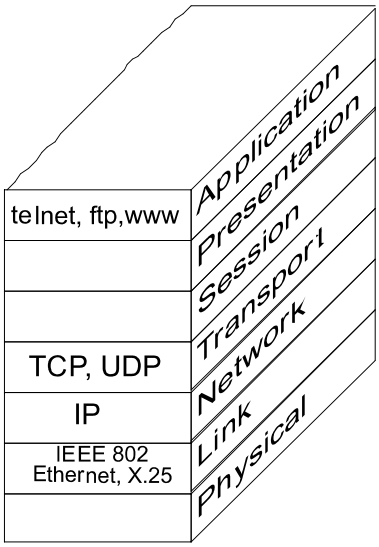
Servizi di UNIX (livello applicazione)



Servizi di tre tipi fondamentali

TERMINALE REMOTO

accesso a nodi remoti

FILE TRANSFER

possibilit 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 pi generali _ **ftp**, **telnet**, **mail**, ...

Propriet fondamentali

Trasparenza (o meno)
Modelli Cliente/Servitore (evoluzioni)
Standardizzazione (evoluzioni)

Servizi APPLICATIVI di un SO

a livello di applicazione per sistemi UNIX: protocolli

Virtual Terminal Procotol: **telnet**

File Transfer Procotol: **ftp**

Trivial File Transfer Procotol: **tftp**

Simple Mail Transfer Procotol: **smtp**

Network News System Transfer Protocol: **nntp**

Line Printer Daemon Procotol: **lpd**

Domain Name System: **dns**

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

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

telnet e rlogin

Per gestire l'eterogeneit di sistemi operativi ed hardware:
Il terminale locale diventa un terminale del sistema remoto

rlogin per i sistemi UNIX (**r**emote **login**)
telnet standard in TCP/IP internet

Protocollo telnet

telnet costruito su TCP/IP
connessione TCP con **server** per accesso remoto
possibilit di aggancio a server qualunque

Caratteristiche:

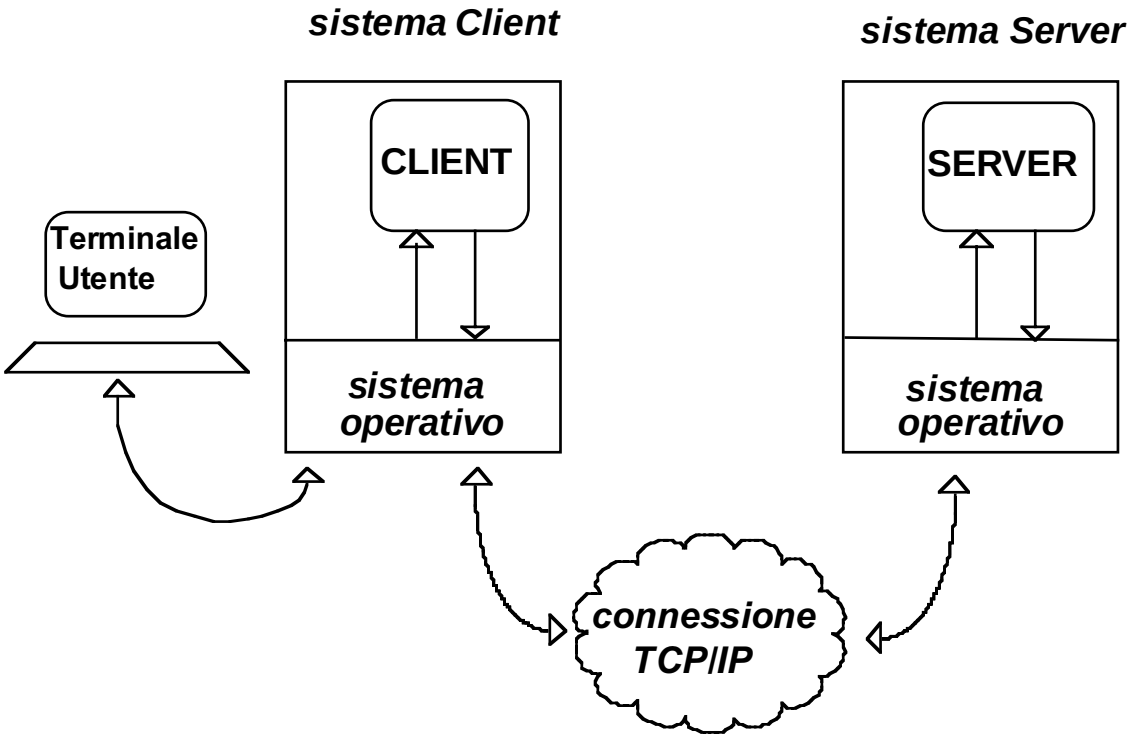
- **gestione eterogeneit** tramite interfaccia standard (NVT)

- **Client** e **Server** negoziano le opzioni del collegamento
  (es., ASCII a 7 bit o a 8 bit)

- comunicazione **simmetrica**

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 con nome logico host o indirizzo fisico IP
anche il *numero di porta*

telnet [host [port]]

Esempio:

```
telnet 137.204.57.33  (telnet lia02.deis.unibo.it)
username:beppe
password:*******
```

TERMINALE VIRTUALE

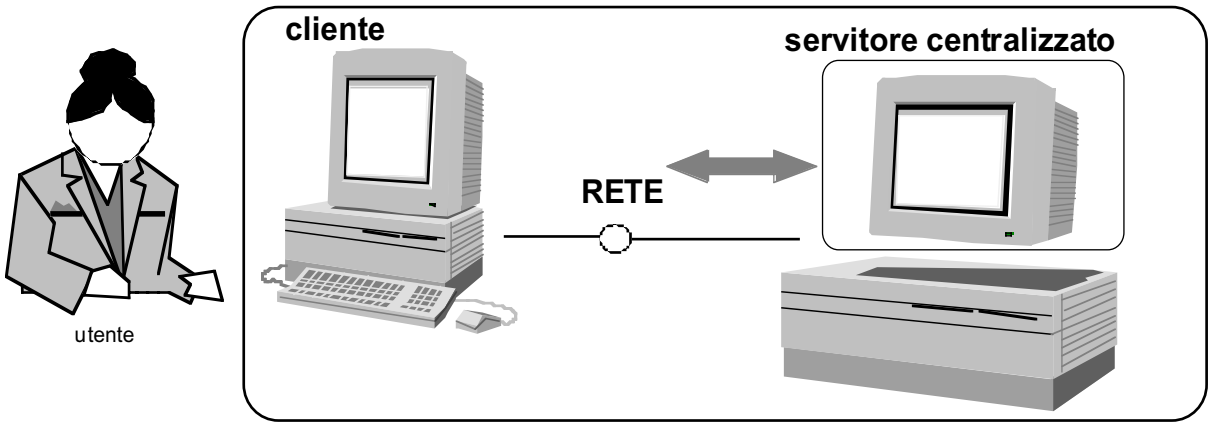
esigenza sentita in generale nelle reti e in particolare nel internetworking

problema: mancanza di standardizzazione 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)

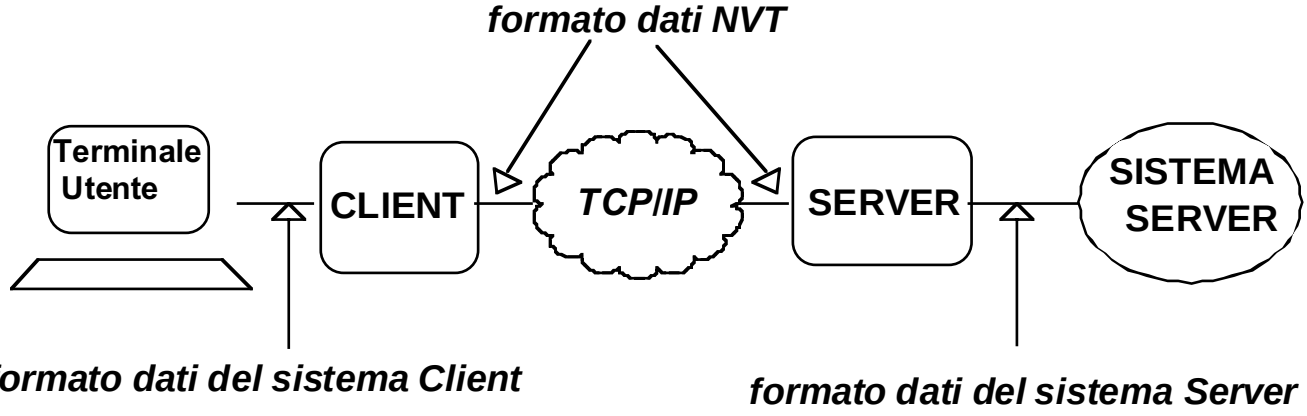
soluzione:
definizione di un **terminale virtuale**

La rete considera un unico tipo di terminale.
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 **NVT** (ossia **N**etwork **V**irtual **T**erminal)



Implementazione

Uso di formato **NVT (Network Virtual Terminal)**
All'inizio NVT prevede uso di rappresentazione a 7 bit USASCII
(*sequenze di comando* in byte con codifiche alte, 8º bit settato,)



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

Server dal formato NVT nel formato remoto
viceversa al ritorno

NEGOZIAZIONE
Possibilit 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**, usa 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 che richiede la terminazione del programma

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

Funzioni di controllo NVT

(codificati con bit pi significativo a 1)

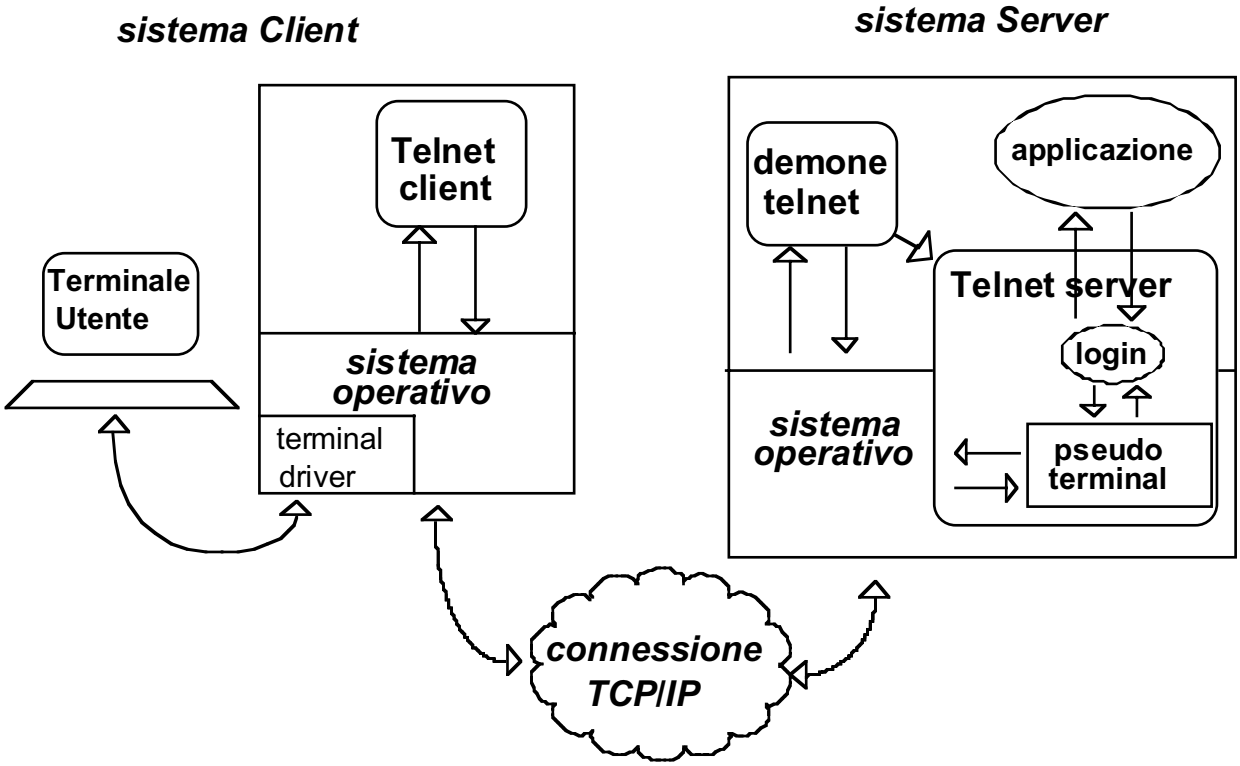
SEGNALE	SIGNIFICATO
IP	INTERRUZIONE DEL PROCESSO
AO	ABORT IN USCITA (SCARTA I CONTENUTI DEI BUFFER)
AYT	CI SEI? (TEST DELLA PRESENZA DEL SERVER)
EC	CANCELLA IL PRECEDENTE CARATTERE
EL	CANCELLA LA CORRENTE LINEA
SYNC	SINCRONIZZAZIONE
BRK	SOSPENSIONE TEMPORANEA (ATTESA DI UN SEGNALE)

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

Problema se lo stream dei dati pieno

Uso di **dati urgenti** o fuori banda di TCP (out-of-band signal)

Implementazione



Pseudo-terminal pu 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:beppe
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)

Problemi di **sicurezza** rlogin:
    - nell'uso di .rhosts ed hosts.equiv
    - password in chiaro

Caratteristiche RLOGIN

- utilizzo di una sola connessione TCP/IP
- 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
- flow control: il client rlogin tratta **localmente** i caratteri di controllo del terminale (ctrl-S e ctrl-Q fermano e fan ripartire l'output del terminale) (in modo simile il ctrl-C)

out-of-band signaling 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)

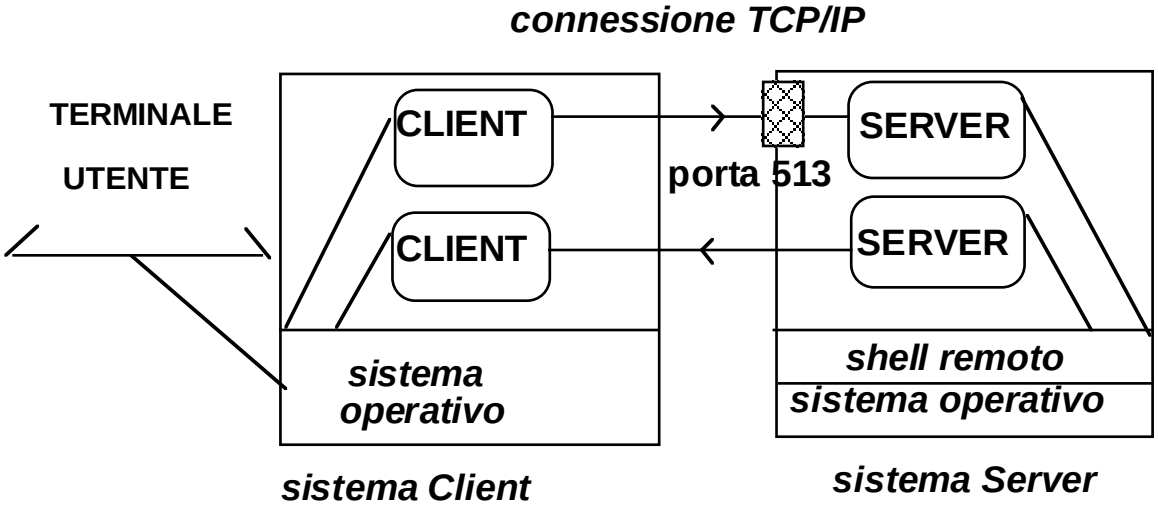
in-band signaling per i comandi dal cilent al server (spedizione dimensione finestra)

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

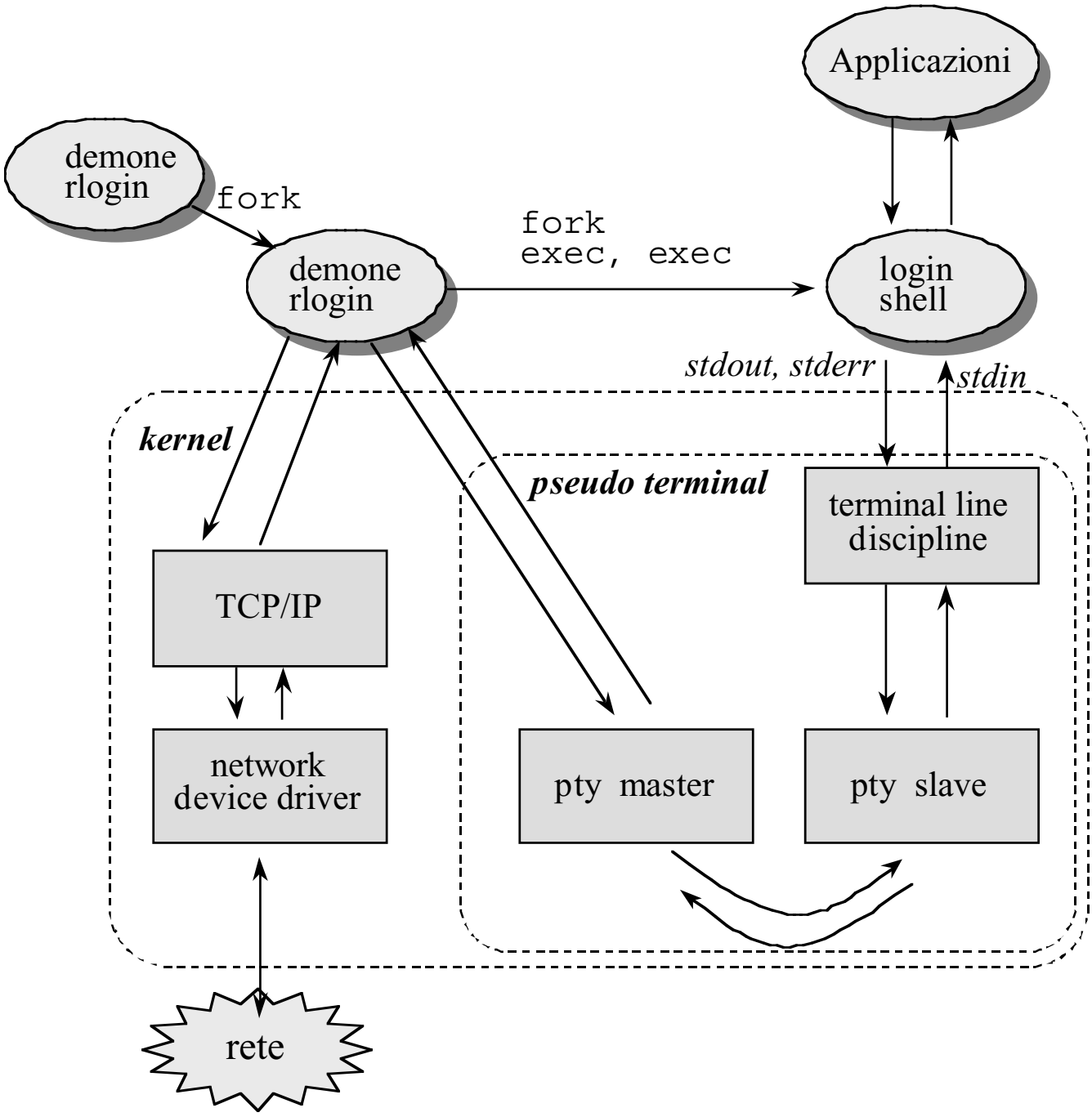
Il server si collega ad uno shell remoto
con coppia master-slave di uno pseudoterminale



Invocazione remota

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

Implementazione e pseudoterminal

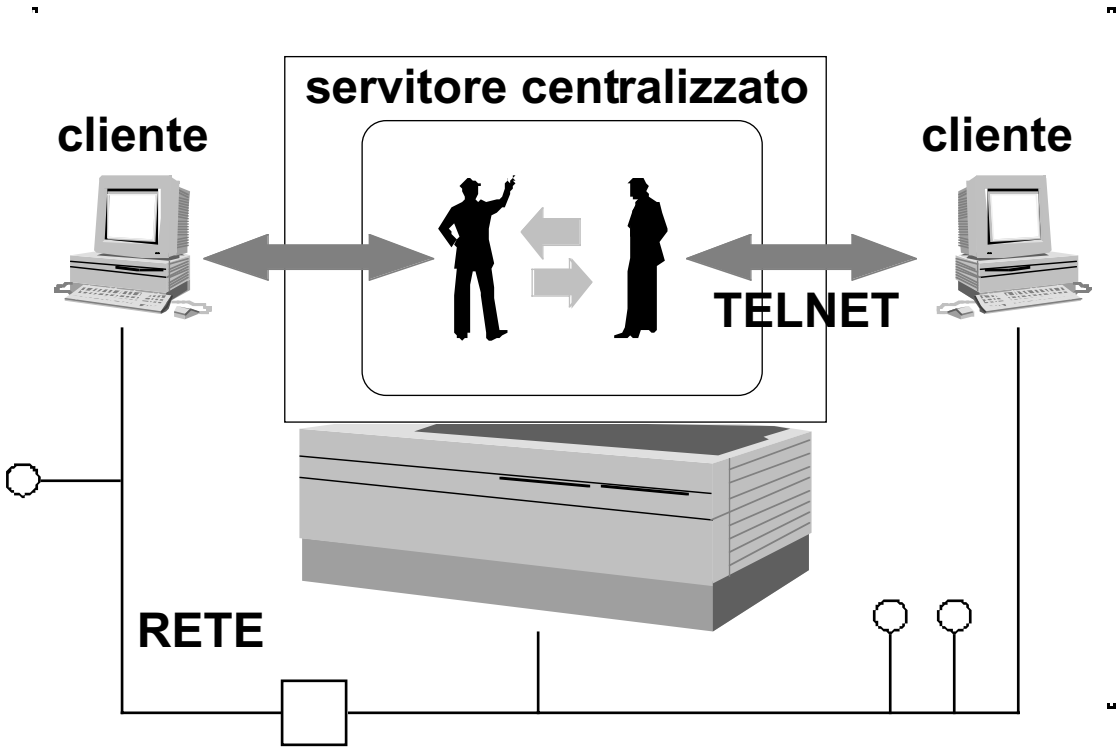


EVOLUZIONI possibili

Modello di comunicazione

client / server avanzato
- parallelo
- stateful
- condivisione dello stato

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



Uso del server come framework di interazione

ACCESSO E TRASFERIMENTO FILE

Uso di TCP (affidabile ed orientato alla connessione)

ftp (file transfer protocol)

tftp(trivial file transfer protocol)

Permettono la copia di file nei due sensi.

Inoltre,

- Eseguito dai programmi applicativi o con accesso **interattivo** (si pu 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 **Identit** (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, ...
```

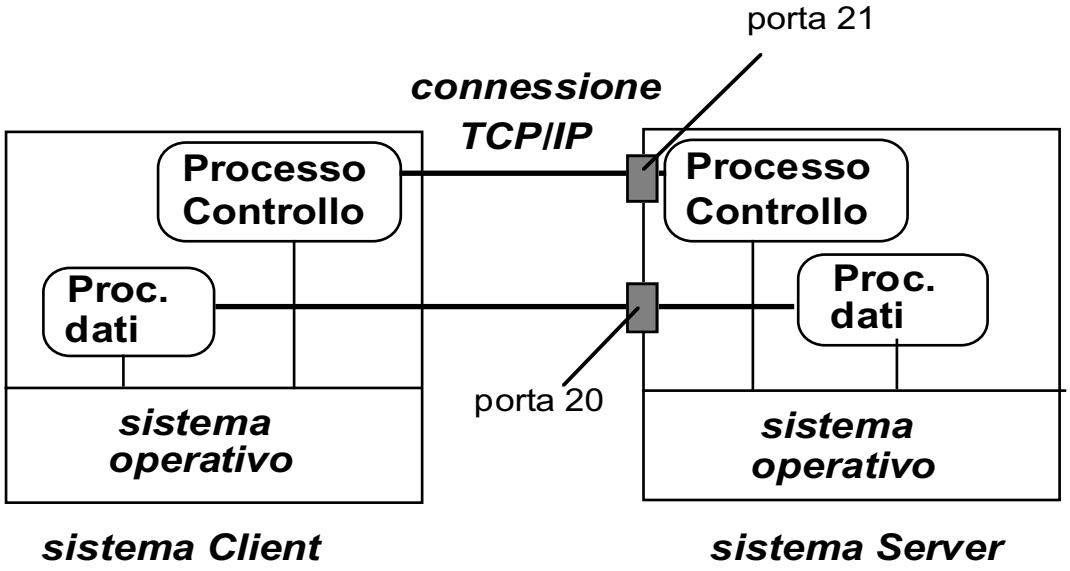
tfpt pi semplice e con meno possibilit (uso di UDP)
Esistono nodi server di ftp che sono contenitori di informazioni a cui si pu accedere "liberamente"

Uso di ftp anonymous verso i server

Implementazione FTP

accesso concorrente da parte di pi client ad un unico server ftp
uso di TCP per la connessione al server

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 si crea uno **slave** per ciascuna connessione

Ogni slave composto da:

- un processo che gestisce il collegamento di controllo con il client
  (persiste per tutta la durata del collegamento)
- un processo per il trasferimento dati
  (possono essere pi di uno all'interno dello stesso collegamento)

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

In certe macchine vi un unico processo incaricato sia del controllo che del trasferimento dati (in generale sempre su diverse connessioni TCP)

Uso di numeri di porta tcp

TCP: entrambi gli estremi individuano una connessione

Nel caso di FTP due connessioni: una dati e una controllo:

Collegamento controllo
la porta di trasferimento lato server fissa (21)
accordo sulla porta dalla parte del cliente (xxx)

Collegamento dati
la porta di trasferimento lato server fissa (20)
porta da parte del cliente (yyy)

Client collegamento iniziale con server comunicando la
propria porta (xxx)
Se i servizi sono sequenziali,
la stessa porta xxx del cliente pu essere usata per la connessione dati

I valori di porta passati al server rappresentano una forma di negoziazione senza la quale il servizio pu non andare a buon fine

Quale formato per il passaggio delle informazioni di controllo? => uso di NVT

Confronto telnet ftp

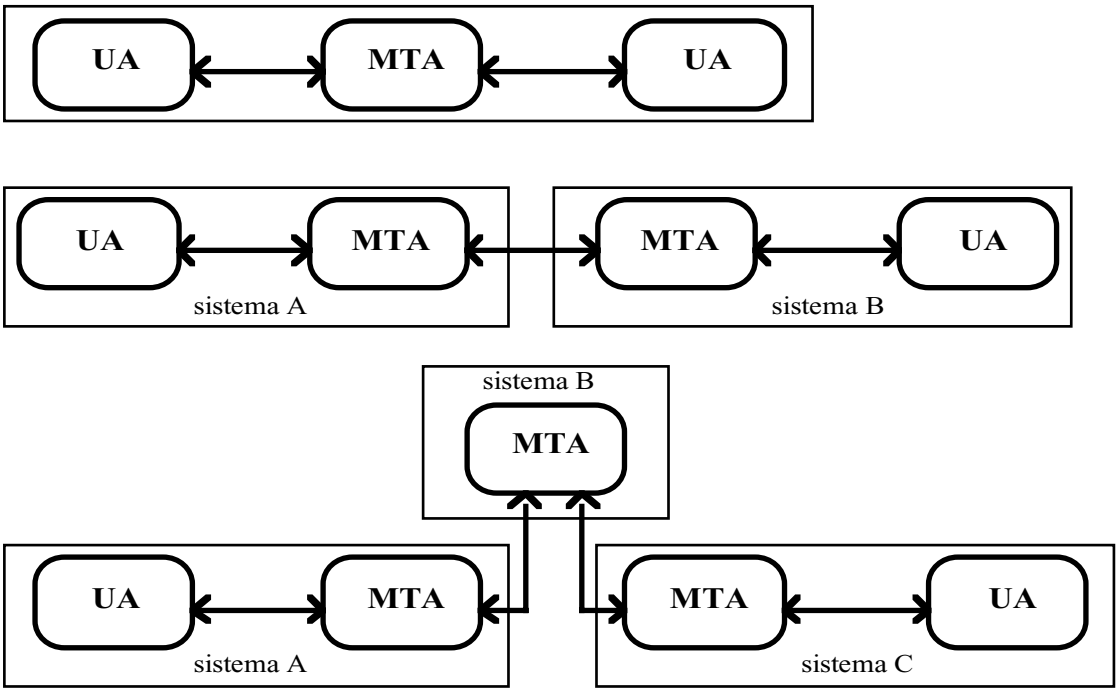
Servizio	FILE TRANSFER ftp tftp	VIRTUAL TERMINAL telnet rlogin
Oggetto	file	caratteri
Distribuzione	punto a punto	punto a punto
Protocollo	NVT	NVT

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

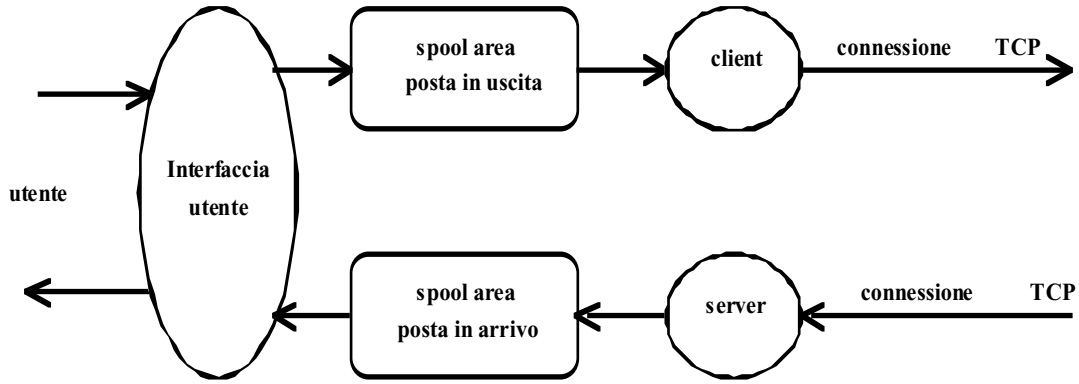
Diversi metodi di collegamento sistemi di posta elettronica



Uso di mailbox come area riservata ad un solo utente

posta elettronica come *sistema **asincrono***
- il mittente non aspetta il destinatario
- spooling

Componenti del servizio di posta elettronica



Un processo in background diventa il cliente

- 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

Protocollo SMTP (Simple Mail Transfer Protocol)

Un processo di trasferimento (in background)
controlla spool area
dopo un certo tempo se invio non OK
torna il messaggio al mittente

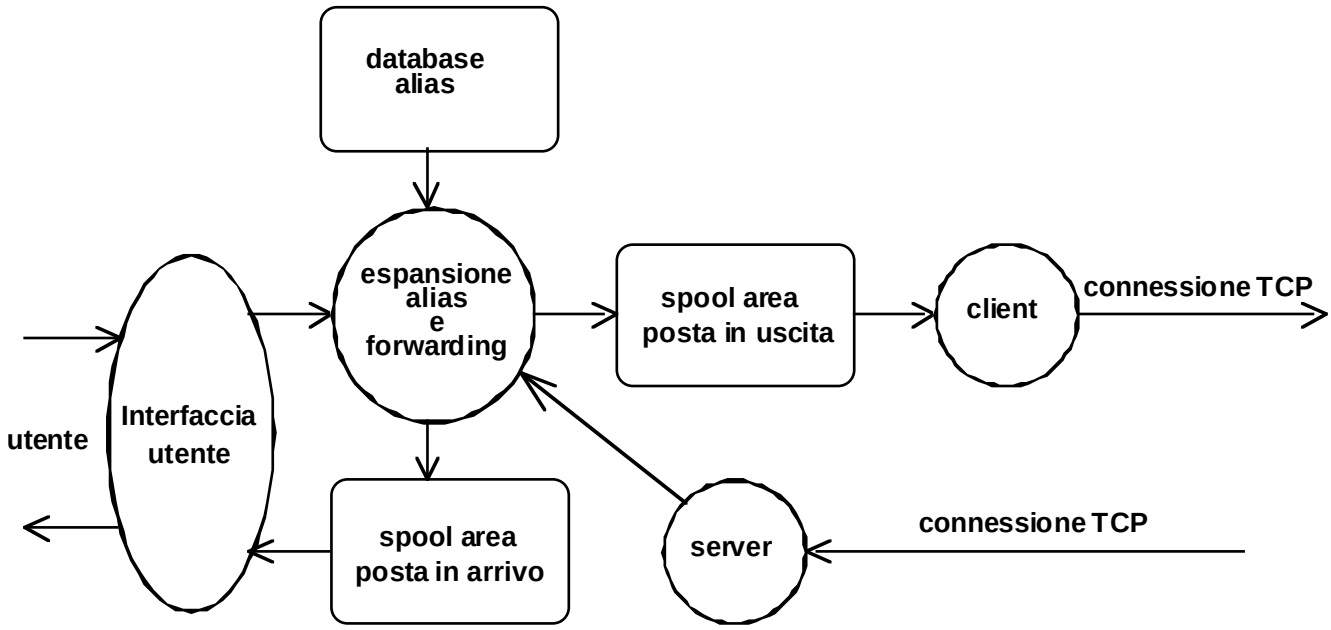
Indirizzi di mail

destinatario come

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

Altri indirizzi

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



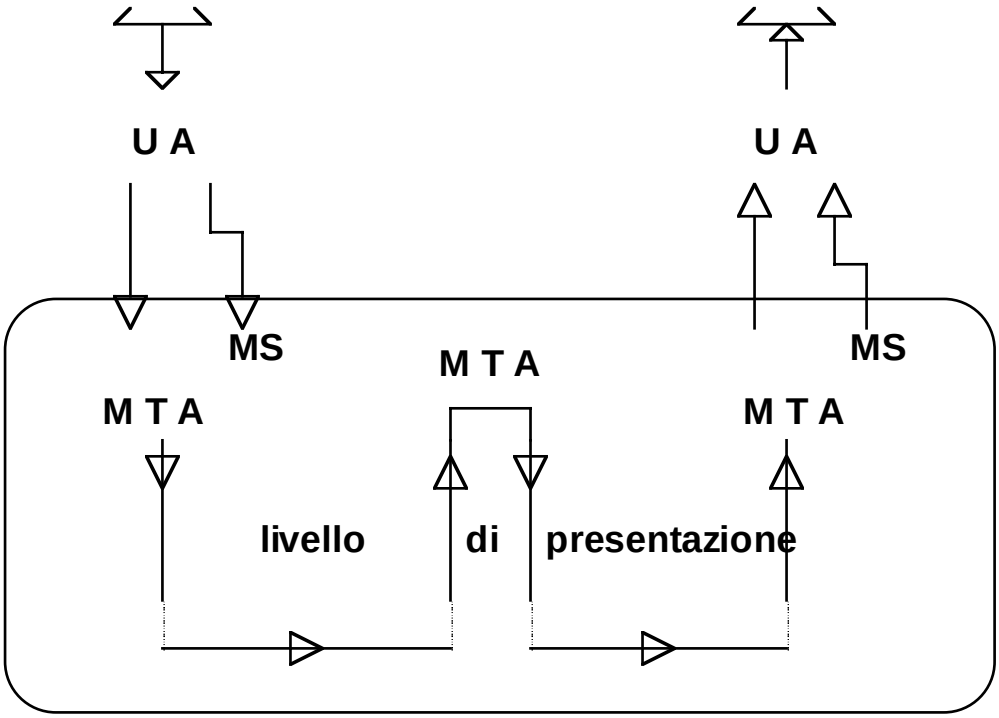
un utente pu avere **pi identificatori di mail**
ma anche
unico identificatore per un gruppo di destinatari

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*

Mail in INTERNET TCP/IP

Servizio di posta elettronica con

1) connessione di tipo **end-to-end diretto** (TCP-IP)
2) uso di mail gateway (macchine intermedie)



Evoluzione dello **standard** di **mail**
con **messaggi multimediali**
e **garanzie di sicurezza**

Formato dei messaggi

From

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

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

opzionali
`Cc:` copia destinatari
`Replay-To:` indirizzo per la risposta
`Message-Id:` Identificatore unico del messaggio

Corpo

il testo dei messaggi in formato ASCII

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

Collegamento con il domain name

mailbox@subdomainN..subdomain1.top-level-domain

pi sottodomini

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

Mail

Esempio di uso:

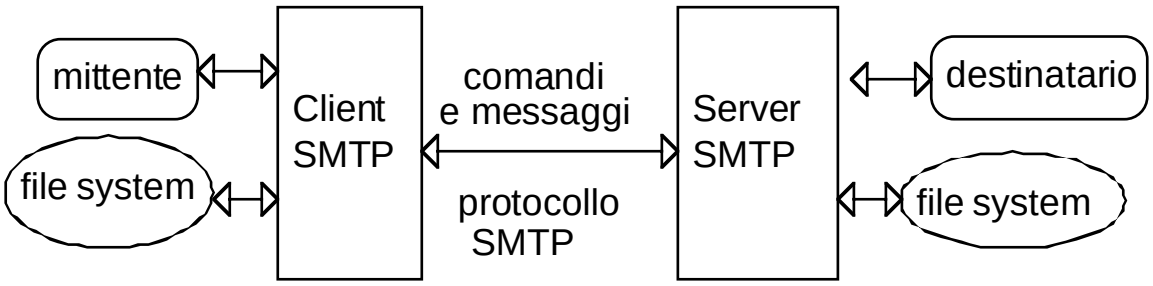
```
cesare deis32 ~ 19 >Mail beppe@deis.unibo.it
Subject:  Prova di mail

testo del mail

ctrl-D

beppe deis ~ 19 > 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 cesare              Sun May 21 16:48   14/296
Prova di mail

& return
Message  1:
From cesare Sun May 21 16:48:38 1995
Received: by deis.unibo.it (4.1/4.7); Sun, 21 May
95 16:48:37 +0200
From: cesare (Cesare Stefanelli)
Subject: Prova di mail
To: Beppe
Date: Sun, 21 May 95 16:48:37 MET DST
X-Mailer: ELM [version 2.3 PL11]
Status: RO

Testo del mail
```

Lettori di posta elettronica:
Mail, mail, elm, eudora, ...

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 pu 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 pi 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 (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 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

E' 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.

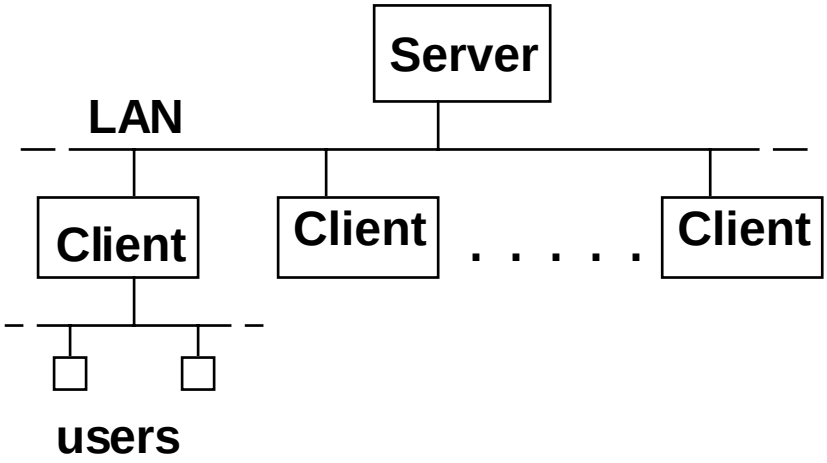
Insiemi aperti di interessi pubblici
GERARCHIE PRINCIPALI DI NEWS

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**:

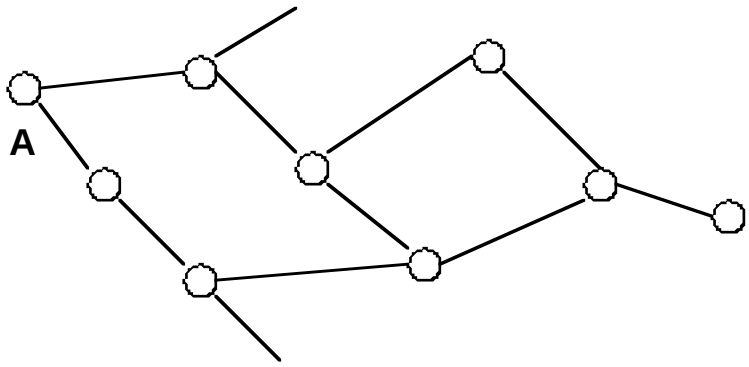
- un **client** di news mantiene le news
- presenza di **lettori** di news

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



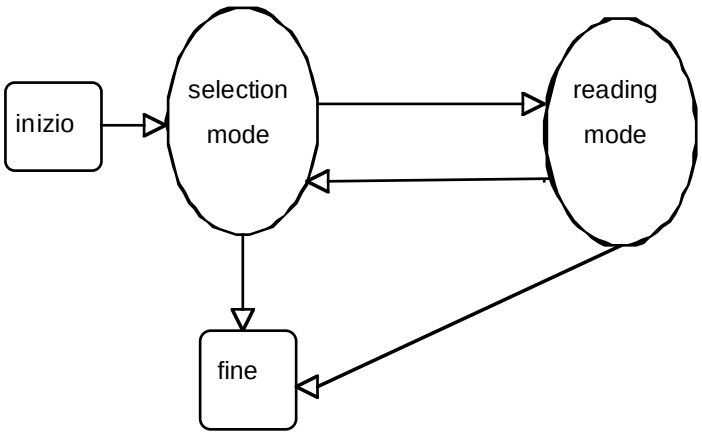
Implementazione

nn Lettore news - interfaccia utente
nntpd Demone protocollo TCP/IP

nn news reader

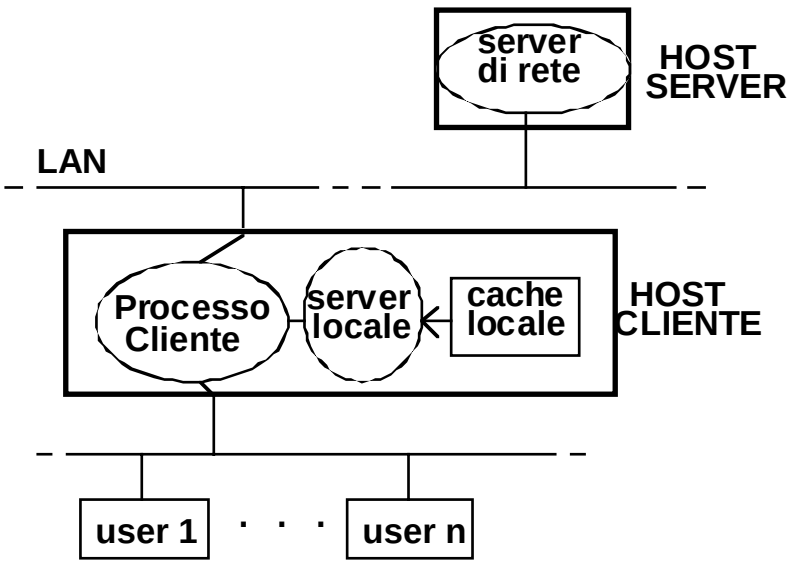
nn [opzioni] [news group | +folder | file]

Stati di nn:
Stato di **selezione** dei gruppi di news
Stato di **lettura** di un gruppo di news selezionato



Stati di funzionamento di nn

Possibile interconnessione



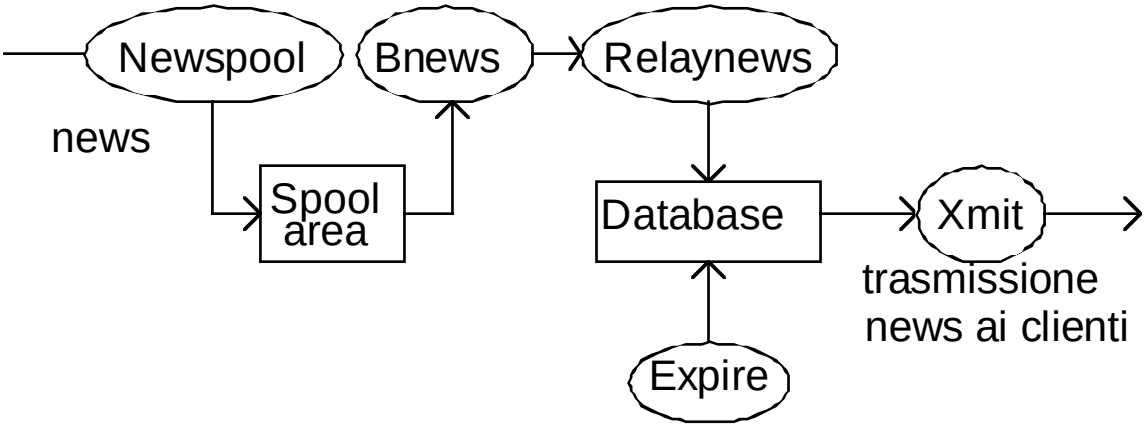
News SERVER

Funzioni del server:

- Accettazione delle news in ingresso
  (newspool, rnews)

- Realizzazione e gestione del database
  (relaynews, expire)

- Invio delle news richieste dai processi cliente
  (xmit)



Protocollo news:

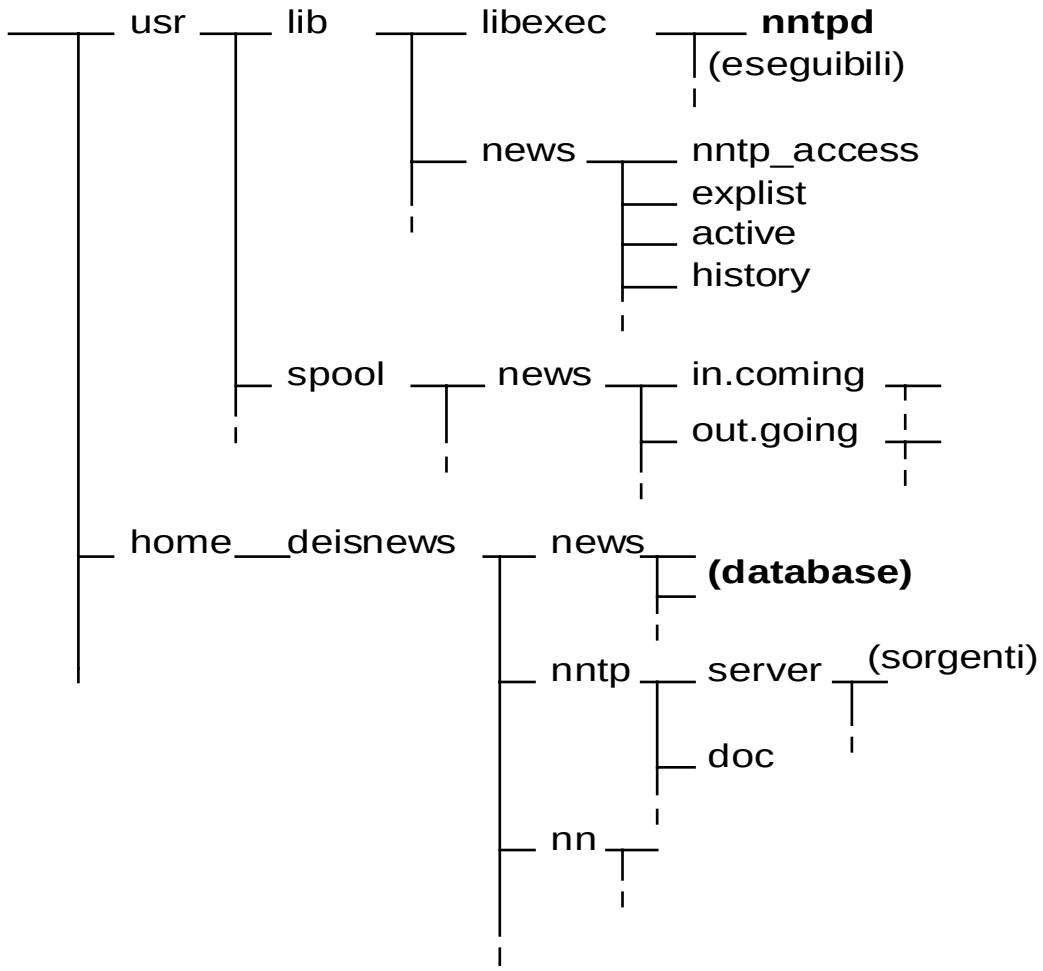
Il protocollo *NNTP (USENET)*

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**

Esempio di file system

in cui si memorizzano le news



explist → eliminazione
history → pr evitare duplicazioni
active → gruppi ricevuti
in.coming → news da memorizzare
out.going → news da esportare

Implementazione

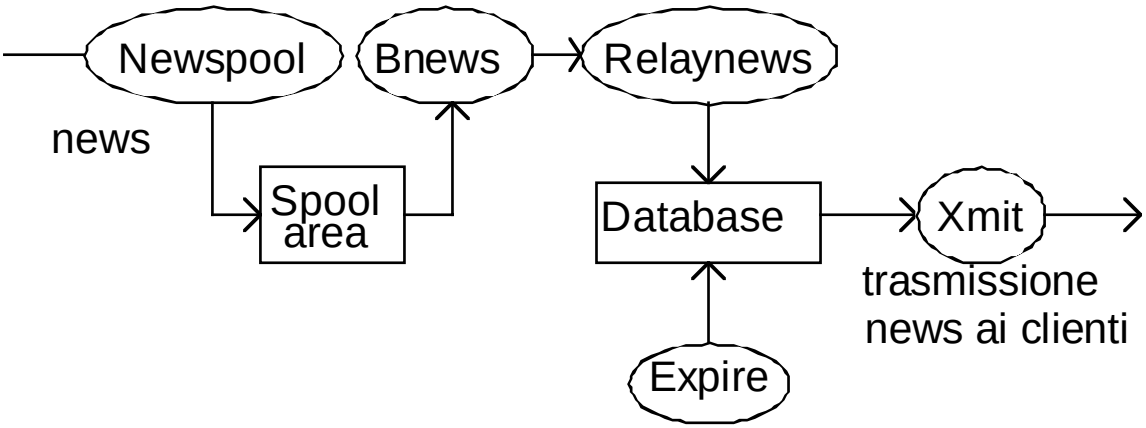
nn Lettore news - interfaccia utente
nntpd Demone protocollo TCP/IP

Funzioni del SERVER nntp
(**porta 119**)

Accettazione delle news in ingresso
(newspool, rnews)

Realizzazione e gestione del database
(relaynews, expire)

Invio delle news richieste dai processi cliente
(xmit)



Protocollo NNTP

Protocolli a negoziazione

Come smtp, cos 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)

pi 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 sintax 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

Confronto mail news

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

Network File System

Sistema Distribuito per **UNIX SUNOS**

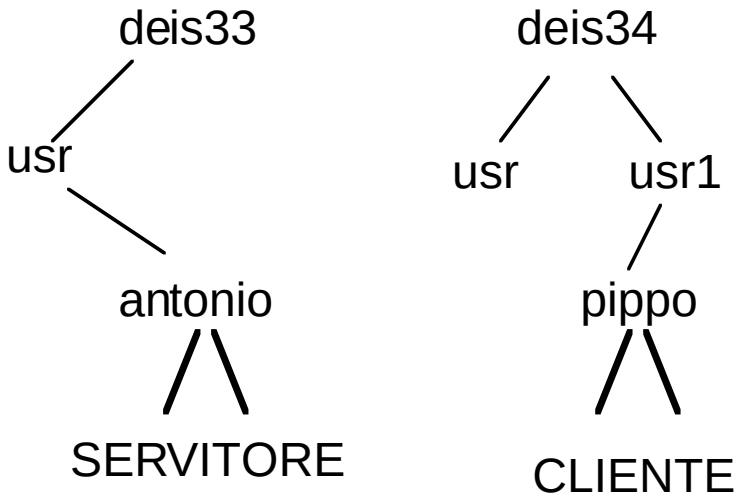
non TRASPARENZA della ALLOCAZIONE

il SUPERUTENTE monta
cio decide di rendere visibile
una parte del file system di un nodo
su un altro file system

P.e. su deis34

```
mount deis33:/usr/antonio /usr1/pippo
```

A questo punto, tutta la gerarchia /usr/antonio visibile dal file system come /usr1/pippo



Da deis34 si vedono i file di deis33
e scompare la visibilit del direttorio locale

TRASPARENZA nella COMUNICAZIONE

a livello utente ===>
l'UTENTE vede i file come se fossero LOCALI

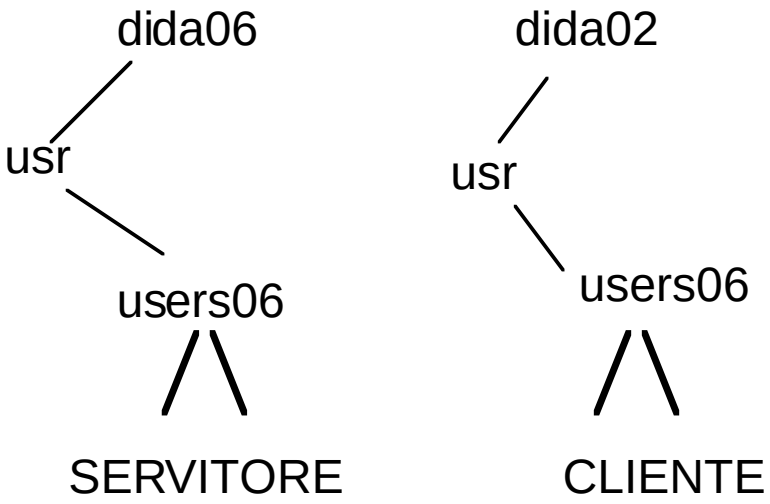
Ogni file system pu contenere parti degli altri
I file relativi sono condivisi tra le diverse macchine

COSTO

Ogni cambiamento fatto su un file ==>
propagato dal file system servitore al cliente

In genere, si tende a mantenere la semantica di UNIX,
cio propagazione di ogni cambiamento agli altri
utenti del file

Miglioramento attraverso l'uso di cache



Altri strumenti (BSD)

iniziali tutte con r

`rcp` copia dei file da remoto a locale e viceversa

`ruptime` verifica la presenza di nodi remoti
`rwho` uso di un demone rwhod che invia pacchetti broadcast (molto costoso)
`rup` presenza di nodi remoti

`rexec` esecuzione su un nodo remoto
uso di un demone **rexecd**

System Management

a livello TCP

`hostname` identit del nodo corrente

`netstat` stato locale TCP
- `-a` comunicazione
- `-i` statistica
- `-r` tabelle di routing
- `-s`
- `-m` buffer
- `-I` monitoring

`ifconfig` esamina l'interfaccia con la rete

`ping` invia pacchetti ICMP al nodo remoto

`trpt` monitoring

`inetd` fa partire un server unico per i servizi locali

relativo ad RPC

`rpcinfo` informazioni sul port mapper

`nfsstat` informazioni su NFS

`spray` invio di RPC per un totale di 100K byte

strumento di monitoring di basso livello

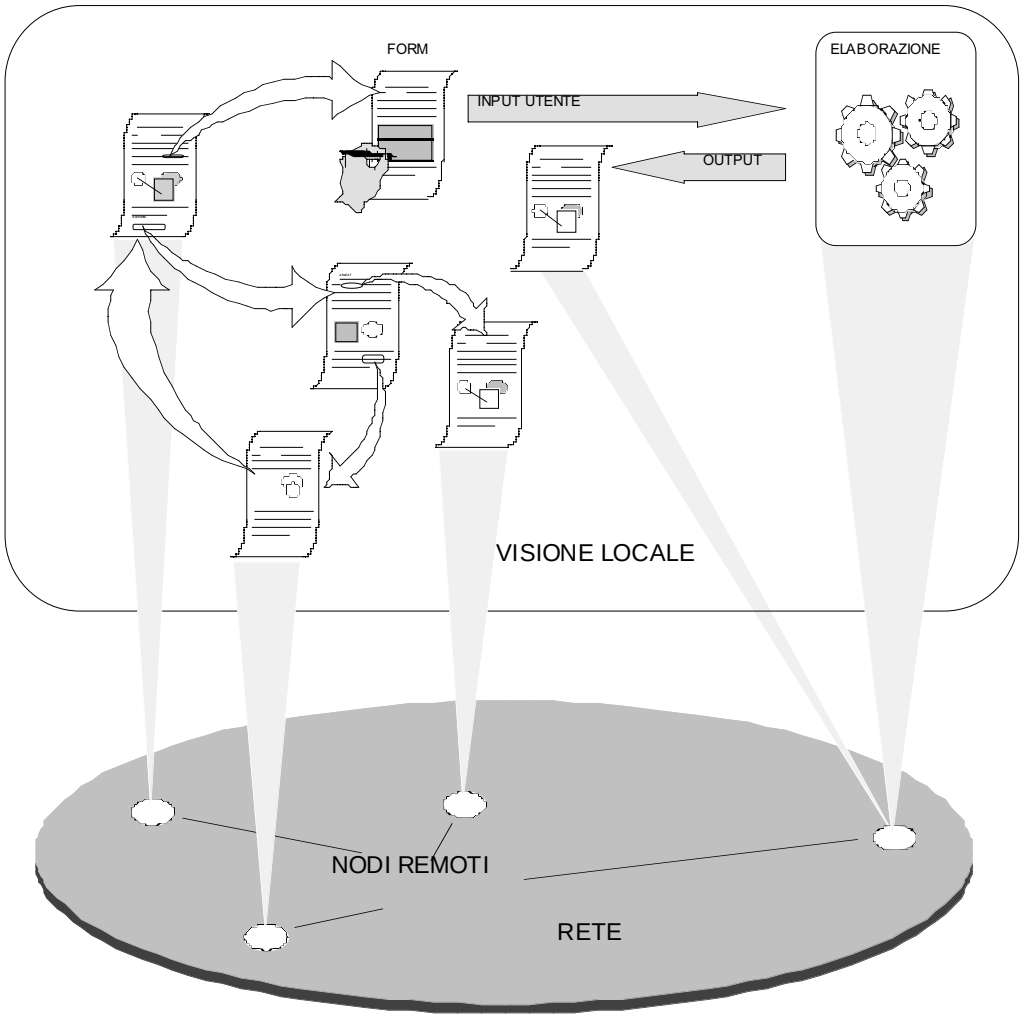
`etherfind` estrae informazioni riguardo ai pacchetti sulla rete (invocabile solo da root)

`traceroute` invio di messaggi fornendo informazioni sul routing

Strumenti trasparenti
unico insieme di informazioni accessibile a tutti i possibili utenti

Gopher

WWW (Explorer, Netscape)
strutturazione ipertestuale delle informazioni (trasparenza della allocazione delle informazioni) e uso di interfacce grafiche (semplicità di utilizzo)



Primo passo (gopher)

strumenti di visualizzazione a **caratteri** di informazioni

Creazione di un unico direttorio che nasconde le informazioni di allocazione

UNICO indice di informazioni per argomento

Allargamento della fascia di utenza

Secondo passo (WWW)

trasparenza allocazione (strutturazione ipertestuale) gestione informazioni di tipo diverso (multimediali)
strumenti e **protocolli** con ***informazioni multimediali***

- evoluzione della mail e altri tool tradizionali
- informazioni trasparenti e multimediali
- protocolli eterogenei

Il secondo approccio *amplia ulteriormente* la fascia di utenza (in modo esponenziale)

ma introduce problemi

*di **banda di occupazione** di risorse*
*di **sicurezza** delle informazioni*
*di **standardizzazione** delle informazioni*
*di **visualizzazione rapida** delle informazioni*

World Wide Web (WWW)

CERN (1989)

Progettodi integrazione in forma ipertestuale risorse esistenti in INTERNET

Scopi

- Trasparenza accesso e allocazione
- Presentazione multimediale
- Interfaccia unica per protocolli diversi (integrazione con gli altri protocolli)
- Modificabilità e condivisione delle informazioni

Ampia scelta di interfacce testuali e grafiche
Possibilità di estensioni sperimentali del sistema

Componenti

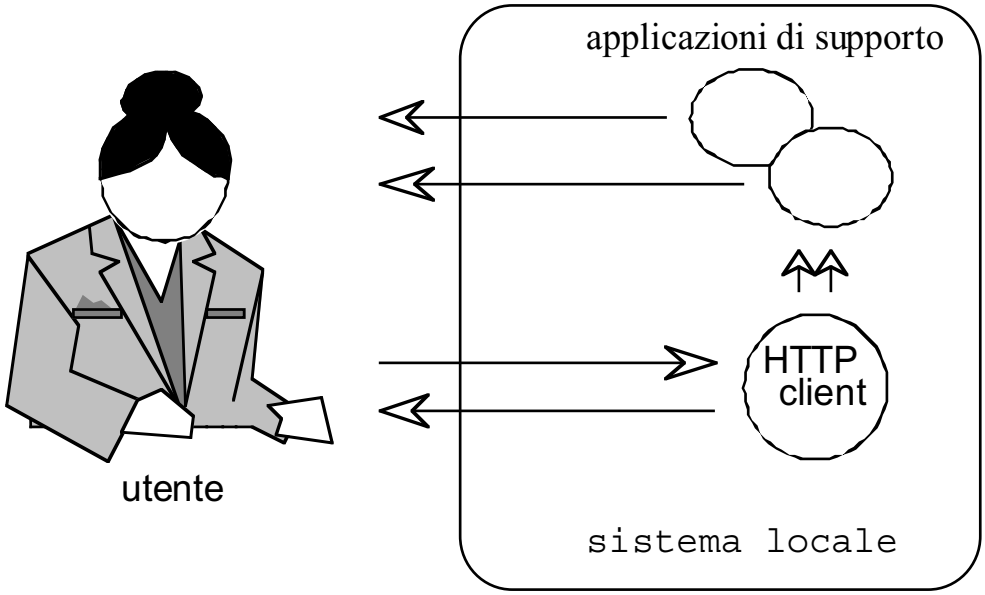
- Browser (presentazione e gestione richieste)
- Server (accesso e invio informazioni)
- Helper applications (particolari presentazioni)
- Applicazioni CGI (esecuzione remota)

Specifiche standard

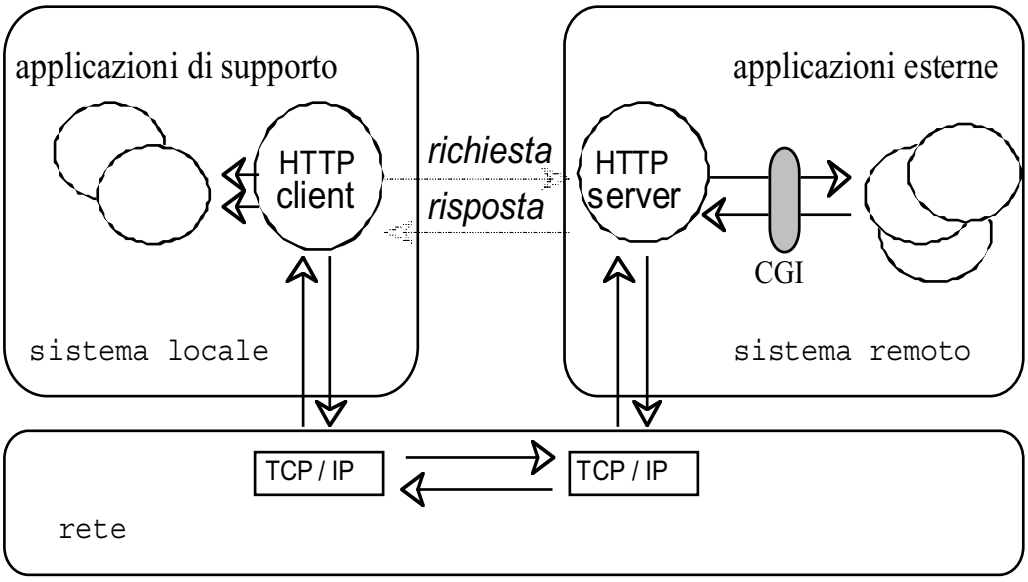
- Sistema di indirizzamento universale URI e URL (Uniform Resource Identifier/Location)
- Protocollo HTTP (HyperText Transfer Protocol)
- Linguaggio HTML (HyperText Markup Language)
- Interfaccia CGI (Common Gateway Interface)

SISTEMA **WWW**

Cliente e sua interazione

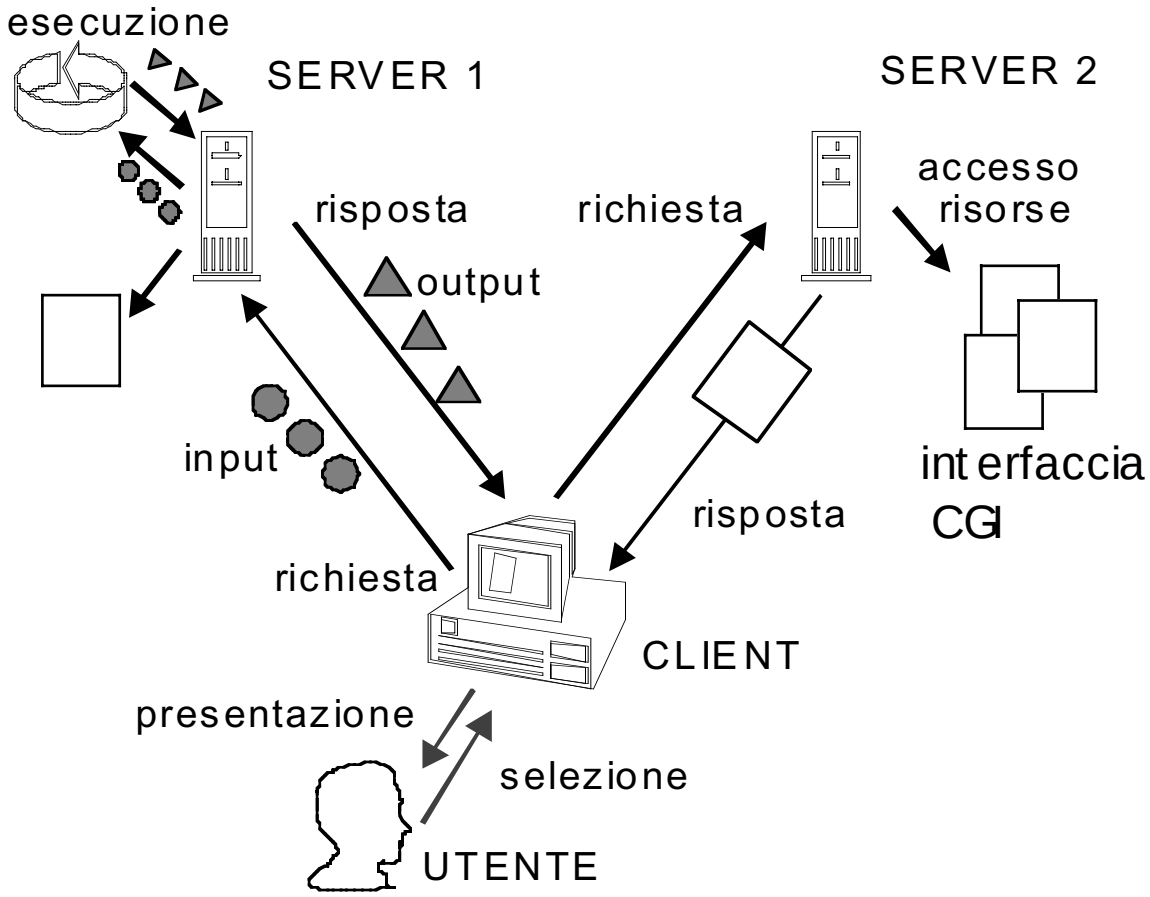


Il Cliente HTTP usa un modo cliente/servitore nei confronti di un server per volta e può anche interagire con risorse locali



Le interazioni cliente/servitore usano il protocollo TCP creando una connessione per ogni informazione da ritrovare (tipica porta fissa **80**)

Implementazione WWW



Modello di comunicazione
client / server
- parallelo
- stateless (ma CGI)

Funzionalità offerte
- Accesso ipertestuale a risorse informative
- Esecuzione applicazioni remote
  - ◊ invio di input da utente
  - ◊ presentazione output

URL Uniform **R**esource **L**ocators

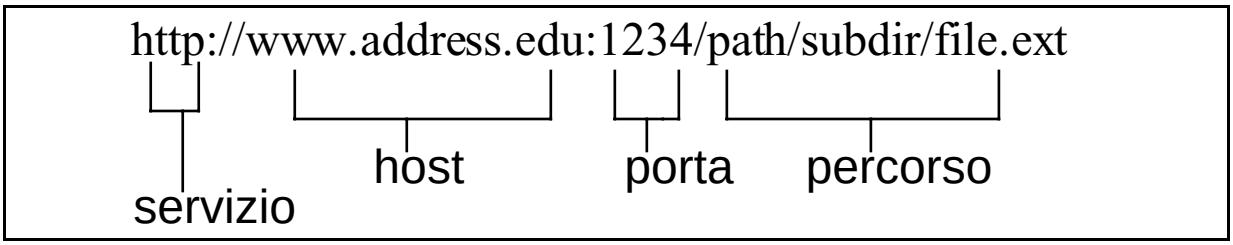
nomi unici per le risorse del sistema
specificati dal cliente per determinare il servitore

*o **URN** (Uniform Resource Names) e servizio di name*
o **Uniform Resource Locators (URL):**

- nodo contenente la risorsa (*documento o dati*)
- protocollo di accesso alla risorsa (*e.g. http, gopher*)
- numero di porta TCP (*porta di default del servizio*)
- localizzazione della risorsa nel server.

```
<protocollo>[://<host>][:<porta>][<percorso>]
```

Sono riconosciuti i servizi internet e relativi protocolli =>
http, gopher, ftp, wais, telnet, news, nntp, e mail



Uso di default per localizzare risorse

Un **URL** può anche determinare un insieme di risorse:
ad esempio versioni multilingue tra cui scegliere

HTTP **H**yper**T**ext **T**ransfer **P**rotocol

protocollo di interfaccia tra cliente e servitore

Uso di TCP e di connessione (porta **80** default)

Caratteristiche HTTP:

- request/response
- one-shot connection
- stateless

Request/response: richiesta e ricezione di dati.
One-shot connection: la connessione TCP è mantenuta solo per il tempo necessario a trasmettere i dati
Stateless: non mantiene nessuna informazione tra una richiesta e la successiva

in genere:

- **richiesta** del cliente con **informazioni** per il **server**
- **risposta** con informazioni dal server

il cliente può determinare una forma di scelta (**negoziazione**) sulle informazioni ed i servizi

```
HTTP-message = Simple-Request          ;HTTP/0.9
   / Simple-Response
   / Full-Request                      ;HTTP/1.0
   / Full-Response
```

NON c'e' stato del server

HTTP request/response

Formato del messaggio di **richiesta**:

indirizzo del server	metodo di richiesta	campi opzionali	path+ nome file (in generale, i dati)

Metodo di richiesta:

- GET richiesta di leggere una pagina web
- HEAD richiesta di leggere l'header di una pagina web (es. contiene data ultima revisione del documento) (metodo usato nelle tecniche di caching nei client)
- PUT richiesta di pubblicare una pagina web
- POST append di dati (di solito HTML form)
- DELETE rimuove una pagina web

Campi opzionali:

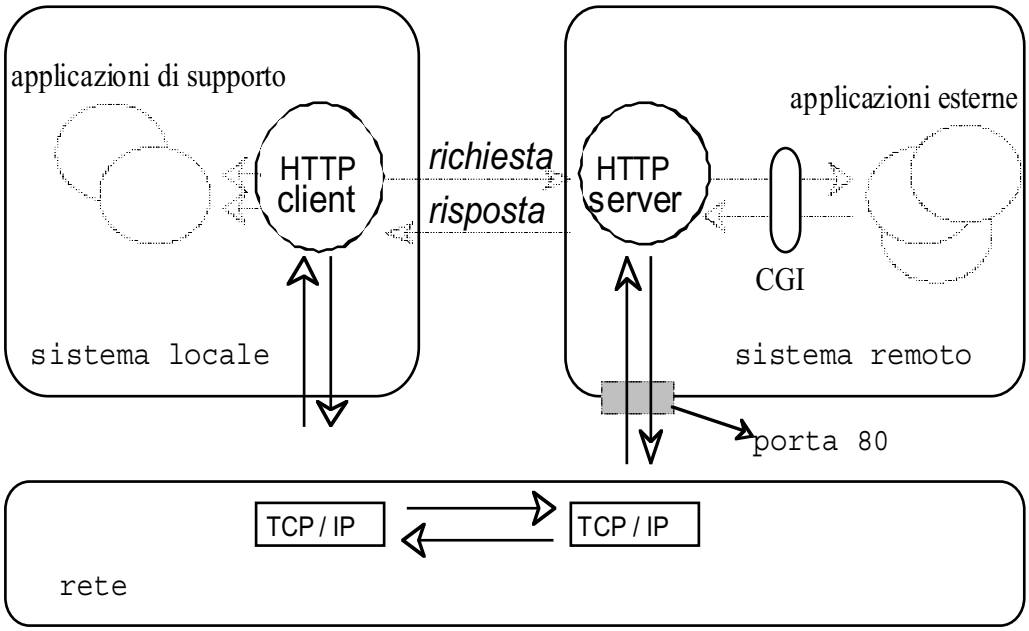
- from: identità dell'utente richiedente (debole forma di autenticazione degli accessi)
- referer: il documento contenente il link che ha portato al documento corrente

Formato del messaggio di **risposta**:

status code	informazioni sull'oggetto	dati

- **status code**: successo o fallimento (es. file not found)
- **informazioni sull'oggetto**: data di modifica, tipo di oggetto. Ogni tipo di oggetto (immagine, HTML, audio) attiva una gestione specifica da parte del client.

HTTP One-shot connection



1. Il browser (http client) riceve un URL dall'utente
2. Il browser richiede al DNS di risolvere il nome della macchina specificata nell'URL
3. Il DNS risponde con l'indirizzo IP
4. Il browser stabilisce una connessione TCP sulla porta 80 dell'indirizzo IP
5. Il browser manda una richiesta con il metodo:
   GET nome_pagina _web
6. Il server risponde mandando la pagina web richiesta

Attenzione: la presenza in una pagina web di altri oggetti (immagini, applets, ecc.) costringe il client ad aprire una **differente connessione** per ognuno degli oggetti necessari.

Il cliente http può eseguire richieste seguendo protocolli differenti (es. ftp) ⇒ porte differenti.

HTML **H**yper**T**ext **M**arkup **L**anguage

HTML è un linguaggio di specifica delle informazioni che deriva da SGML (Standard Generalized Markup Language). E' un **markup language** (TeX, RTF).

I linguaggi markup usano dei **tag** definiti **funzionalmente** per caratterizzare il testo incluso.

tag HTML

testo di tipo header 1: `<H1>testo</H1>`
testo in grassetto: `<STRONG>testo</STRONG>` oppure
`<B>testo</B>`
Visualizzazione dipendente dal browser

link: `<A HREF = "destinazione"> descrizione </A>`

immagini: `<IMG SRC = "myimage.gif">`

applet Java:
```
<APPLET CODE="Hello.class" WIDTH=100 HEIGHT=80>
```

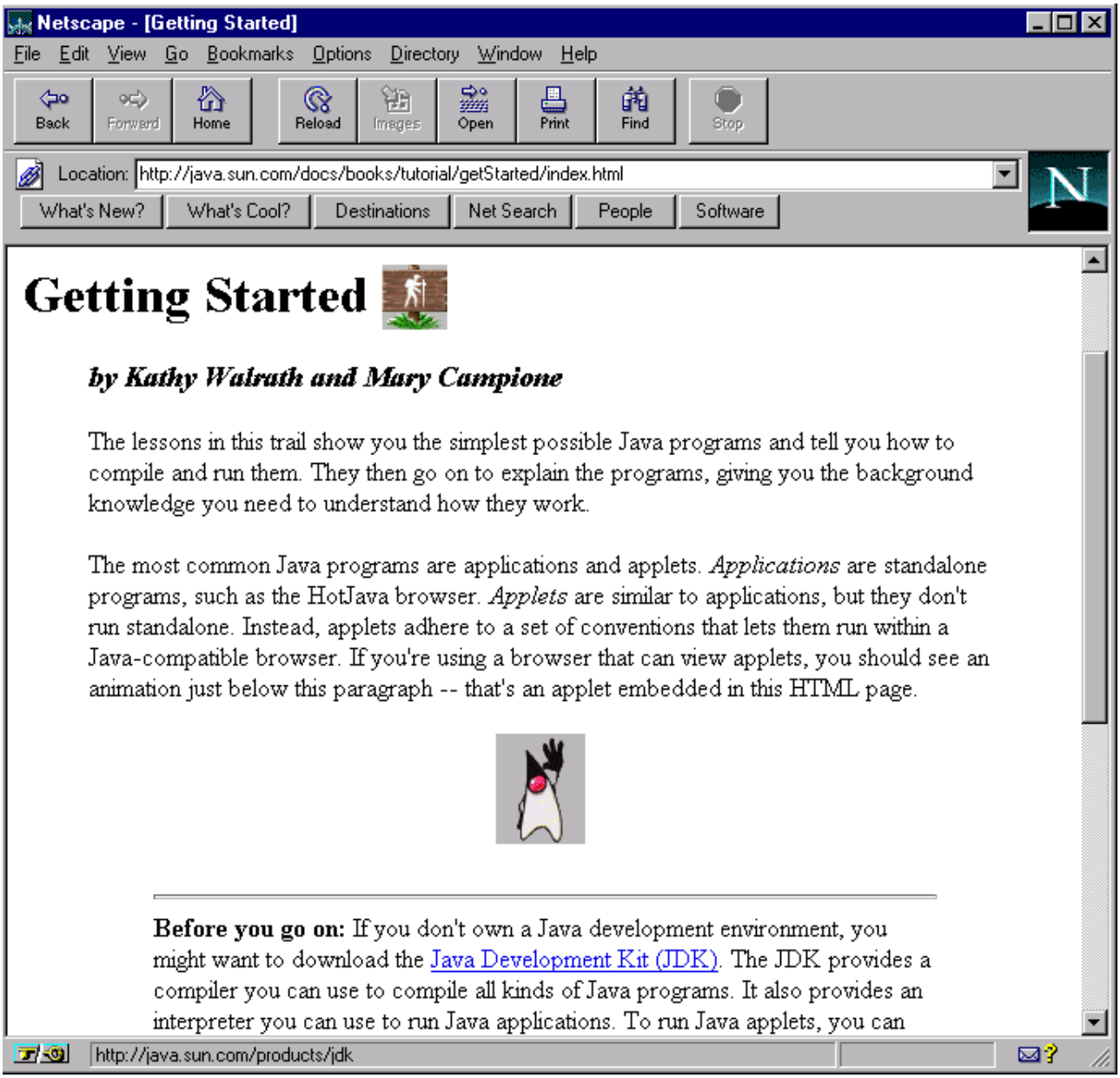
HTML **molto semplice** per non complicare il cliente

versione	**browser**	**proprietà**
1.0	storico	header, liste, enfasi
2.0	Mosaic	Inline Image, form
2.1	Navigator/Explorer	tabelle, allineamento
3.2	Navigator/Explorer	frame, JavaScript...
4.0	Navigator/Explorer	Stili (CSS), …
5.0/6.0	Navigator/Explorer	XML, …

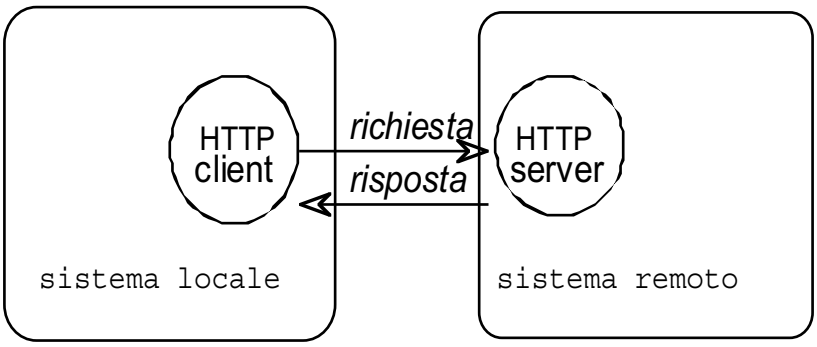
Esempio pagina HTML (codice)

```
<head> <title>Getting Started</title> </head>

<body>
<h1> Getting Started <img src=../images/Start.gif
height=40 width=40 align=top>
</h1>
<p>
<h3><em>by Kathy Walrath and Mary
Campione</em></h3>
<p>
The lessons in this trail show you the simplest
possible Java programs and tell you how to compile
and run them. They then go on to explain the
programs, giving you the background knowledge you
need to understand how they work.
<p>
...........

<p align=center>
<center>
<applet code=Animator.class codebase="../example"
width=55 height=68>
        <param name=endimage value=10>
        <param name=pauses value="2500|100">
</applet>
</center>
</p>

<hr>
<strong>Before you go on:</strong> If you don't
own a Java development environment, you might want
to download the <a
href="http://java.sun.com/products/jdk"> Java
Development Kit (JDK)</a>. The JDK provides a
compiler you can use to compile all kinds of Java
programs. It also provides an interpreter you can
use to run Java applications. To run Java applets,
you can ............
```

Esempio pagina HTML (visualizzazione)



Programmazione client/server in WWW

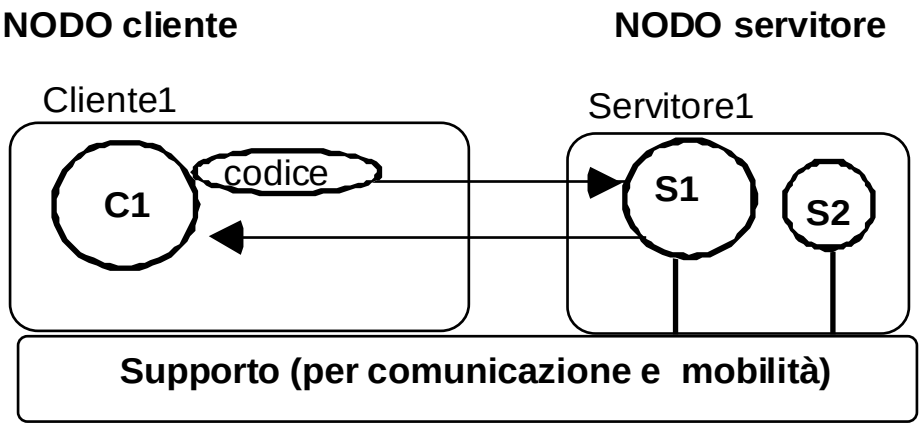


Possibilità di avere ***risposta*** con informazioni dinamiche

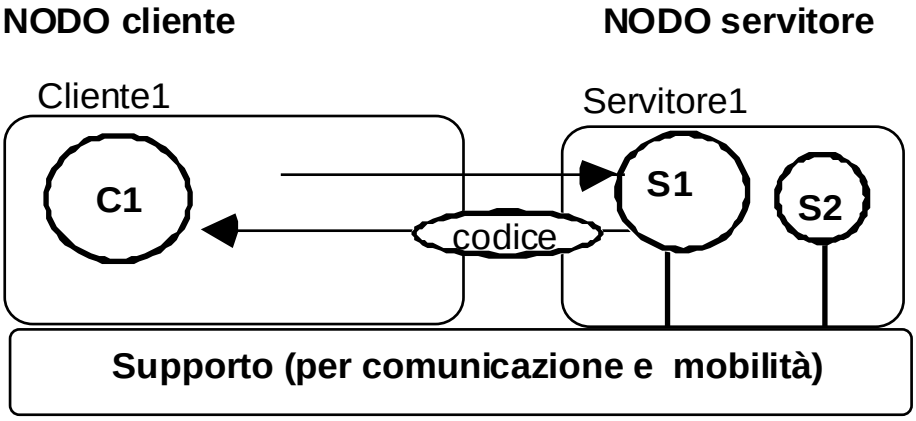
Che tipo di elaborazione delle informazioni e
dove viene eseguita

richiesta	risposta	tipo di elaborazione
Documento HTML	Statica (la pagina è un file, non modificabile)	semplice trasferimento file dal server
CGI	dinamica	qualunque elaborazione sul nodo server
Java applet	statica	codice dal server non modificabile, esecuzione sul client
Java applicazione	dinamica	server elabora dinamicamente il codice (in base alla richiesta), esecuzione dinamica sul client

Modelli di Programmazione e mobilità



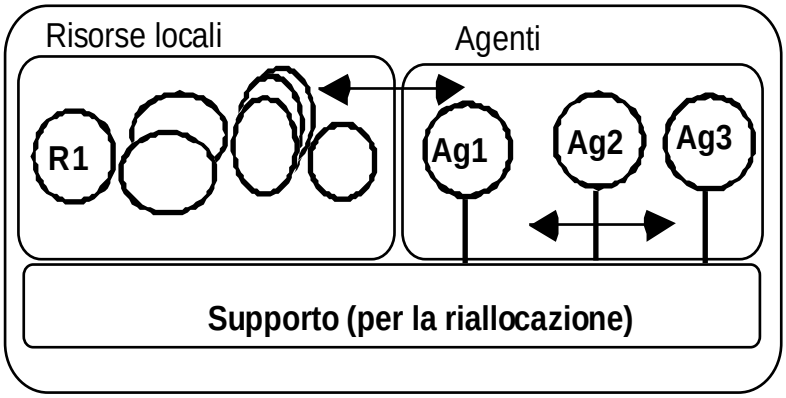
REMOTE EVALUATION

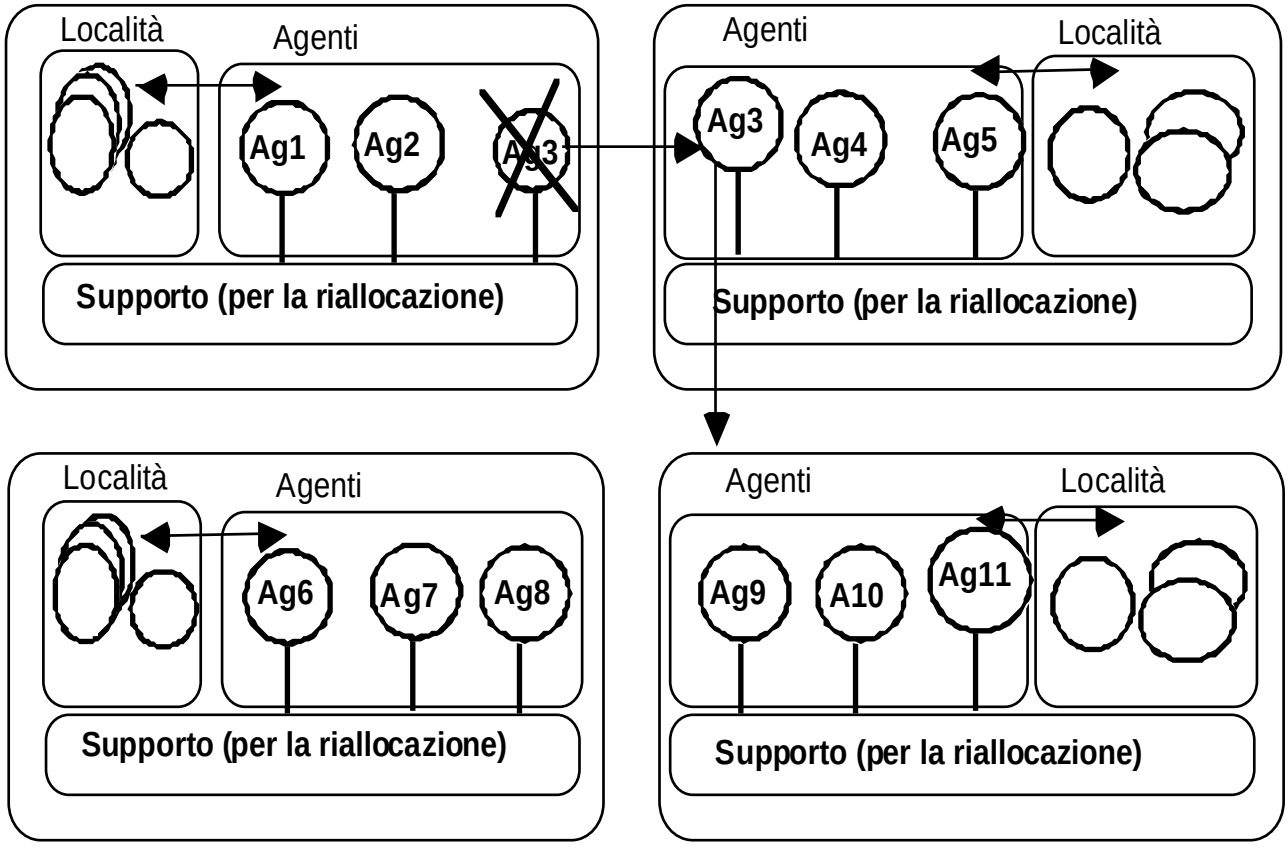


CODE ON DEMAND

Modelli di Programmazione e mobilità (Agenti Mobili)

NODO VIRTUALE di un sistema ad agenti





Modelli di Programmazione e mobilità

Dimensioni di classificazione del modello Client/Server:

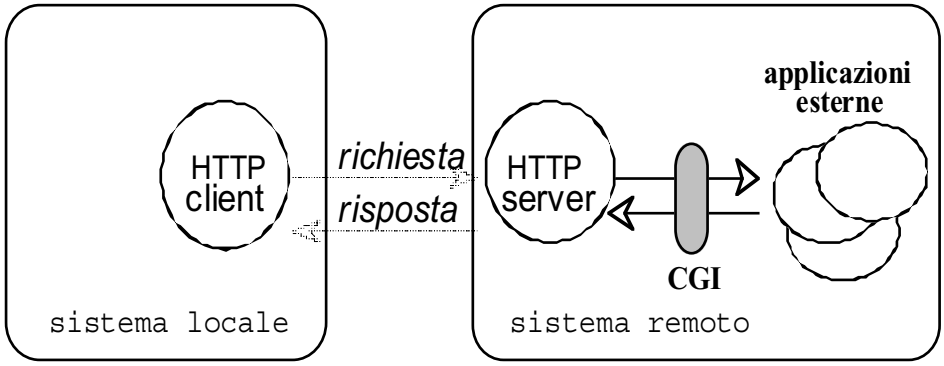
- trasferimento del **codice**
- trasferimento dei **dati**
- trasferimento della **E**xecution **U**nit (lo stato di esecuzione di un processo, senza ripartire dall'inizio)

modello	trasferimento codice	trasferimento dati	trasferimento EU
RPC	NO	SI'	NO
COD	SI'	NO	NO
REV	SI'	SI'	NO
MA	SI'	SI'	SI'

RPC: Remote Procedure Call
COD: Code On Demand (Applets java)
REV: Remote Evaluation (remote execution UNIX)
MA: Mobile Agents

Mobilità **strong** e mobilità **weak**

Common Gateway Interface (CGI)



CGI è uno **standard** per interfacciare un server WWW con applicazioni esterne (residenti sulla macchina server)

CGI fornisce all'utente la capacità di eseguire una applicazione sulla macchina server remota

Strumento tipico non interattivo
Collo di bottiglia

richiesta	***risposta***	***tipo di elaborazione***
CGI	dinamica	qualunque, sul nodo server

Programmazione CGI

Una applicazione CGI permette agli utenti di eseguire una applicazione sul nodo dove risiede il server www.

Applicazioni CGI possono essere scritte in:

- C/C++
- Fortran
- PERL
- TCL
- Any Unix shell
- Visual Basic
- AppleScript

normale attivazione di programma Unix, modello filtro con variabili di ambiente predefinite

Devono rispettare l'interfaccia CGI con il server

Interfaccia tra **server www** e applicazione **CGI**:

- variabili di ambiente
- linea di comando
- standard input
- standard output

Variabili d'ambiente

sono utilizzate dal server per dare informazioni di servizio all'applicazione CGI:

SERVER_SOFTWARE, nome e versione del server HTTP
SERVER_NAME, nome nodo server o suo indirizzo
GATEWAY_INTERFACE, versione interfaccia CGI cui il server aderisce
REQUEST_METHOD, metodo invocato nella richiesta
REMOTE_HOST, REMOTE_ADDR, AUTH_TYPE, REMOTE_USER, CONTENT_TYPE, CONTENT_LENGTH, ...

Linea di comando

per richieste di tipo ISINDEX, per ricerche di testo nei documenti. Le parole da ricercare sono inserite dal server sulla linea di comando della applicazione CGI. (compatibilità)

Standard input

il server ridirige sull'ingresso della applicazione CGI i dati ricevuti dal client (browser). Il numero di byte è nella variabile d'ambiente CONTENT_LENGTH, il tipo dei dati MIME nella CONTENT_TYPE.

Standard output

l'applicazione CGI manda il risultato dell'elaborazione sullo standard output verso il server, che a sua volta prepara i dati e li spedisce al client.

Client HTTP → server HTTP → CGI

Tipicamente, uso di **form**

```
<TITLE>Esempio di Form </TITLE>
<H1>Esempio di Form </H1>

<FORM METHOD="POST" ACTION="http://www-
lia.deis.unibo.it/cgi-bin/post-query">

Inserisci del testo: <INPUT NAME="entry">

e premi per invio: <INPUT TYPE="submit"
                    VALUE="Invio">
</FORM>
```

Visualizzazione **form**

Client HTTP → server HTTP → CGI

Attributi del **form tag**

```
<TITLE>Esempio di Form </TITLE>
<H1>Esempio di Form </H1>

<FORM METHOD="POST" ACTION=
"http://www-lia.deis.unibo.it/cgi-bin/post-query">

Inserisci del testo: <INPUT NAME="entry">

e premi per invio: <INPUT TYPE="submit"
                    VALUE="Invio">
</FORM>
```

Dove:

ACTION URL di chi processa la query

METHOD metodo usato per sottomettere il form:

- **POST** il form con i dati è spedito come data body (metodo consigliato)
- **GET** il form con i dati è spedito attaccato all'URL (`action?name=value&name=value`)

caso `GET`

```
http://www-lia.deis.unibo.it
                    /cgi-bin/get-query?entry=testo
```

caso `POST`

```
http://www-lia.deis.unibo.it
                    /cgi-bin/post-query
```

e come data body:

```
                    entry=testo
```

Applicazione CGI → server HTTP

Applicazione CGI usa **standard output** per mandare al server i dati. I dati sono identificati da un header.

Tipi di dati forniti:

- full document con il corrispondente MIME type (text/html, text/plain per testo ASCII, etc.)

  Esempio: per spedire una pagina HTML

```
Content-type: text/html

<HTML><HEAD>
<TITLE>output di HTML da script CGI</TITLE>
</HEAD><BODY>
<H1>titolo</H1>
semplice <STRONG>prova</STRONG>
</BODY></HTML>
```

- reference a un altro documento

  Esempio: riferisco un documento gopher

```
Content-type: text/html
Location: gopher://httprules.foobar.org/0

<HTML><HEAD>
<TITLE > Sorry ... it moved </TITLE>
</HEAD><BODY>
<H1>go to gopher </H1>
Now available at
<A HREF="gopher://httprules.foobar.org/0">
         new location</A> of gopher server.
</BODY></HTML>
```

Applicazione CGI

Esempio: generazione della pagina di risposta (caso full document)

```
#include <stdio.h>
..............


main(int argc, char *argv[]) {
  int cl;
```

generazione di un full document in risposta

```
  printf("Content-type: text/html");

  cl = atoi(getenv("CONTENT_LENGTH"));

  for(x=0;cl && (!feof(stdin));x++) {
     ...
```

elaborazione dell'input (stdin)

```
     ...
  }

  printf("<H1>Query Results</H1>");
  printf("You submitted ...");

  for(x=0; x <= m; x++)
    printf(".....", ... , ....);
}
```

INTERNET: PRINCIPALI PROBLEMI

Problema del **traffico**

alcuni formati di informazioni possono richiedere un **eccesso di banda**

necessità di adeguare le infrastrutture

Problema della **sicurezza**

la trasparenza richiede la integrazione con i sistemi locali di esecuzione/visualizzazione: intrusioni o virus da controllare

riservatezza delle transazioni e autenticazione clienti

Problema della **visualizzazione**

alcuni formati richiedono una lunga visualizzazione: tempi di accesso molto rallentati

sfruttare la possibile concorrenza/asincronismo

Uso di clienti con strategie adatte a

abbreviare il **tempo di risposta**

favorire il browsing delle informazioni

(web navigation e ricerche mediante robot, spider, worm, ecc.)

evoluzione dei protocolli

per garantire una apertura alle esigenze man mano determinate

La ricerca di informazioni sul WEB

Il web è un ipertesto (un grafo) con ormai milioni di nodi → problema di reperire le informazioni

Esistono **indici del web** (detti anche cataloghi o directories), realizzati per facilitare la ricerca di informazioni su Internet.

Organizzazione indici:

- alfabetica
- per argomento (gerarchici)
- per area geografica (gerarchici)
- con possibilità di ricerca (parole chiave)

Gli indici sono costruiti utilizzando dei programmi che esplorano i site web presenti in rete.

Programmi più diffusi:

- search engine
- spider
- crawler
- worm
- knowbots (knowledge robots)

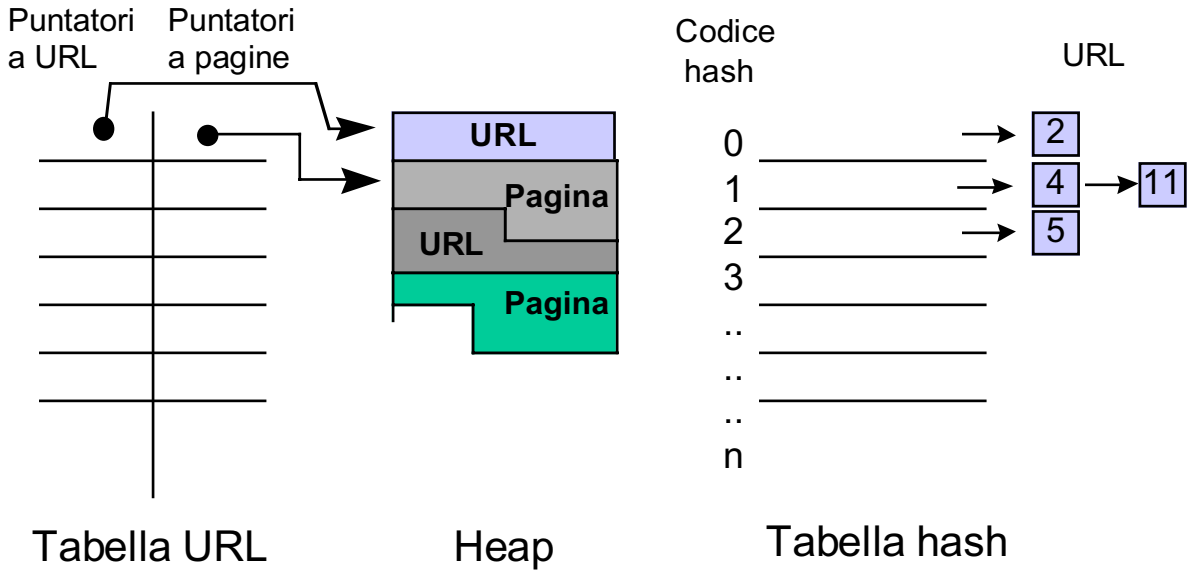
Esempio:
AltaVista (http://altavista.digital.com) Web index:

- 30 milioni pagine da 275,600 servers
- 4 milioni articoli da 14,000 Usenet news groups.

E' acceduto più di 21 milioni di volte al giorno
(dati di Ottobre 96)

I motori di ricerca

Struttura dati tipica:



Due fasi: **ricerca** e **indicizzazione**

Passi della **ricerca** (algoritmi tipo breadth-first, depth-first):

- prelevare un URL
- eseguire hash URL
- Se hash URL presente in Tabella hash allora STOP

altrimenti

- aggiungere hash URL in Tabella hash
- aggiungere Puntatori a URL e a pagina in Tabella URL
- aggiungere URL e Pagina (o titolo) in Heap
- ripetere tutti i passi per ogni link presente nella pagina

Problemi:

- dimensioni del grafo web
- punto di partenza della ricerca
- tipo di ricerca: depth-first → stack overflow
  breadth-first → dimensioni heap
- come trattare i link presenti nelle active map (CGI)
- URL obsoleti
- Macchine non raggiungibili rallentano la ricerca

I motori di ricerca

Fase di **indicizzazione**

La procedura di indexing estrae le parole chiave da ogni pagina (o titolo) web memorizzati nell'heap nella fase di ricerca (sintesi delle pagine)

Per trovare le parole chiave:
- si scartano le parole poco significative (articoli, etc.)
- si scelgono parole che nella pagina hanno la frequenza maggiore (es. Lycos)

Per ogni parola ottenuta si memorizza in una tabella la parola e l'URL che la contiene.

Alla fine della indicizzazione si ordina la tabella sulle parole e si salva su file che verrà consultato per le ricerche da parte degli utenti

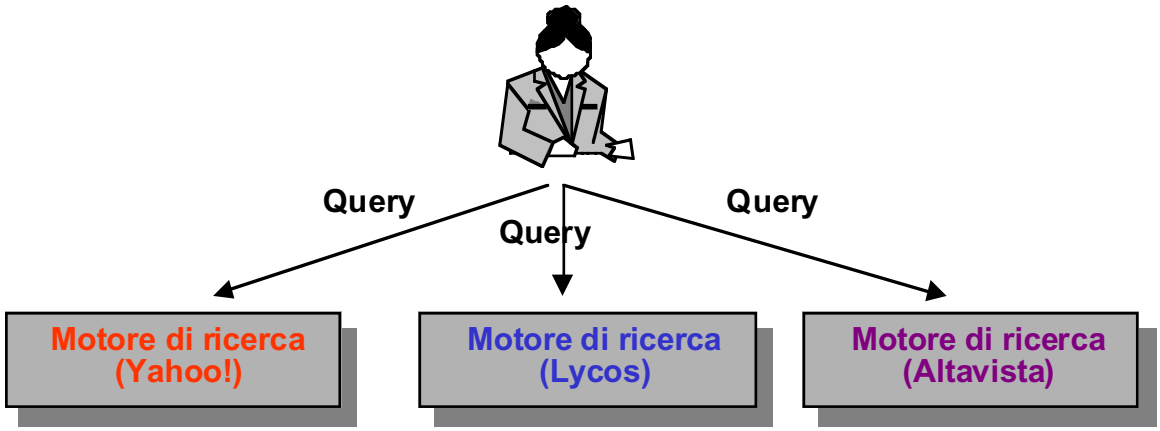
problemi:
- titoli pagine spesso poco significativi
- analisi intere pagine costosa
- pagine solo video o audio, oppure active map

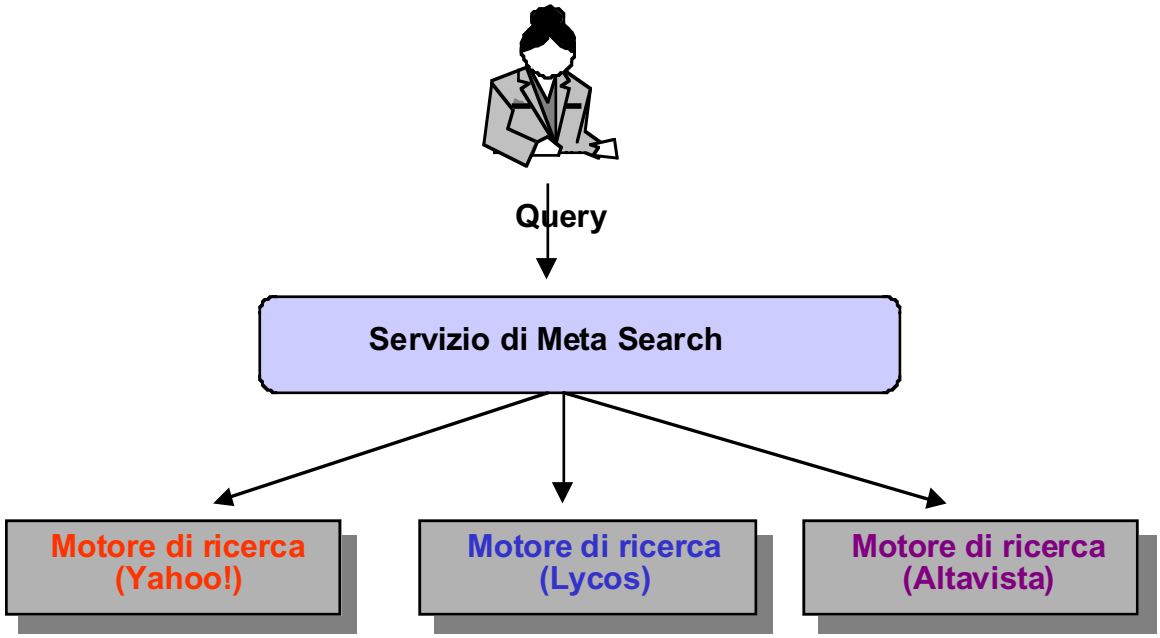
Ricerche e indicizzazioni cooperative: Harvest è un motore di ricerca che richiede a tutti i server www di eseguire una applicazione per indicizzare localmente la macchina. Un motore centrale raccoglie tutti i risultati
Problema della sicurezza

I meta-searcher

Effettuano la ricerca delle informazioni su più indici del web contemporanemente.

Un meta-searcher invia la stessa query contemporaneamente a più indici del web, NON contiene un database





Evoluzioni HTTP

Necessità di **evoluzione** ==> (HTTP 1.1)
- stato associato al cliente (cookies)
- stato nel servitore
- uso di **una** connessione per più richieste

Anche apertura da parte del cliente di più connessioni contemporaneamente

Estensione dei metodi di richiesta:
- operazioni tipo **FTP**

Evoluzione verso un passaggio di informazioni in unico **stream** con **ritrovamento asincrono** dalla parte del cliente

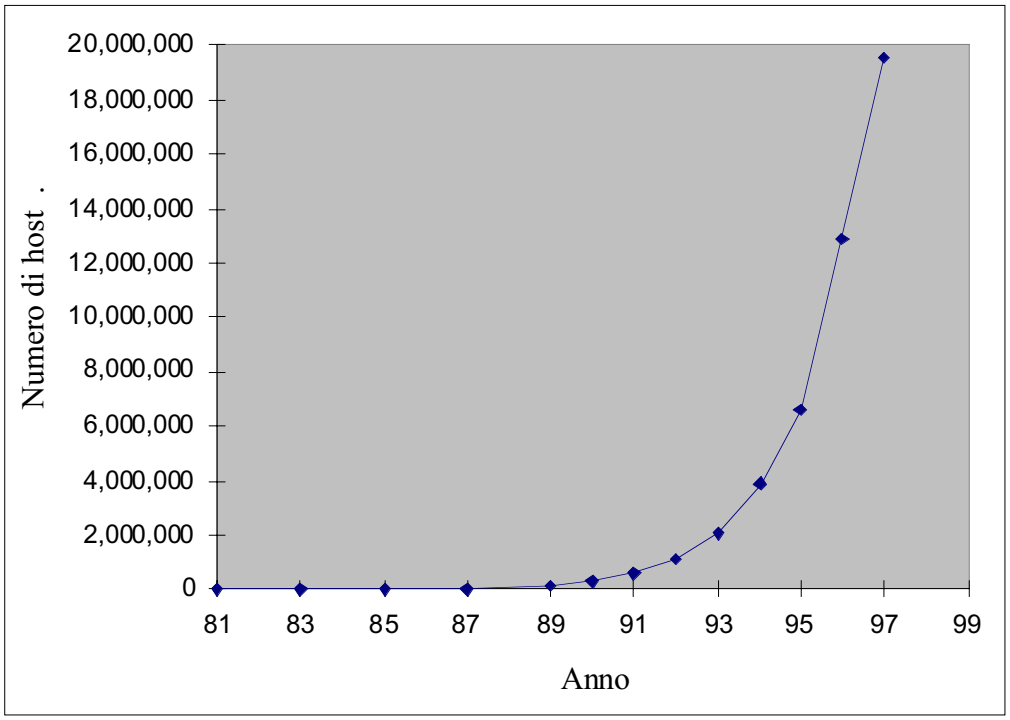
Limiti della **banda**
- uso di caching su servitori intermedi
- cache configurate come agenti con capacità di aggiornare reciprocamente informazioni

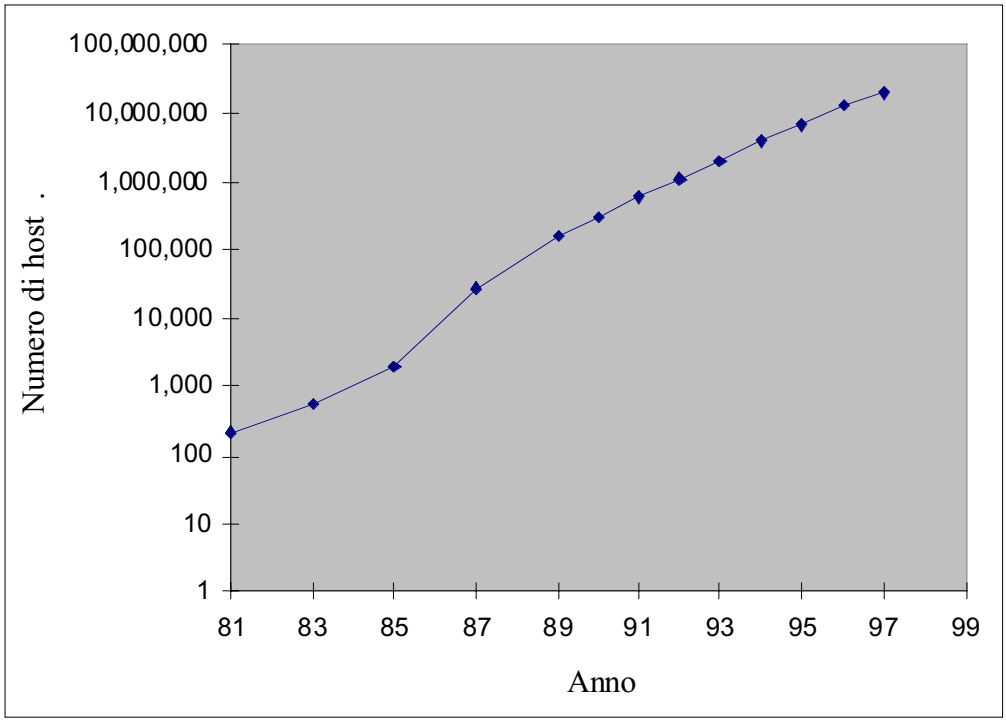
Non adeguatezza degli **strumenti**
- capacità di integrare tool per scrivere e usare informazioni multimediali
- uso di protocolli Real-Time

Modifica dei **modelli**
- superare i limiti di interazione, interfacce user-friendly non appena bande più ampie disponibili

Processo di standardizzazione diffuso e veloce

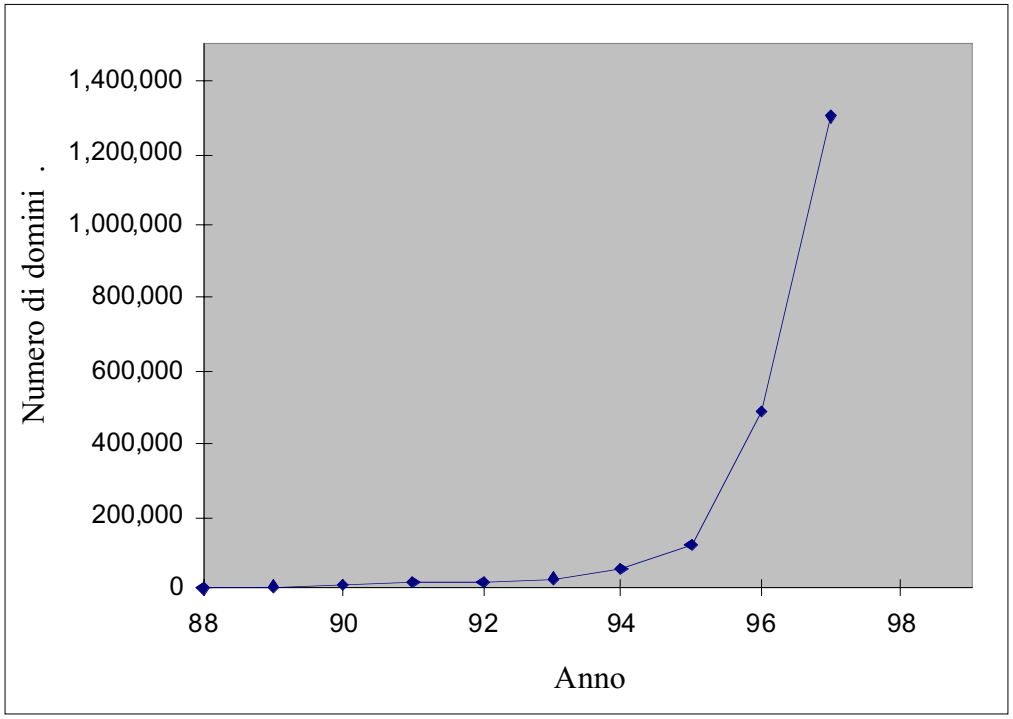
La crescita di Internet

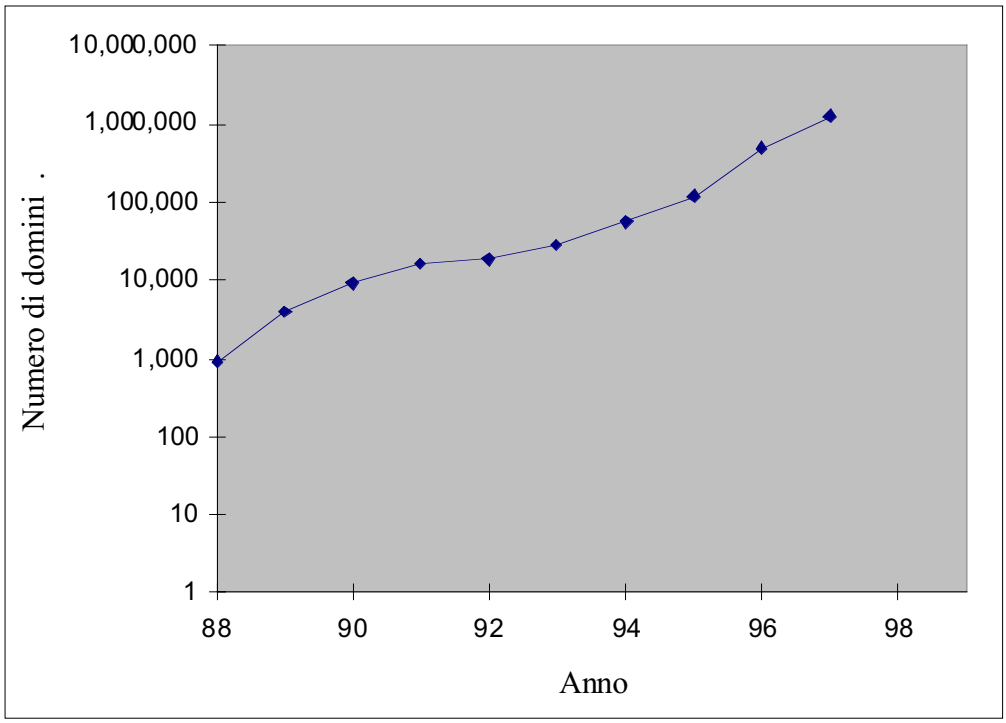




numero di **macchine** collegate ad Internet
(stima utenti = No. Host x 10)

La crescita di Internet





numero di **domini** Internet

<NIC.MERIT.EDU> /nsfnet/statistics/history.hosts

Internet Connectivity