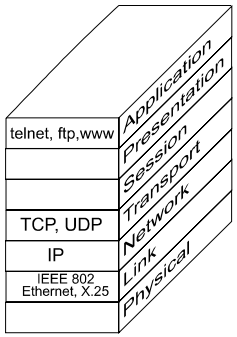
Servizi di UNIX (livello applicazione)



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/Servitore (senza stato con evoluzioni)
Standardizzazione

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 (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 (**r**emote **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

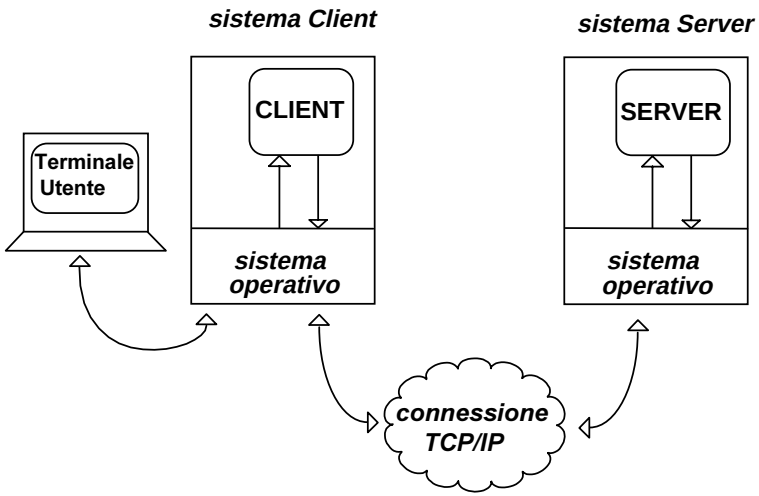
- **gestione eterogeneità** tramite interfaccia di terminale virtuale

- **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:

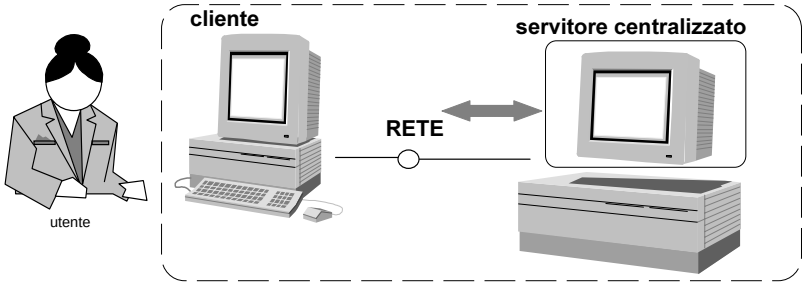
definizione di un **terminale virtuale**

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

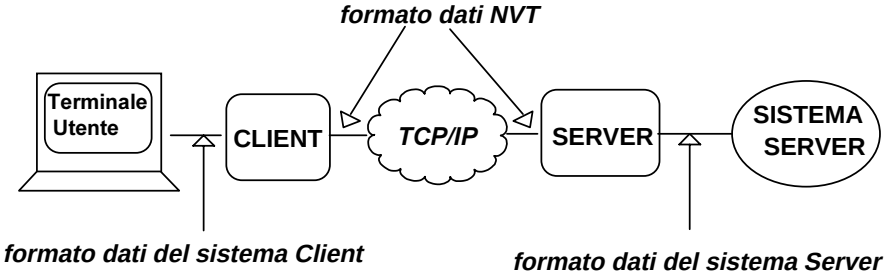
detto **standard NVT** (ossia **N**etwork **V**irtual **T**erminal)



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^{o} 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 láccordo 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

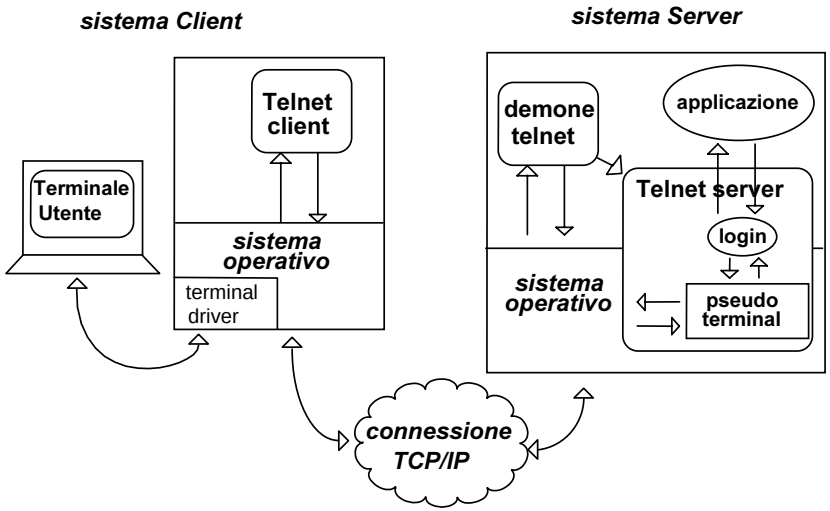
half-duplex
one char at a time (echo fatto sempre dal server)
problemi di overhead
one line at a time
problemi di ritardo nei dati
linemode

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)

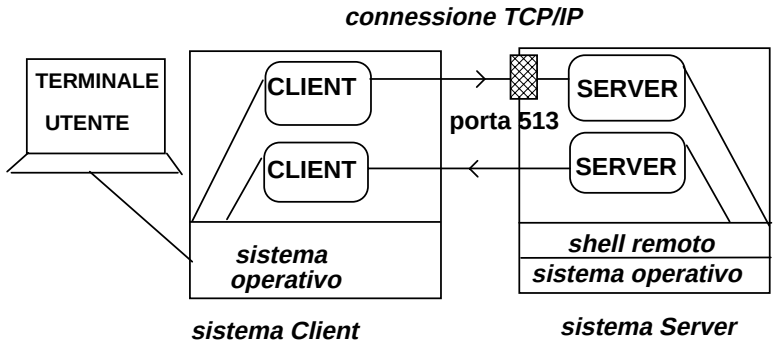
- rlogin molto più snello di telnet ma minori prestazioni
  migliaia di linee vs. decine di migliaia

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

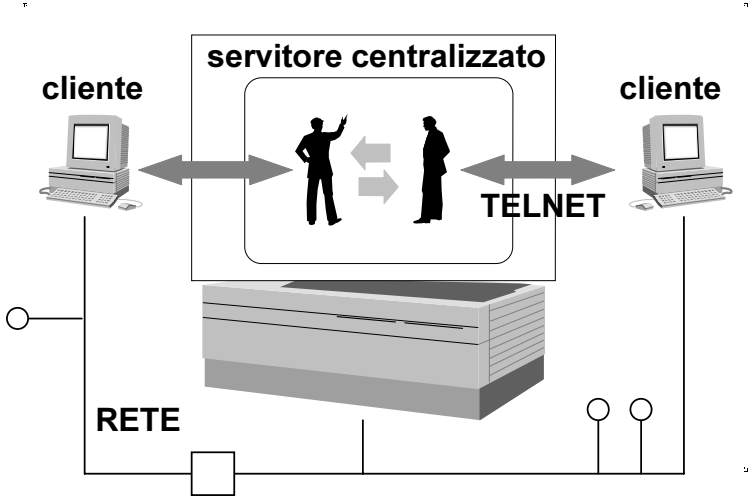
Anche *rsh*
invoca l'interprete (shell) remoto UNIX
con gli argomenti della linea di comando
rsh nodo_remoto comando

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, ...`

tfpt 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: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
```

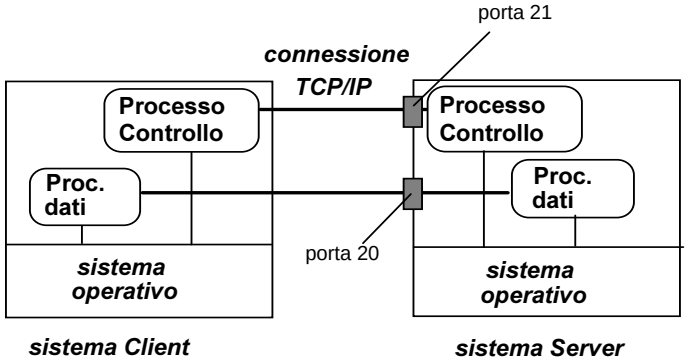
Implementazione FTP

Vari tipi di file riconosciuti

filetype	**ASCII**,	EBCDID,	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 coordianemto 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 tftp	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
```

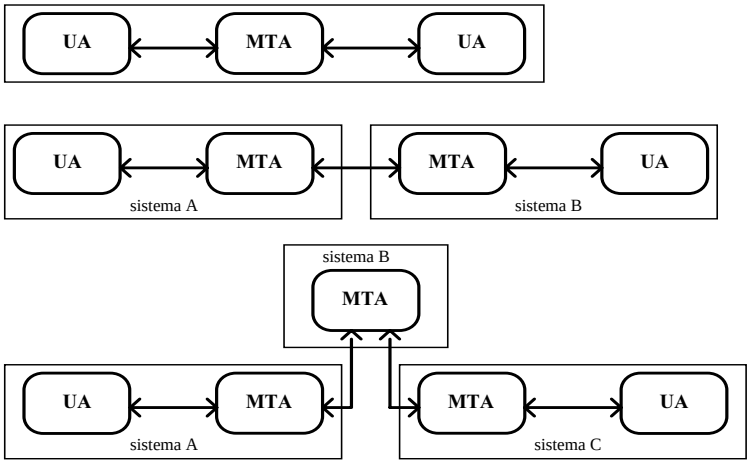
```
beppe ingbo ~ 10 > Mail
Mail version SMI 4.1-OWV3 Mon Sep 23 07:17:24 PDT
1991  Type ? for help.
"/usr/spool/mail/beppe": 1 message 1 unread
>U   1 antonio                 Fri  May  22  16:48
14/296   Prova di mail

& 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
```

Architettura del servizio di mail

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

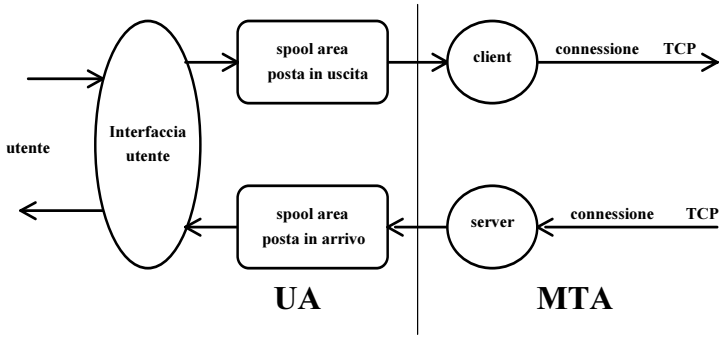
Mail Transport Agent (MTA) trasferisce mail dal user agent (UA) sorgente a quello di destinazione

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

Si deve regolare il coordinamento tra diversi MTA
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

SMTP regola come gli MTA si organizzano tra di loro
le politiche adottate per decidere la topologia di scambio delle informazioni non fa parte del protocollo

Indirizzi di mail

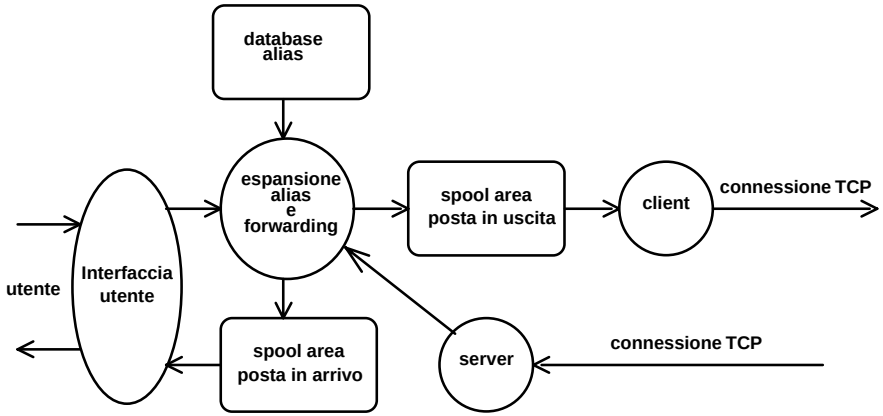
destinatario come

{ **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 può avere **più 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*

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

più sottodomini e nomi multipli

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

ROUTING tra MTA

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

Il sistema di nome della posta elettronica può essere

- risolto sul sistema di nomi di DNS
  e riportato a IP
- basato sulle corrispondenze di DNS
  ma 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

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
`Replay-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:
```

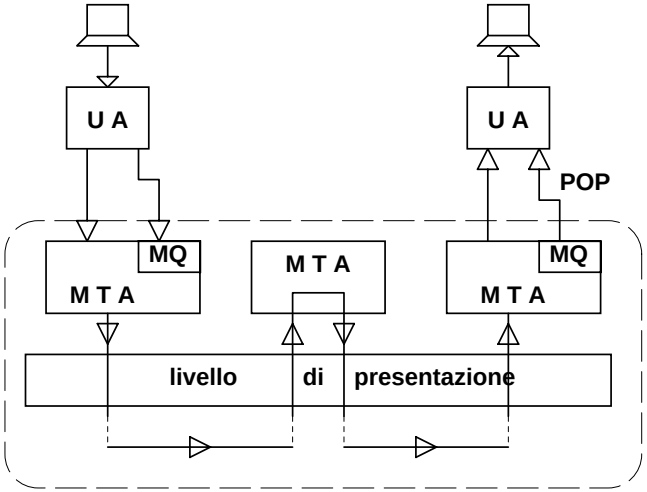
Content-type	*Subtype*	
text	*plain* *richtext* *enriched*	testo non formattato testo con semplice formattazione (bold, ecc) raffinamento del richtext
multipart	*mixed* *parallel* *digest* *alternative*	parti da processare seq. parti su cui lav. in parallelo message digest molte copie con la stessa semantica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

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

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)



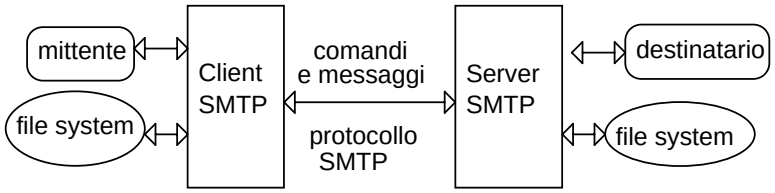
La porta TCP per gli scambi tra MTA è la **25**
e i singoli messaggi sono regolati

POP **P**ost **O**ffice **P**rotocol
Molti lettori diversi di posta elettronica:
Mail, mail, elm, eudora, ...
anche **sicuri**
IMAP. ecc., anche integrati con Kerberos, DES, 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 (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

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.

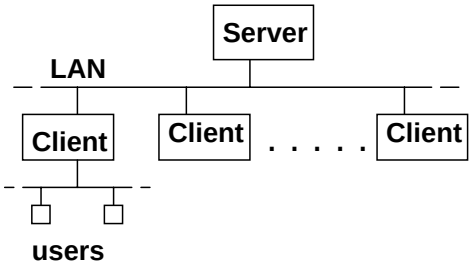
Insiemi aperti di interessi pubblici
GERARCHIE PRINCIPALI DI NEWS

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

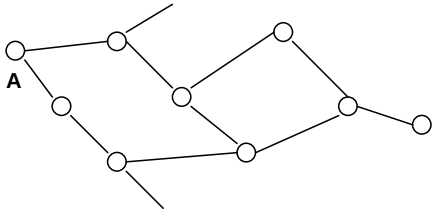
Architettura del servizio di NEWS



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.*

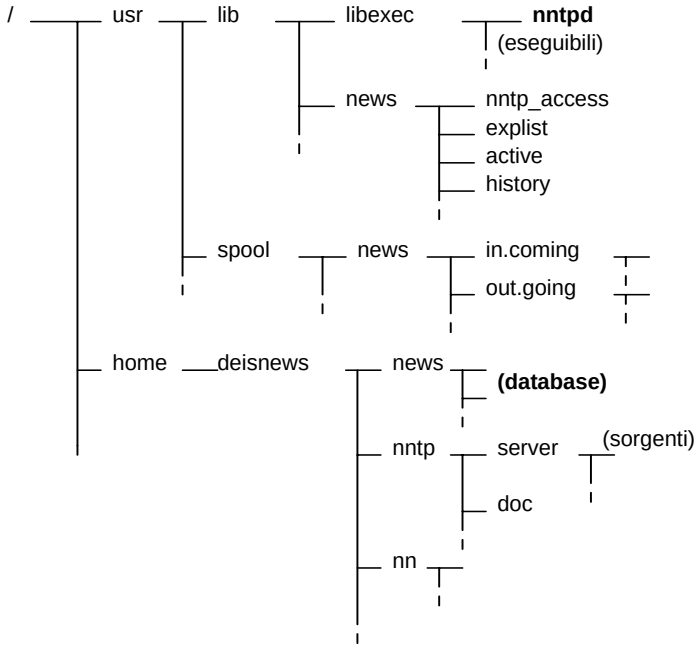
Uso di agenti con **TCP/IP** di connessione
News: uso di ***database coordinati*** per le informazioni



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**

Esempio di file system

in cui si memorizzano le news



explist	→	eliminazione
history	→	per 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
(dedicata la **porta 119**)

Protocollo NNTP

Protocolli a negoziazione

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 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 che sono distribuiti
Protocollo	SMTP	NNTP

Per **USENET**
Si notino

- la **dimensione globale** anche delle informazioni
- la distribuzione anche a **flooding**
- la distribuzione anche a **gruppi**

☹ nessuna sicurezza

si comincia a delineare la idea di una
INFRASTRUTTURA DI SUPPORTO

lasciata bianca (vedi poi)

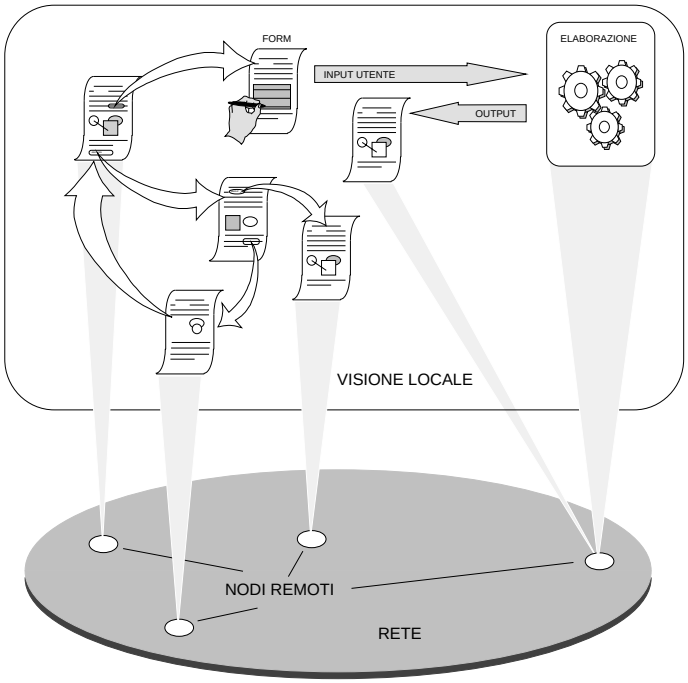
Strumenti trasparenti

unico insieme di informazioni accessibile a tutti i possibili utenti

Gopher

WWW (Mosaic, 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 ha *ampliato ulteriormente* la fascia di utenza (in modo esponenziale)

ma introdotto 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)
Progetto di integrazione in forma ipertestuale delle 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

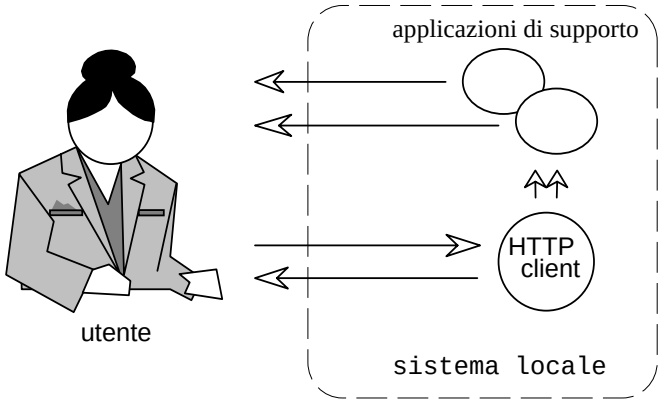
- Browser (presentazione e gestione richieste)
- Server (accesso e invio informazioni)
- Helper applications (particolari presentazioni)
- Applicazioni CGI (esecuzione remota)

Specifiche standard

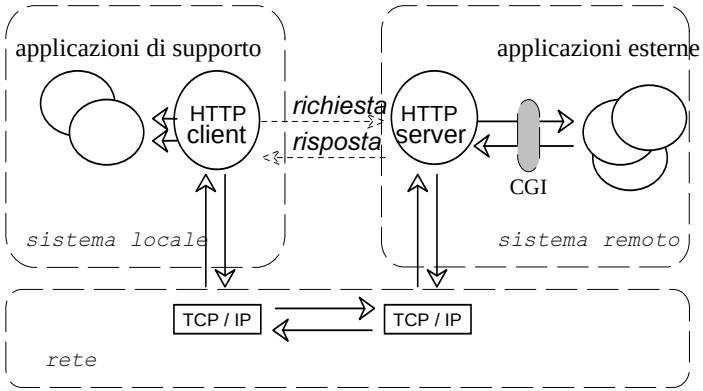
- Sistema di nomi universale URI e URL (Uniform Resource Identifier/Location)
- Protocollo HTTP (HyperText Transfer Protocol)
- Linguaggio HTML (HyperText Markup Language)
- Interfaccia CGI (Common Gateway Interface)

SISTEMA **WWW**

Cliente e sua interazione

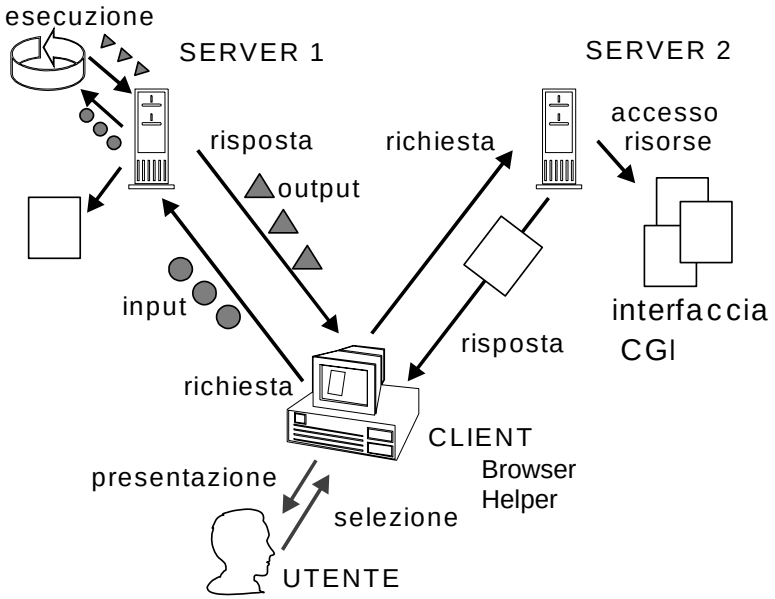


Il Cliente HTTP usa un modo cliente/servitore nei confronti di un server per volta e può anche interagire con risorse locali



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
- server parallelo
- server stateless
  (ma uso di **Common Gateway Interface**)

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

URL Uniform Resource Locators

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

- ***URN*** *(Uniform Resource Names) e servizio di name*
- **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

http://www.address.edu:1234/path/subdir/file.ext

servizio — host — porta — percorso

ftp://username:password@host.domain/path/file.ext
file:///drive/path/subdir/file.ext

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 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'è **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)
  (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

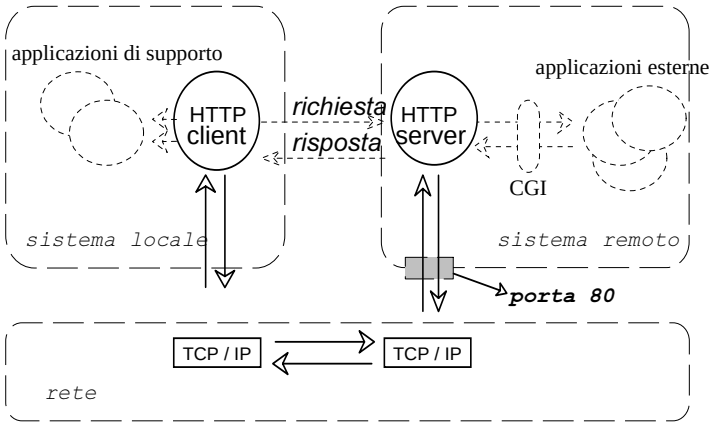
GET URL protocol vers.	headers	delimitatori	(dati opzionali)
	Browser tipi accettati	**CRLF**	**Dati Utente**

Formato del messaggio di **risposta** (CRLF)

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

Codice di stato di risposta

solite 3 cifre in chiaro

- **1xx: Informational**

risposta di successo parziale temporanea

- **2xx: Success**

Il server accetta la richiesta

200 OK (Get)
201 Created
206 Partial Content

- **3xx: Redirection**

altre informazioni richieste

301 la informazione mossa definitivamente
302 la informazione mossa temporaneamente
304 la informazione non modificata

- **4xx: Client error**

richiesta non soddisfatta per errore del cliente (errore sintattico o richiesta non autorizzata)

400 Bad Request
401 Unauthorized
402 Payment Required
403 Forbidden
404 Not Found

- **5xx: Server error**

richiesta corretta, ma il server non può soddisfarla (problema del server o di applicazioni CGI)

500 Internal Error
501 Extension Header

Interazione HTTP

Request

```
GET /beta.html HTTP/1.0
Referer: http://www.alpha.com/alpha.html
Connection: Keep-Alive
User-Agent: Mozilla/4.61
Host: www.alpha.com:80
Cookie: name=value
Accept: image/gif, image/jpeg, */*
Accept-Encoding: gzip
Accept-Language: en
Accept-Charset: iso-8859-1,*,utf-8
Expires: ...
If-modified-since: ...
```

Reply

```
HTTP/1.1 200 OK
Date: Fri, 12 Nov 2001 11:46:53 GMT
Server: Apache/1.3.3 (Unix)
Last-Modified: Mon, 12 Jul 2000 22:55:23 GMT
Accept-Ranges: bytes
Content-Length: 34
Content-Type: text/html
```

Si noti che le cose cominciano ad evolvere per facilitare le operazioni

Keep-alive *la connessione viene mantenuta*
condizioni *richieste condizionali*

proxy *agenti intermedi che fanno cache*

cookie *la possibilità di avere stato della interazione è delegata al cliente che mantiene associazioni nome=valore (cookie)*

HTTP1.1 RFC2616 (1999), state management RFC2109

HTML HyperText Markup Language

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>`

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
Visualizzazione dipendente dal browser

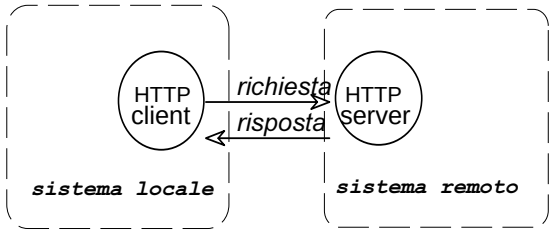
versione	browser	proprietà
1.0	storico	header, liste, enfasi
2.0	Mosaic	Inline Image, form
2.1	Netscape/Microsoft	tabelle, allineamento
3.2	Netscape/Microsoft	frame, ...

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

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)

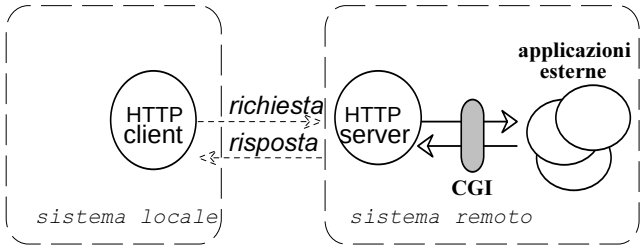
modello	trasferimento codice	trasferimento dati	trasferimento EU
RPC	NO	SI	NO
COD	SI	NO	NO
REV	SI	SI	NO
MA	SI	SI	SI

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 --> **standard per la interazione** con le **risorse del server** aldifuori del Web



CGI è uno **standard** per la interfaccia con applicazioni esterne (residenti sulla macchina server)

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

Uso di un direttorio specifico per contenere i programmi
/cgi-bin/

URL da eseguire del programma
http://host/cgi-bin/program

Strumento tipicamente **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 Web** e la applicazione **CGI**

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

Variabili d'ambiente
utilizzate dal server per dare informazioni alla CGI:

- `SERVER_NAME`, `SERVER_PORT`,
  - nome nodo server o suo indirizzo
- `SERVER_SOFTWARE`,
  - nome e versione del server HTTP
- `PATH_NAME`,
  - cammino per il programma CGI
- `GATEWAY_INTERFACE`,
  - versione interfaccia CGI cui il server aderisce
- `REQUEST_METHOD`,
  - metodo invocato nella richiesta GET/POST
- `QUERY_STRING`,
  - parametri passati nella GET
- `REMOTE_HOST, REMOTE_ADDR, AUTH_TYPE, REMOTE_USER, CONTENT_TYPE, CONTENT_LENGTH,`

Linea di comando
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.
numero di byte variabile d'ambiente `CONTENT_LENGTH`
tipo dei dati MIME nella `CONTENT_TYPE`

Standard output
l'applicazione CGI manda il risultato elaborato sullo standard output al server, che prepara i dati (HTML) 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
```


server HTTP → CGI

la CGI deve decodificare i dati in arrivo
arrivati con **POST o GET**

- gli spazi ("") sono rimpiazzati da "+"
- gli elementi del form sono coppie chiave - valore
  tipo nome=antonio
- ogni coppia è separata da &
- caratteri non numerici sono introdotti oda un carattrere di escape "!"-> %21

caso GET
leggiamo i dati come stringhe dall'environment
var -> `QUERY_STRING`

caso POST
leggiamo i dati dallo standard input

I passi della CGI

- **decodificare i parametri**
- **eseguire il programma**
- **riportare il risultato al Web server usando lo standard output**

(pagina HTML o statica o dinamicamente costruita)

Applicazione CGI → server HTTP

Applicazione CGI usa **standard output** per mandare al server i dati
I dati devono essere preceduti 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)

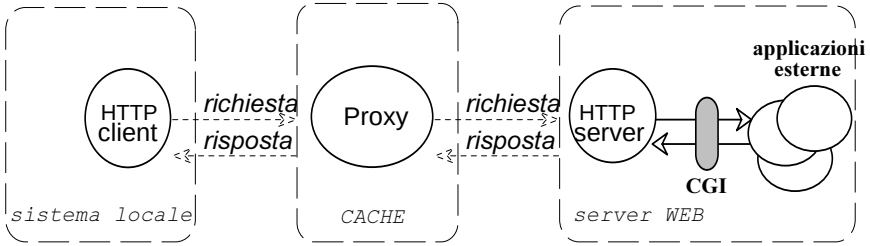
```
#include <stdio.h>
..............


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(".....", ... , ....);
}
```

Ambiente Complessivo di Esecuzione

I sistemi Web cominciano a delineare un nuovo modello di operazione
cliente
intermedi (**proxy, cache**)
servitore
con accesso alle **risorse** del servitore



Integrazione con **risorse di calcolo accedute via server**

Sistemi a diversi livelli (multitier)
in cui la elaborazione risultante viene fornita
in modo legacy (**Web compatibile**) ma
ottenuta tramite la interazione di molte entità in gioco

Il sistema comincia ad avere **stato** e a diventare il punto di accesso ad una **infrastruttura**

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

Web come ipertesto (grafo) con milioni di nodi → problema di reperire le informazioni attraverso i link

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** (**know**ledge ro**bots**)

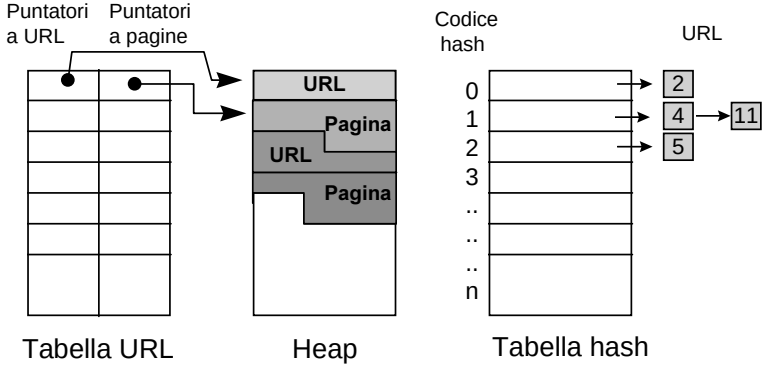
Esempio:
AltaVista (http://altavista.digital.com) Web index:

- 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 di supporto tipica:



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

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

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

altrimenti

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

Problemi:

- **dimensioni** del grafo web
- **punto di partenza** della ricerca
- **peso** dei link (anche autorità e centralità)
- tipo di ricerca: **depth-first** → stack overflow
  breadth-first → dimensioni heap
- come trattare i link presenti nelle **active map** (CGI)
- **URL obsoleti** e **macchine non** raggiungibili

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

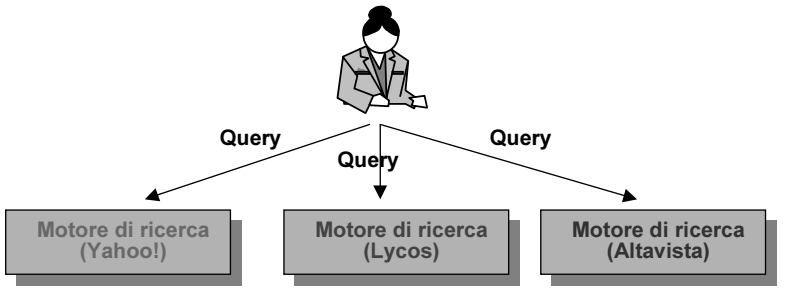
problemi:
- titoli pagine spesso poco significativi
- analisi intere pagine costosa
- pagine solo video o audio, oppure active map

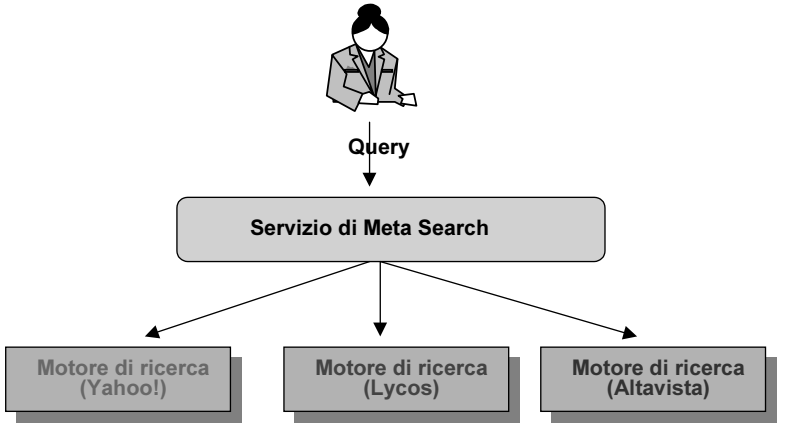
Ricerche e indicizzazioni cooperative
Harvest è un motore di ricerca che richiede a tutti i server www di eseguire una applicazione per indicizzare localmente la macchina
Un motore centrale raccoglie tutti i risultati
Problema della sicurezza

I meta-searcher

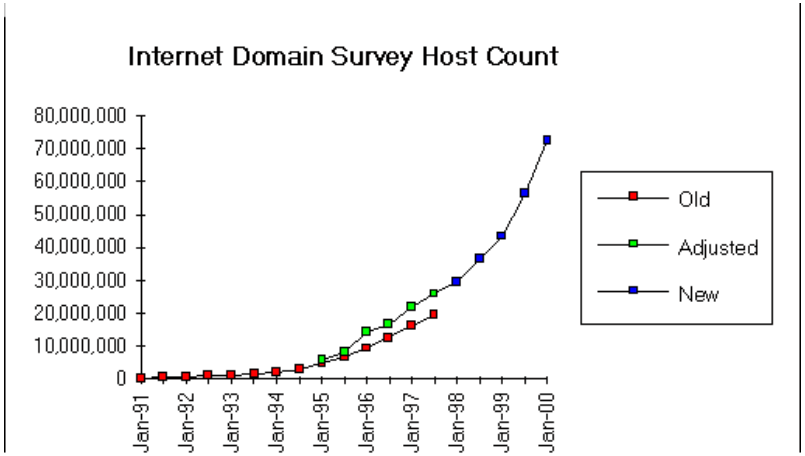
Effettuano la ricerca delle informazioni su più indici del web contemporanemente.

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





Dati recenti su Web



Source: Internet Software Consortium (www.isc.org)

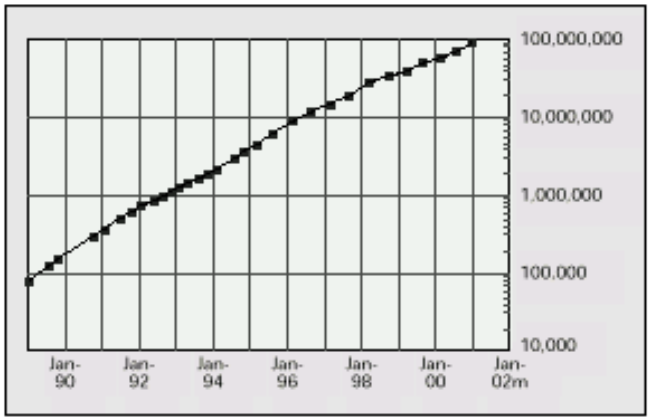


Figure 1. *Overall trends in Internet hosts*