- **una dati** per ogni trasferimento file

## **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)

# IMPEGNO DI RISORSE E QUALITÀ

---

## Client collegamento con server su porta (xxx)

**Servizio sequenziale**, la stessa porta xxx del cliente può essere usata per connessione dati → Il valore di porta passato al server rappresenta una **forma di coordinamento** senza cui il servizio non può funzionare

## Servizi **paralleli** per lo stesso cliente o **porte multiple (yyy)**

Il cliente può indicare una porta aggiuntiva (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?**

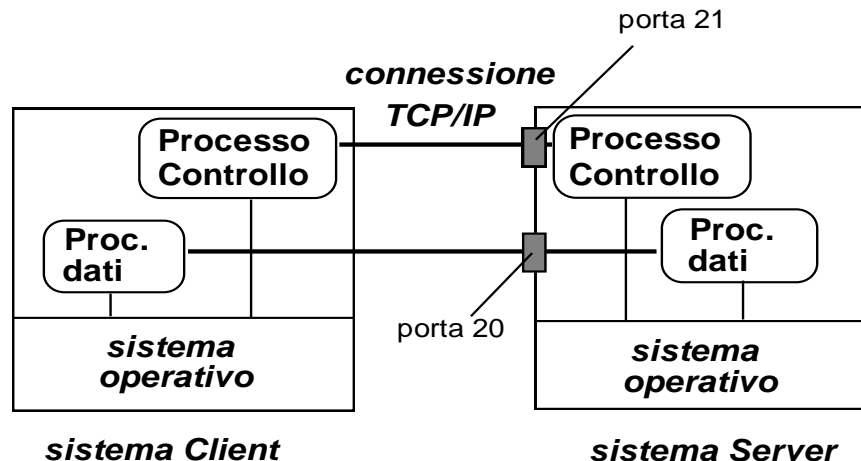
# FTP: INIZIATIVA CONNESSIONI

## Connessione controllo iniziativa del cliente o **cliente attivo**

- la porta di trasferimento comandi lato server è fissa (**21**)
- il cliente specifica una porta (**xxx**) per la connessione e il server ha fatto una listen ed è in attesa di richieste...
- il client esegue la **connect**, conoscendo la porta del server con la sua porta xxx (esegue la **accept**)

## Connessione dati

- la porta di trasferimento lato server è fissa (**20**)
- la porta del cliente differenziata (**xxx/yyy**)
- in questa connessione, creata su comando **get** o **put** del client, chi esegue la connect?



# FTP: CONNESSIONE DATI

---

Sulla connessione dati, chi esegue la connect?

Chi esegue la connect è detto attivo, il pari passivo

## Iniziativa del cliente o cliente attivo

il client esegue la connect, il server deve avere già fatto la listen, ed essersi messo in attesa di richieste del cliente

- In questo caso il cliente deve intervenire garantendo che le azioni siano nell'ordine giusto e coordinarsi con il server (che deve mandare un evento di pronto)

## Iniziativa del server (server attivo) o cliente passivo

il client esegue la listen e fa una **accept** sulla sua porta (**xxx/yyy**), (poi **comanda la get/put**)

il server deve solo fare una **connect** sulla porta del cliente

- In questo caso il cliente può facilmente intervenire producendo le azioni nell'ordine necessario (**SENZA OVERHEAD**)

# CONFRONTO TELNET/FTP

---

**Cosa giustifica la scelta del livello di trasporto di ognuno dei due protocolli?**

ftp e telnet ...

**SERVIZI SINCRONI**

**vs.**

**SERVIZI ASINCRONI**

mail, news ...

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

# LA POSTA ELETTRONICA

---

La posta elettronica (**e-mail**) permette **scambio di messaggi tra utenti**, come nel servizio postale, in **modo asincrono**

Caratteristica fondamentale - servizio **asincrono**: il mittente non aspetta il destinatario (vs. telnet ed ftp) → **spooling**

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

## Mail – Esempio di uso

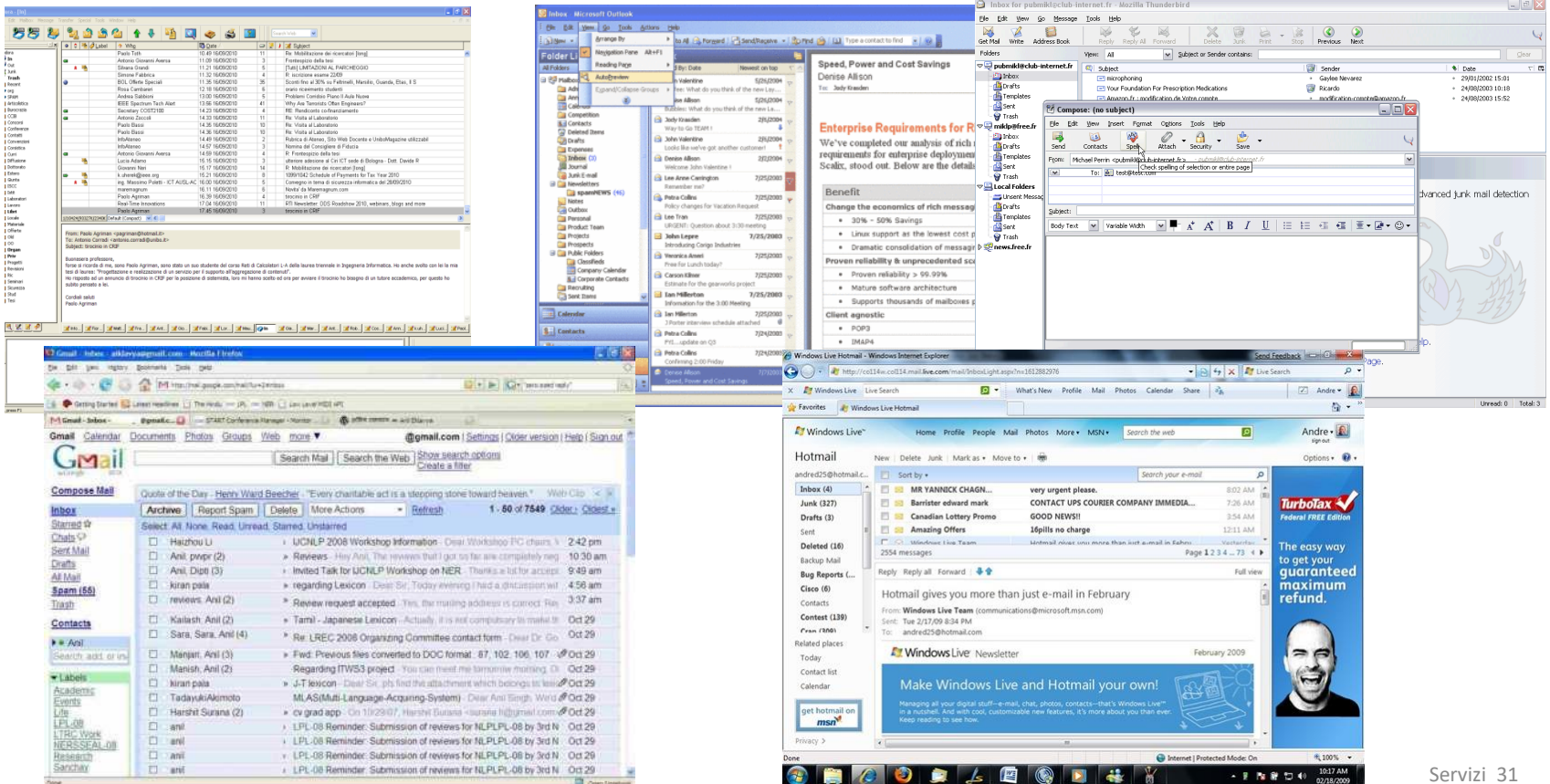
```
antonio deis33 ~ 12 >  
Mail beppe@ing.unibo.it  
Subject: Prova di mail  
Testo del mail  
ctrl-D
```

```
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 della mail
```

# LETTORI DI POSTA ELETTRONICA

Noi interagiamo con la mail attraverso lettori, molto differenziati  
Questi non c'entrano con il protocollo di mail (RFC 822, ...)

Programmi installati come Outlook, Thunderbird, Eudora, o Strumenti  
compatibili Web (gmail, hotmail, ...)



# DATI E PROTOCOLLO

---

**Il formato dei dati iniziale della mail, uno standard dei primi di Internet è molto semplificato ed è legacy**

**I messaggi sono di testo puro e sono divisi in**

**Header: descrittore**

**Corpo: contenuto del messaggio**

**il testo dei messaggi nello standard RFC822 è in formato ASCII puro**

**Lo standard definisce anche le funzioni del supporto che specifica come deve essere trattato il messaggio**

Come sapete possiamo mandare altri formati...

Per estendere il formato del corpo due vie:

- prevedere la possibilità di **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 inserire di messaggi con formati diversi in un unico corpo di un messaggio che il protocollo riconosce automaticamente



# FORMATO DEI DATI

---

## Header

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

## opzionali

**Cc:** → **copia** ai destinatari  
**Bcc:** → **copia nascosta** ai destinatari  
**Replay-To:** → **indirizzo** per la risposta  
**Message-Id:** → **identificatore unico** del messaggio

**Corpo:** il testo dei messaggi è in **formato ASCII**

Con estensioni di formato nel solo corpo

**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

Da caratteri in formato speciale, fino a blocchi che possono contenere binari specializzati

# 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-Transfer-Encoding:**

**7bit (NVT ASCII), 8-bit,  
binary-encoding, ...**

## Content-type

## Subtype

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

# SISTEMA DI NOMI PER INDIRIZZI DI MAIL

---

Destinatario e Mittente come **identificatore IP nodo di destinazione** e **mailbox sul nodo** (nome login)

## **Indirizzi di posta elettronica**

varie possibili forme, con alcune convenzioni

From:antonio.corradi@disi.unibo.it (A. 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

antonio.corradi@disi.unibo.it

acorradi@disi33.disi.unibo.it

antonio@disi33.disi.unibo.it

**Electronic mailing list** anche con destinatari **non locali**

**Possibili problemi di indirizzamento nelle mailing list creando ciclo senza fine**

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

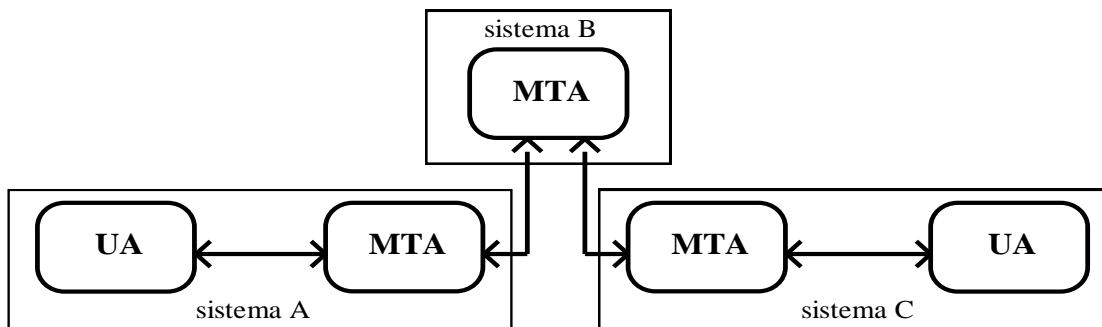
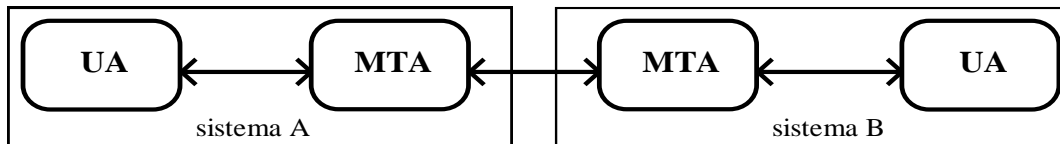
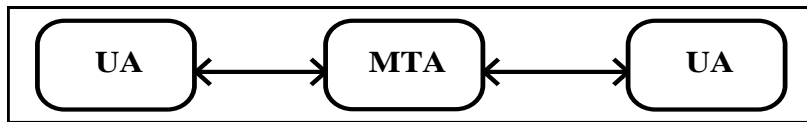
nella mailing list di B, y mappato in x di A

# ARCHITETTURA DEL SERVIZIO DI MAIL

Uso di comunicazioni **punto a punto** attraverso una rete di **User Agent (UA)** e **Mail Transfer Agent (MTA)** che sono **indipendenti tra loro**

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

**Diversi pattern di collegamento possibili**



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

Non si assume la conoscenza completa dei MTA tra loro

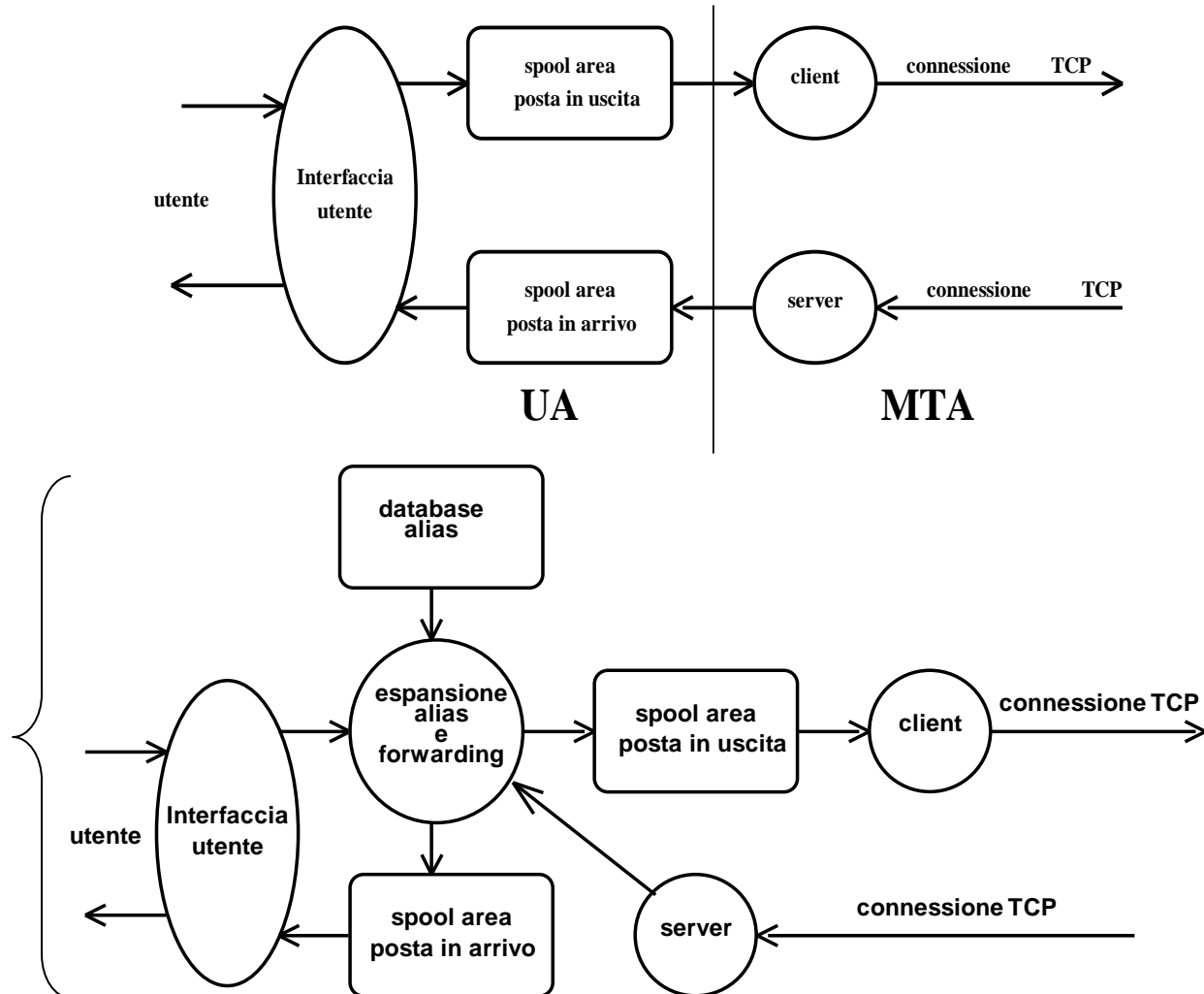
**Il protocollo regola i modi di comunicazione e non la politica di conoscenza delle entità**

# COMPONENTI DEL SERVIZIO DI MAIL

Il processo UA in background diventa il cliente della attività MTA che

- mappa il **nome della destinazione** in indirizzo IP o di intermediario
- tenta la connessione **TCP** con il mail server successivo o di destinazione
- se OK, **copia un messaggio** al successivo passo

Trattamento dei nomi multipli di utente e liste



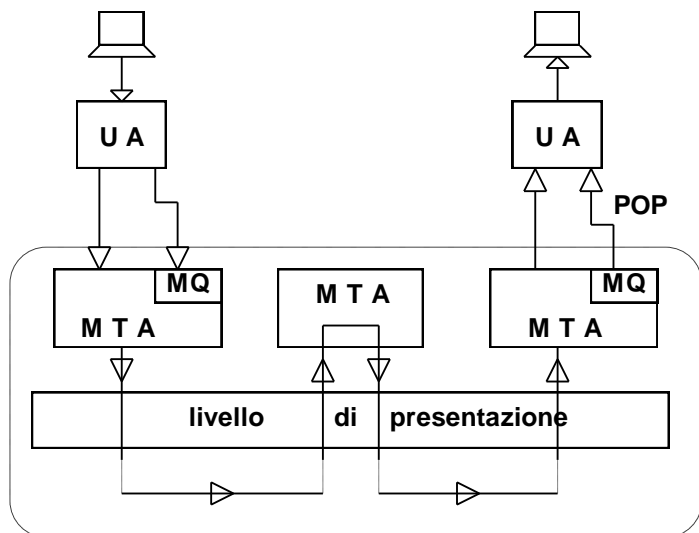
# NOMI DI MAIL E DOMAIN NAME SYSTEM (DNS)

**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)

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

Il protocollo SMTP usa la porta TCP per gli scambi tra MTA (e tra UA e MTA): porta **25**

Accesso finale ai singoli messaggi da parte dell'utente UA sono invece regolati da diversi strumenti e protocolli:

- **POP** Post Office Protocol
- **IMAP** Internet Mail Access Protocol
- anche **sicuri**, ecc.,

Molti lettori diversi di posta elettronica:

**Mail, mail, elm, eudora, outlook, Web based**

# PROTOCOLLO SMTP

## Simple Mail Transfer Protocol RFC 821

È il **protocollo standard** per il **trasferimento della mail tra mailer** (MTA) che si connettono e scambiano messaggi di posta in chiaro

Scambi di messaggi codificati tra un client ed un server

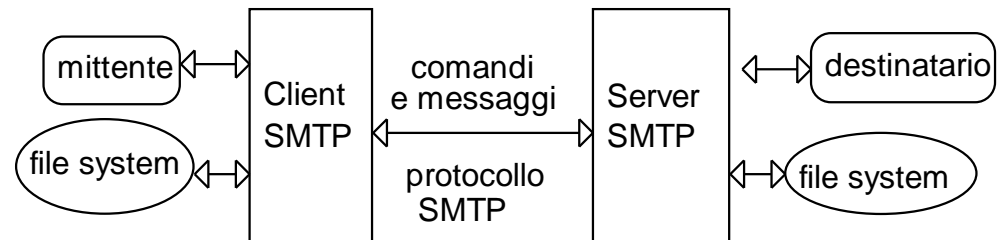
### PROTOCOLLO SMTP

Comandi cliente e risposte server, esempio:

**sender** 'MAIL FROM:' nome mitt.  
**receiver** 'OK'

**sender** 'RCPT TO:' nome dest.  
**receiver** 'OK' abilitato

**sender** 'DATA' corpo messaggio  
**sender** '< cr-lf> < cr-lf>' (fine msg.)  
**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 STANDARD DELLE RISPOSTE

---

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)



# CODIFICA DELLE RISPOSTE

---

La terza cifra specifica più precisamente

## Procedure di SMTP

Procedura di invio come **MAIL TRANSACTION**

→ fatta in modo da **completare la trasmissione**

Se tutto va bene OK

Se problemi → messaggi disordinati e ripetuti  
(azioni di posta **idempotenti** ?)

# ESEMPI DI CODIFICA

---

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>  
Date: 27 Oct 91 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 lo **user name** (VRFY)

**Espansione** di mailing list (EXPN)

**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.

**EXPN** <mailing-list>

S: EXPN Example-People

R: 250-Jon Postel <Postel@USC-ISIF.ARPA>

R: 250-Fred Fone <Fone@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

---

**News** come un **insieme** di **gruppi** di **discussione collettiva**

Ogni gruppo riguarda un **particolare argomento** (**topic**) e permette di partecipare **alla discussione** su tale argomento, scambiando informazioni e facendo domande, ricevendo risposte, ... su insiemi aperti di interessi pubblici

## GERARCHIE PRINCIPALI DI NEWS

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

SOTTOGERARCHIE DI 'comp.unix': admin, aix, amiga, aux, internals, large, misc, programmer, question, etc ...

## STORIA: sviluppo veloce

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

# ARCHITETTURA DEL SERVIZIO DI NEWS

Nodo **client**:

- un **client** di news locale mantiene le news e molti **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 ma non consistenti

**Protocollo news: Network News...**

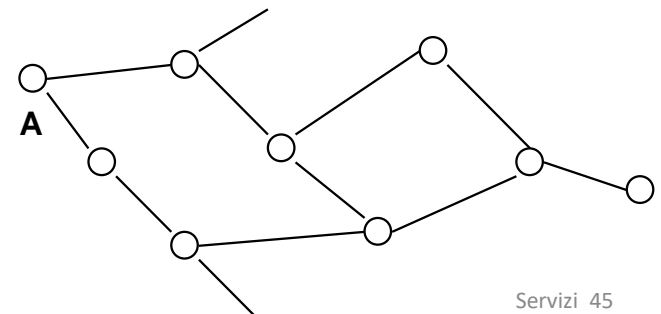
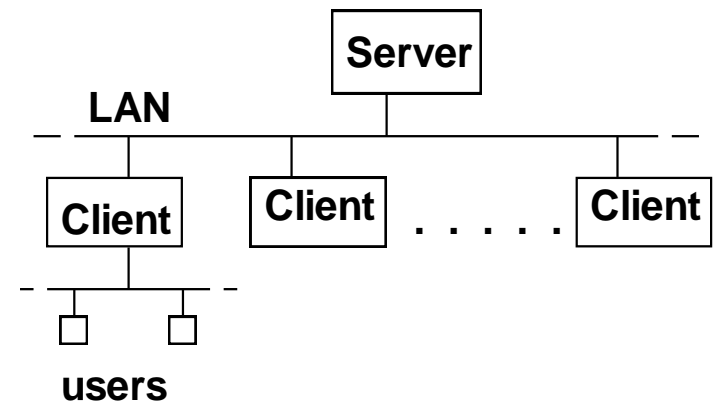
Il protocollo è NNTP (USENET)

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

**COMANDI**: parole composte di caratteri ASCII

**RISPOSTE**: composte di codice numerico di 3 cifre e testo

Gli agenti si coordinano usando la **porta 119**



# PROTOCOLLO NNTP

---

## Protocolli di propagazione a negoziazione (uso well known port 119)

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

Come smtp, così nntp (USENET): **comandi** e **risposte**, e il server restituisce una risposta al comando del client con il risultato dell'azione chiesta

## PROTOCOLLO NNTP (Network News Transfer Protocol)

### Comandi cliente ----- Risposte server

COMANDI: parole composte di caratteri ASCII:

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

RISPOSTE: composte di codice numerico di 3 cifre e testo

#### **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 per le risposte di stato**

# CODIFICA SUCCESSO / INSUCCESSO

---

## Codice numerico di tre cifre

### La prima cifra codifica 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

### La seconda cifra codifica la 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

### Ad esempio

- 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**

# NNTP ESEMPIO

---

## 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	<b>FILE TRANSFER</b>	<b>VIRTUAL TERMINAL</b>
Oggetto	ftp tftp file	telnet rlogin caratteri
Distribuzione Informazioni	punto a punto	punto a punto
Protocollo	NVT	NVT

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

Per **USENET (rete news)**

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

**I SERVIZI COMINCIANO A DELINEARE LA IDEA DI UNA  
INFRASTRUTTURA DI SUPPORTO**

**ANCHE PIÙ DI UNA INFRASTRUTTURA**