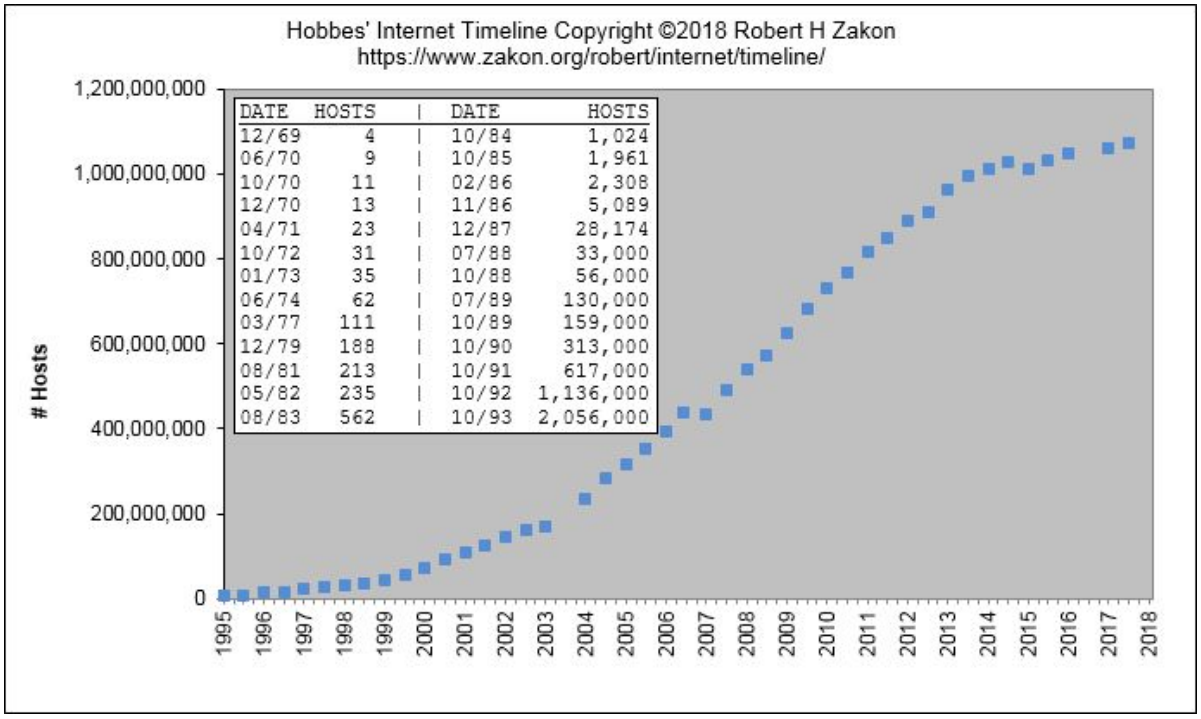
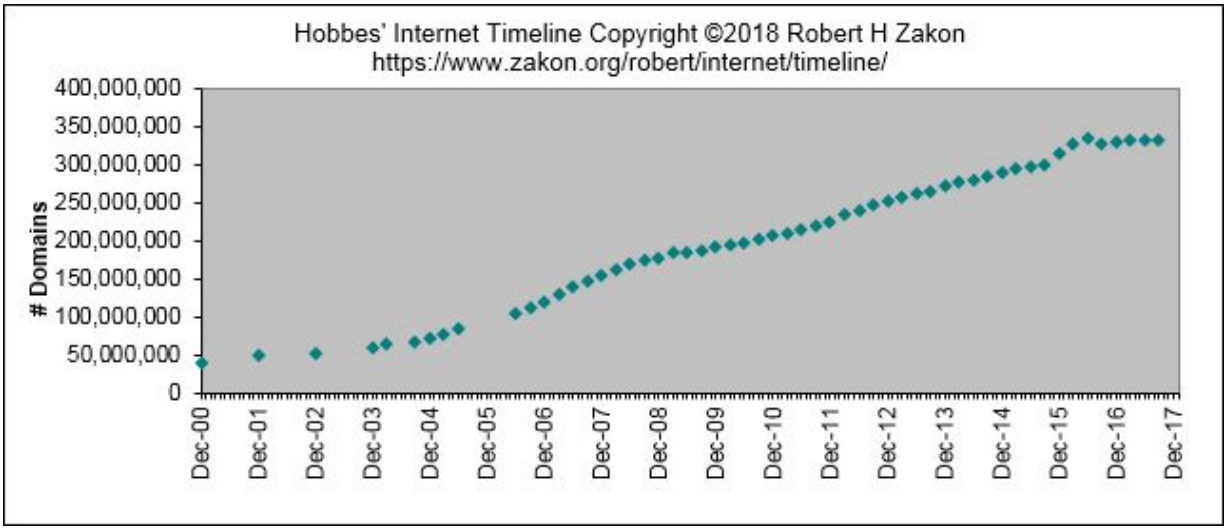
La crescita di Internet (N. host collegati)

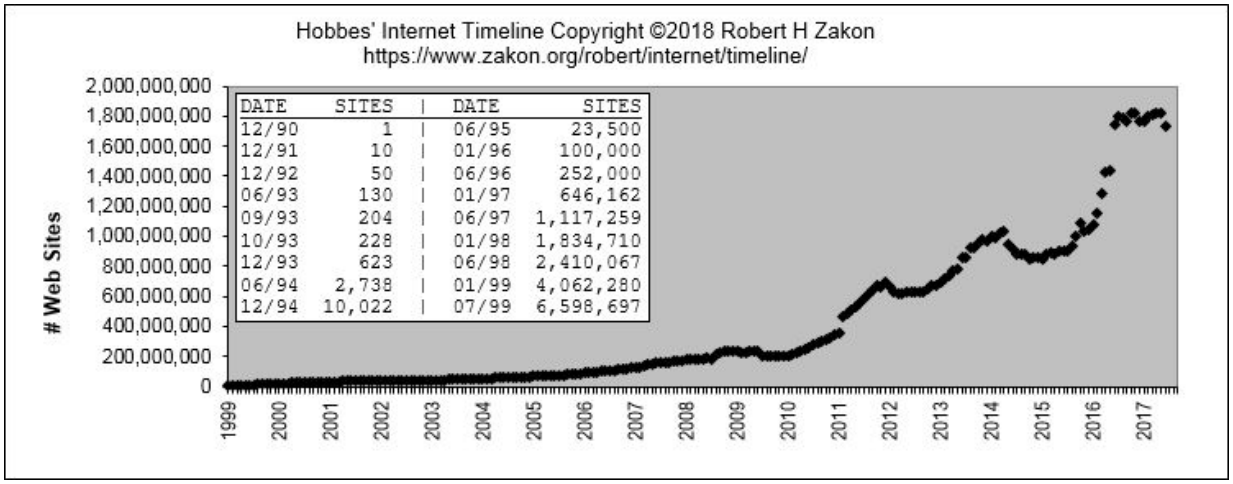


DATE	HOSTS	DATE	HOSTS
12/69	4	10/84	1,024
06/70	9	10/85	1,961
10/70	11	02/86	2,308
12/70	13	11/86	5,089
04/71	23	12/87	28,174
10/72	31	07/88	33,000
01/73	35	10/88	56,000
06/74	62	07/89	130,000
03/77	111	10/89	159,000
12/79	188	10/90	313,000
08/81	213	10/91	617,000
05/82	235	10/92	1,136,000
08/83	562	10/93	2,056,000

La crescita di Internet (N. domini registrati)

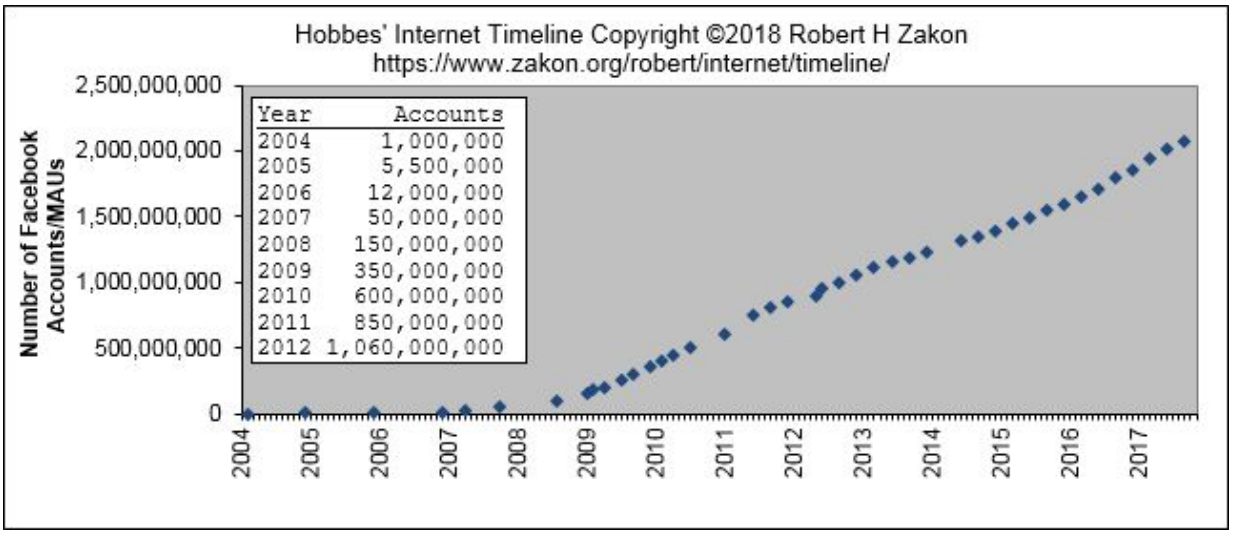


La crescita del Web (N. siti)

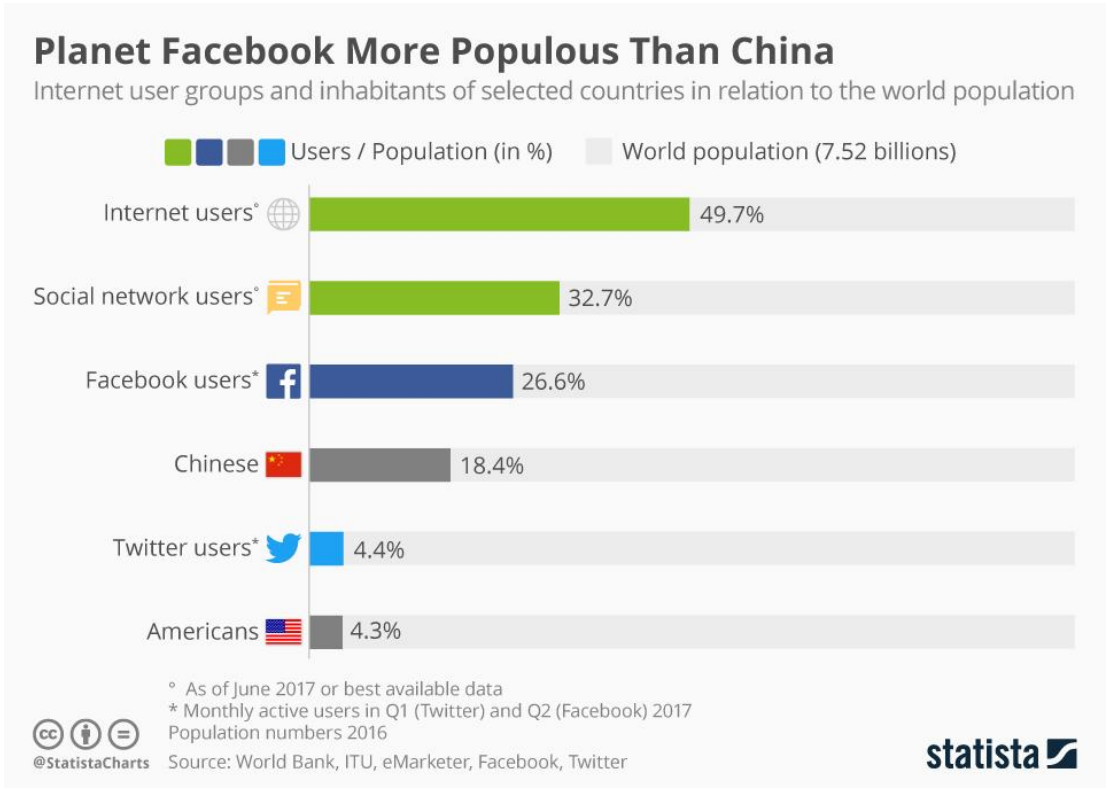


DATE	SITES	DATE	SITES
12/90	1	06/95	23,500
12/91	10	01/96	100,000
12/92	50	06/96	252,000
06/93	130	01/97	646,162
09/93	204	06/97	1,117,259
10/93	228	01/98	1,834,710
12/93	623	06/98	2,410,067
06/94	2,738	01/99	4,062,280
12/94	10,022	07/99	6,598,697

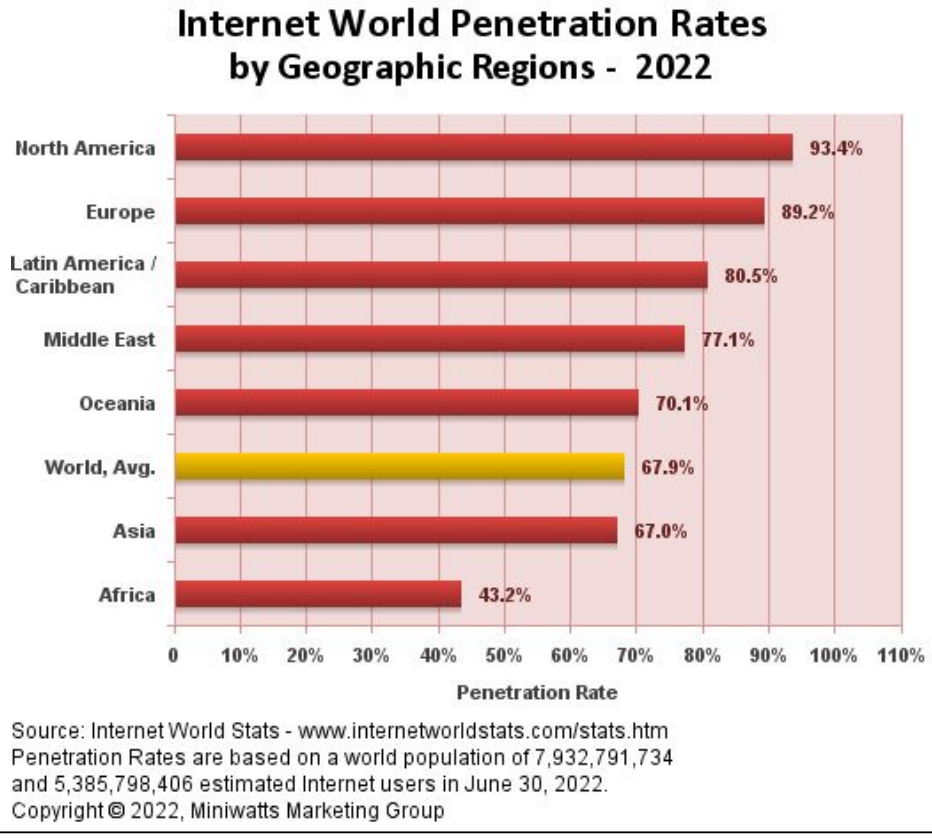
La crescita dei social (N. utenti Facebook)



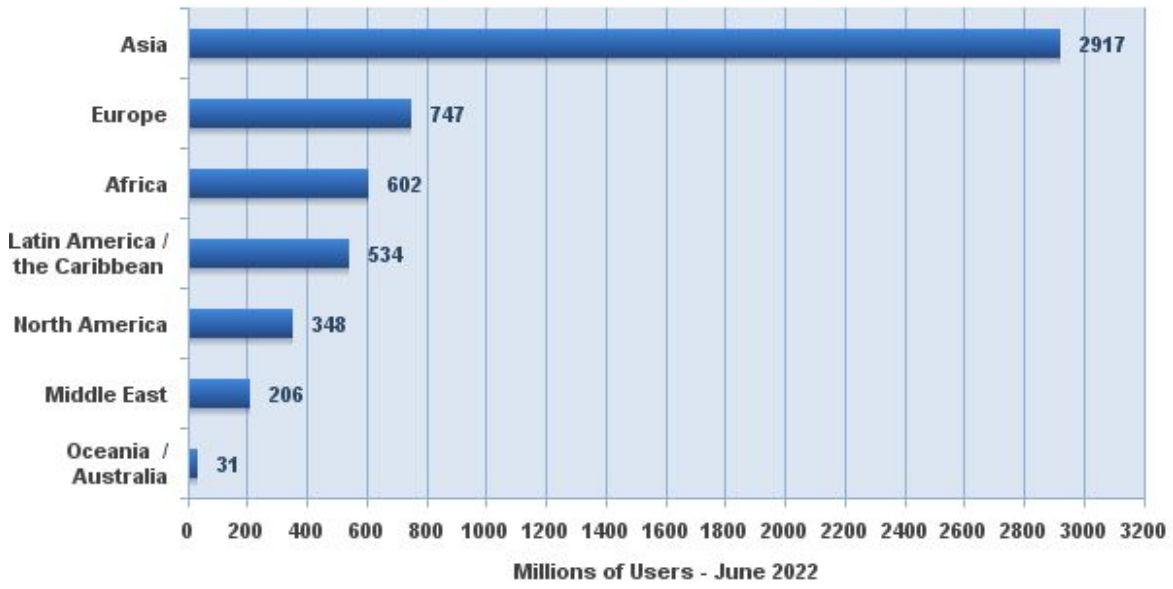
Year	Accounts
2004	1,000,000
2005	5,500,000
2006	12,000,000
2007	50,000,000
2008	150,000,000
2009	350,000,000
2010	600,000,000
2011	850,000,000
2012	1,060,000,000



https://www.statista.com/chart/10457/social-media-facebook-twitter-world-population/

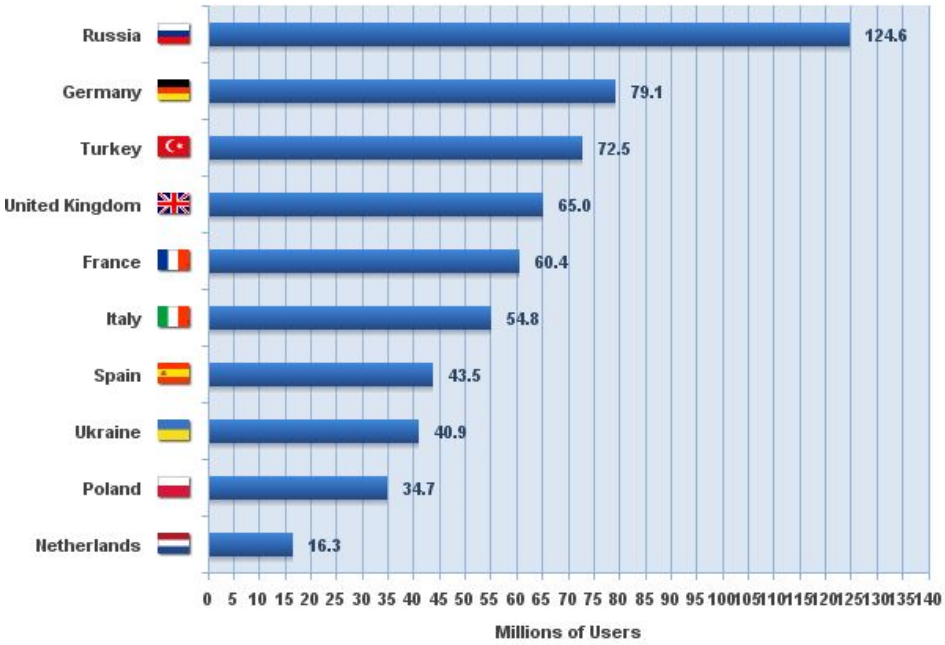


Internet Users in the World by Geographic Regions - 2022

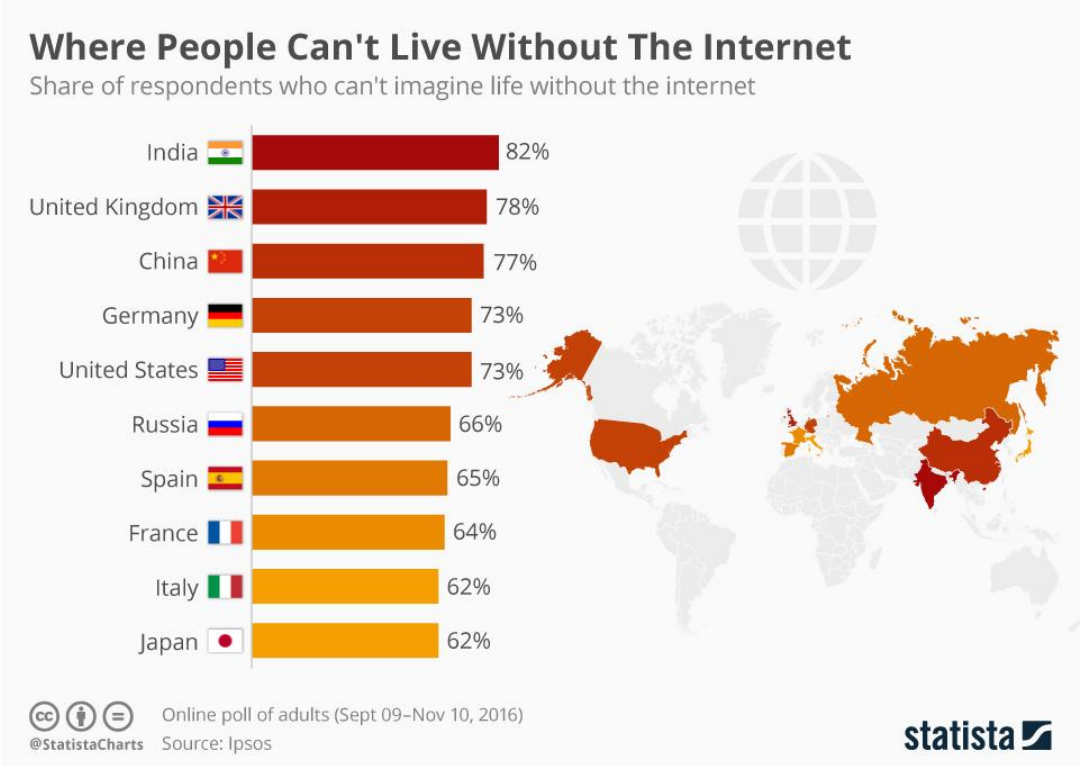


Source: Internet World Stats - www.internetworldstats.com/stats.htm
Basis:5,385,798,406 Internet users estimated in June 30, 2022

Internet Top 10 Countries in Europe July 31, 2022

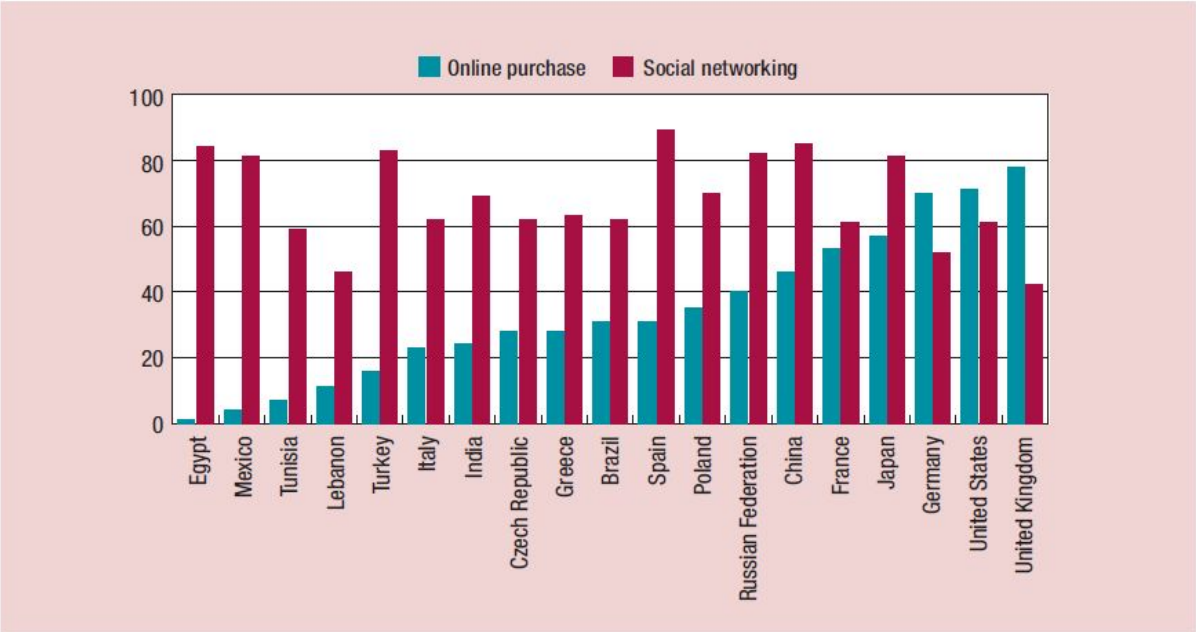


Source: Internet World Stats - www.internetworldstats.com/stats4.htm
Basis: 750,795,876 estimated Internet Users in Europe on July 2022



https://www.statista.com/chart/10878/where-people-cant-live-without-the-internet/

E il grado di alfabetizzazione «digitale»?



http://noisefromamerika.org/articolo/terra-social
Dati presi da UNCTAD Information Economy Report 2015, pp 18

Canali di accesso alla rete Internet

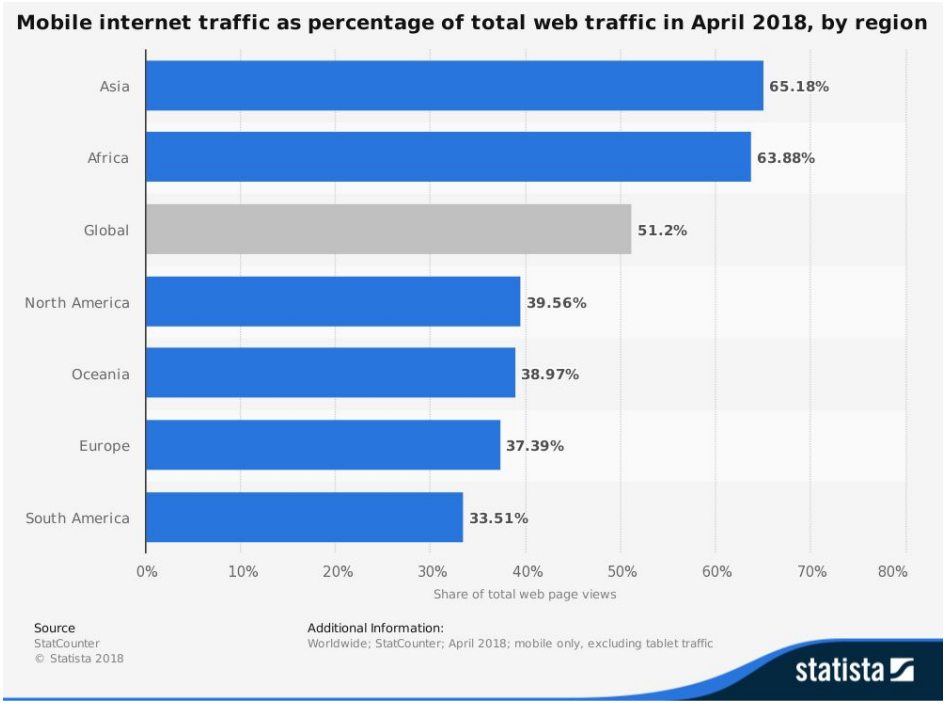
L'accesso alla rete Internet avviene sempre di più attraverso dispositivi mobili (smartphone e tablet, ma anche laptop).

Nel terzo mondo e in Asia il telefono cellulare rappresenta il punto di accesso principale alla rete (basso costo terminali, infrastruttura di rete mobile meno costosa rispetto a quella fissa).

Anche nei Paesi più sviluppati lo smartphone viene sempre più spesso usato come terminale di riferimento per l'accesso alla rete (portabilità, mobilità).

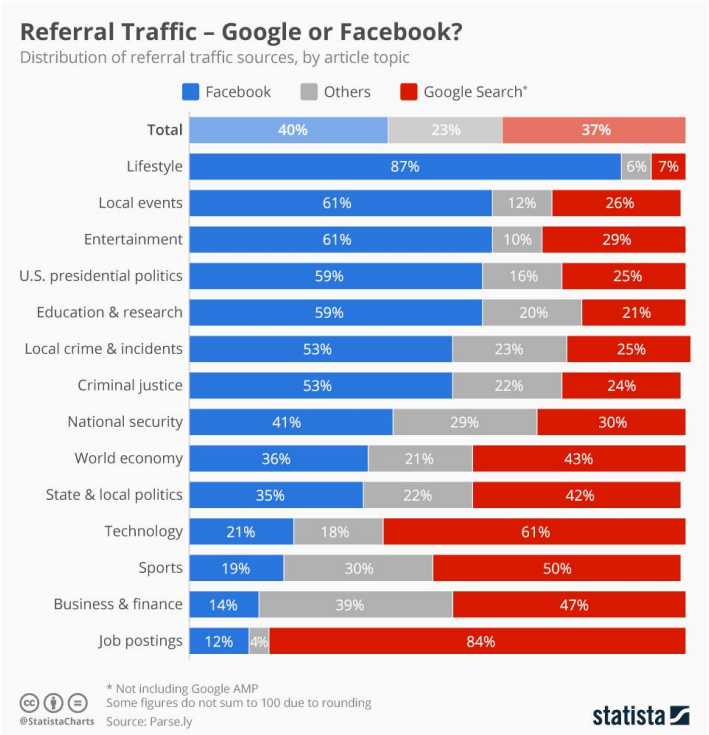
L'accesso alla rete Internet tramite dispositivi “limitati” e in mobilità pone particolari problemi a livello di progettazione dell'intero sistema.

Traffico effettuato da dispositivi *mobile* ad aprile 2018



https://www.statista.com/statistics/306528/share-of-mobile-internet-traffic-in-global-regions/

Punti di ingresso al Web



Nel 2017, Google e Facebook erano i due punti di ingresso principali al Web. (Ma sembra che Google abbia acquisito significative quote di mercato nel 2018.)

Quali conseguenze dal punto di vista della gestione delle fake news e della net neutrality?

https://www.statista.com/chart/9555/referral-traffic---google-or-facebook/

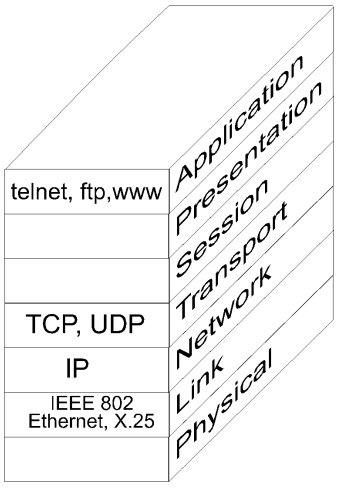
I Servizi del Livello Applicazione

Servizi di tre tipi fondamentali:

TERMINALE REMOTO (accesso a nodi remoti)

FILE TRANSFER (trasferimento file tra nodi diversi)

COMANDI REMOTI (**applicazioni**)
esecuzione di comandi remoti, anche specializzati, e riferimenti a servizi remoti
NEWS, MAIL, gopher, WWW



Proprietà fondamentali:

- Trasparenza
- Modelli Cliente/Servitore ed evoluzioni
- Standardizzazione

I Servizi del Livello Applicazione

Importante distinguere tra:

- **Programmi applicativi** (Client che si interfacciano con l'utente)
- **Protocolli applicativi** o **di trasferimento** (protocolli che regolano lo scambio dei messaggi tra Client e Server)
- **Protocolli** di specifica del **formato dei dati** (definiscono formato messaggi)
- **Protocolli di comunicazione** (protocolli di trasporto, TCP o UDP)

Esempio World Wide Web	Esempio telnet
Programma applicativo – Firefox, IE, etc.	Programma applicativo – telnet (comando)
Protocollo applicativo – HTTP	Protocollo applicativo – telnet (protocollo)
Formato dati – HTML	Formato dati – NVT
Protocollo di comunicazione – TCP	Protocollo di comunicazione – TCP

Altri programmi e protocolli: File Transfer Protocol – **ftp,** Simple Mail Transfer Protocol – **smtp,** etc.

telnet e rlogin

telnet e rlogin permettono il collegamento (sessione login) con macchina remota.

Il terminale locale diventa un terminale del sistema remoto

telnet standard in TCP/IP (Internet) (gestisce eterogeneità di S.O., HW, etc.)
rlogin per i sistemi UNIX (**r**emote **login**)

I **programmi applicativi** telnet e rlogin hanno interfaccia verso utente da linea di comando (sono comandi in Unix).

Il **protocollo di comunicazione** è TCP/IP

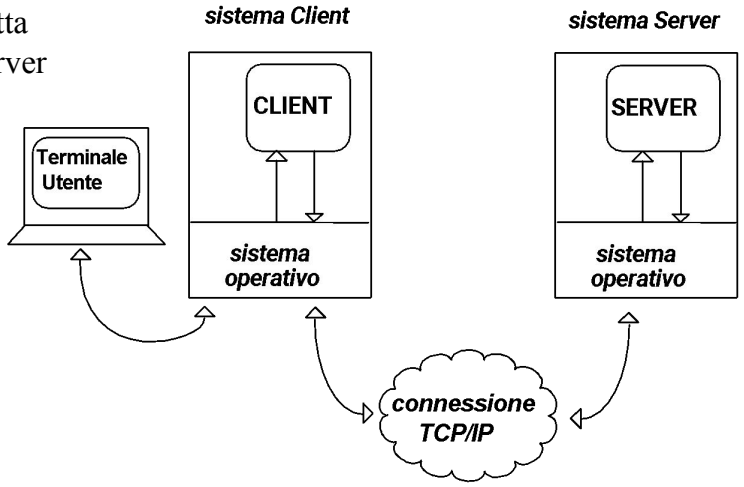
Attenzione: per motivi di sicurezza l'uso di telnet e rlogin è caldamente sconsigliato. Al loro posto si preferisce usare ssh (secure shell), una re-implementazione di rlogin che prevede la cifratura del canale di comunicazione.

Protocollo applicativo di telnet

Client stabilisce una connessione TCP con **Server** (porta 23)**,** quindi accetta caratteri da utente e li manda al Server e *contemporaneamente a*ccetta caratteri del server e li visualizza sul terminale utente.

Server accetta la richiesta di connessione da Client e inoltra dati da connessione TCP al sistema locale.

Standardizzato in molte RFC, a partire da RFC 15



Caratteristiche principali del protocollo applicativo di telnet:

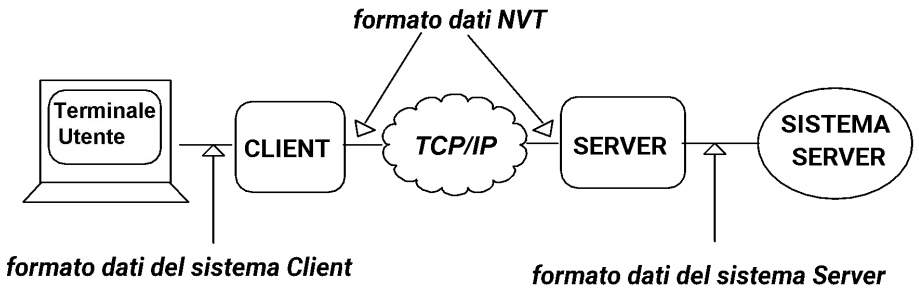
- **gestione eterogeneità** tramite interfaccia standard (**NVT**)
- **Client** e **Server negoziano** le opzioni del collegamento (ASCII a 7 bit o a 8 bit)
- comunicazione **simmetrica**

Network Virtual Terminal (NVT)

Problema nella rete Internet: mancanza di standardizzazione dei terminali. I terminali differiscono per il *set* di caratteri, la *codifica* dei caratteri, la *lunghezza* della linea e della pagina, i *caratteri di controllo* (individuati da diverse escape sequence), ecc.

Soluzione: definizione di un **terminale virtuale**, detto **NVT** (**N**etwork **V**irtual **T**erminal), cioè una standardizzazione di un terminale, con un formato definito e standardizzato di rappresentazione e codifica dei dati, dei caratteri di controllo, delle funzioni.

Il Client riceve l'input dal terminale utente, lo traduce in formato NVT e lo invia al Server, che a sua volta lo traduce nella propria rappresentazione nativa.



All'inizio del collegamento tra Client e Server, si usa rappresentazione a 7 bit US ASCII.

Caratteri di controllo NVT US ASCII

obbligatori; gli altri codici sono opzionali

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 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)

Invio di caratteri di controllo e funzioni di controllo insieme con i dati normali (cosiddetto in-band signaling). Problema se lo stream dei dati è pieno. (Comandi iniziano con 0xFF.)
Da Server verso Client, possibile uso di out-of-band signaling (con TCP urgent data).

Negoziazione

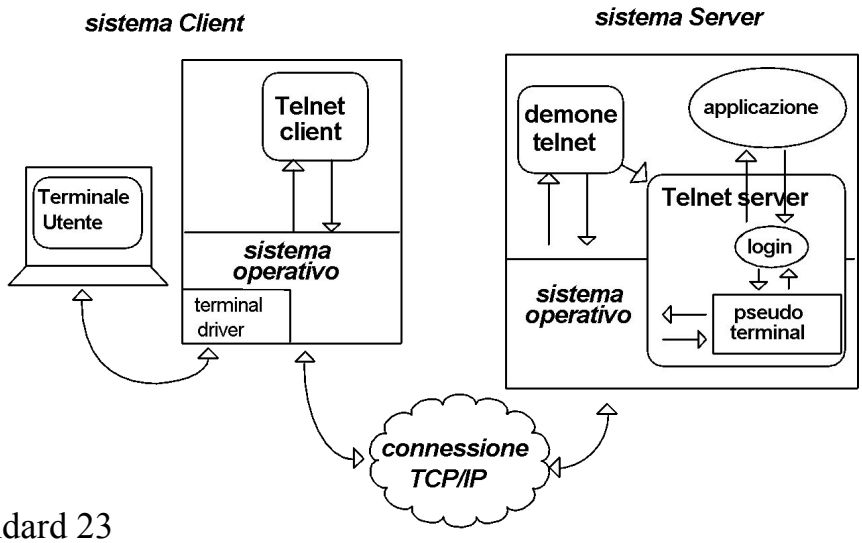
Tutti gli NVT supportano un insieme minimo di funzionalità, ma alcuni terminali hanno più funzioni rispetto al set minimo.

I due endpoint possono negoziare una serie di opzioni (funzionalità “extra” rispetto all’insieme di base) reciprocamente accettabili (set di caratteri, modalità eco, ecc.).

Possibilità di **negoziare** la connessione, sia alla *inizializzazione* sia s*uccessivamente* per selezionare le opzioni del protocollo di comunicazione (modalità a linee o a caratteri, set di caratteri, echo, emulazione terminale noto es. VT220, …).

Protocollo per negoziare le opzioni è **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**

Implementazione Telnet



Porta standard 23

Pseudo-terminal può essere una funzione del sistema operativo

Lo pseudo-terminal sarà esaminato con maggior dettaglio nel caso di rlogin

rlogin

Servizio di login remoto login su un'altra macchina UNIX (porta standard 513)

```
rlogin server.coimbra.br
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 (e permettere uso senza password).

Terminali in Unix

Unix è nato come SO multiutente per computer condivisi di grandi dimensioni a cui ciascun utente accedeva tramite un terminale fisico connesso (es. tramite seriale RS232) all'unità centrale. **Il supporto ai terminali è quindi uno dei componenti fondamentali di Unix** [The Linux Programming Interface, cap. 62; Advanced Programming in the UNIX Environment (3rd Ed.), cap. 18 e 19].

Storicamente, un grande numero di tipologie di terminali fisici veniva connesso a macchine Unix. La mancanza di standardizzazione rendeva difficile scrivere programmi portabili che utilizzassero le funzionalità del terminale. Sono nati quindi i database **termcap** e **terminfo**, che descrivono come eseguire varie operazioni di controllo dello schermo per un'ampia varietà di terminali, e le librerie **curses** per la realizzazione di TUI (terminal user interface). Ancora oggi, questi tre componenti sono installati su tutti i PC Unix.

Successivamente, il mercato è andato verso l'uso di pochi formati di terminale standard, come il famoso VT100 di Digital. Al giorno d'oggi, si usano terminali virtuali, come xterm (o xterm-256color).

Jurassic ~~Park~~ Terminal

Ad esempio, questo è il terminale Lear Siegler ADM-3A su cui Bill Joy ha sviluppato l'editor di testo vi.

(Si notino le frecce sui tasti HJKL.)

https://twobithistory.org/2018/08/05/where-vim-came-from.html

Caratteristiche rlogin

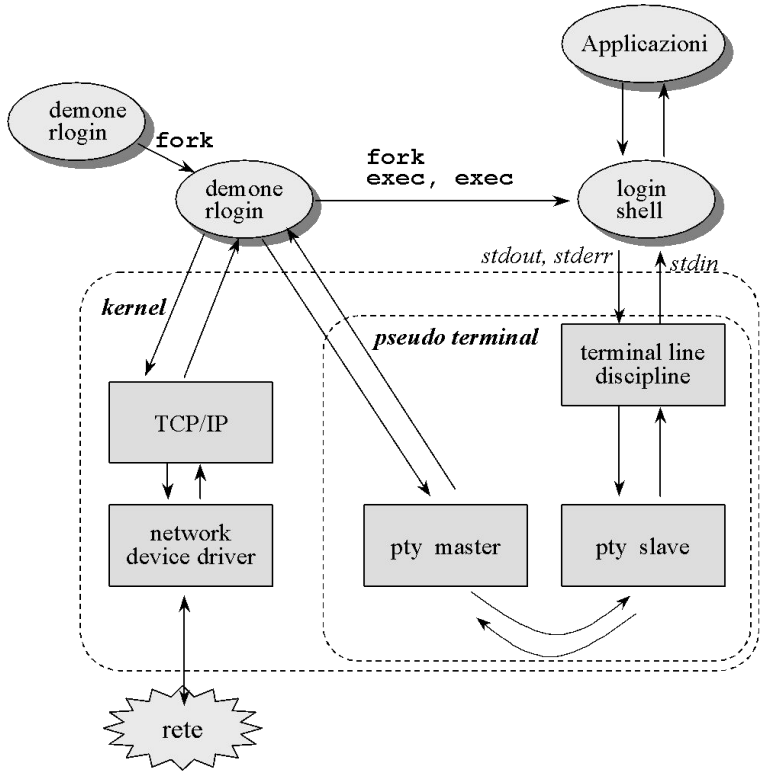
Lavorando solo con macchine Unix, **rlogin non ha bisogno di NVT**:

- esporta l'ambiente del Client (i.e., il tipo di terminale, TERM, che sostanzialmente è una entry nel database termcap e terminfo) verso il Server
- conosce l'ambiente di partenza e quello di arrivo, ha nozione di stdin, stdout e stderr (collegati al client mediante TCP).
- utilizza una sola connessione TCP/IP
- flow control: il Client tratta **localmente** i caratteri di controllo del terminale (ctrl-S e ctrl-Q fermano e fan ripartire l'output del terminale, senza attendere l'invio del carattere al server)

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). **out-of-band** signaling implementato con TCP urgent mode.

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

Implementazione e pseudoterminal



Il supporto agli pseudo terminali di Unix permette implementazioni molto semplici di rlogin [The Linux Programming Interface, cap. 64]

Secure Shell (SSH)

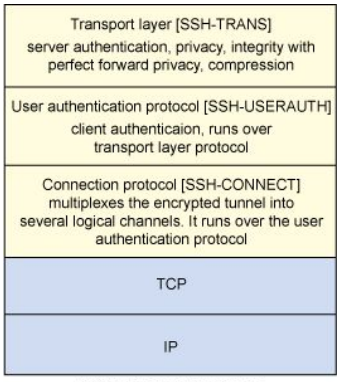
Secure Shell (SSH) è una reimplementazione sicura di rlogin, concepita e progettata per offrire la massima protezione durante l'accesso remoto a un altro host sulla rete.

Comunicazione su un canale cifrato, con uso di protocolli di sicurezza molto robusti e supporto a tecniche crittografiche state-of-the-art.

Al di sopra di SSH sono definite funzioni come Secure Copy (SCP), Secure File Transfer Protocol (SFTP), X session forwarding e port forwarding (per incapsulare il traffico di altri protocolli non sicuri).

SSH è un servizio utilizzatissimo, in pratica lo standard (sia de facto che de iure) per l'amministrazione dei sistemi remoti (anche in Cloud).

Secure Shell (SSH)



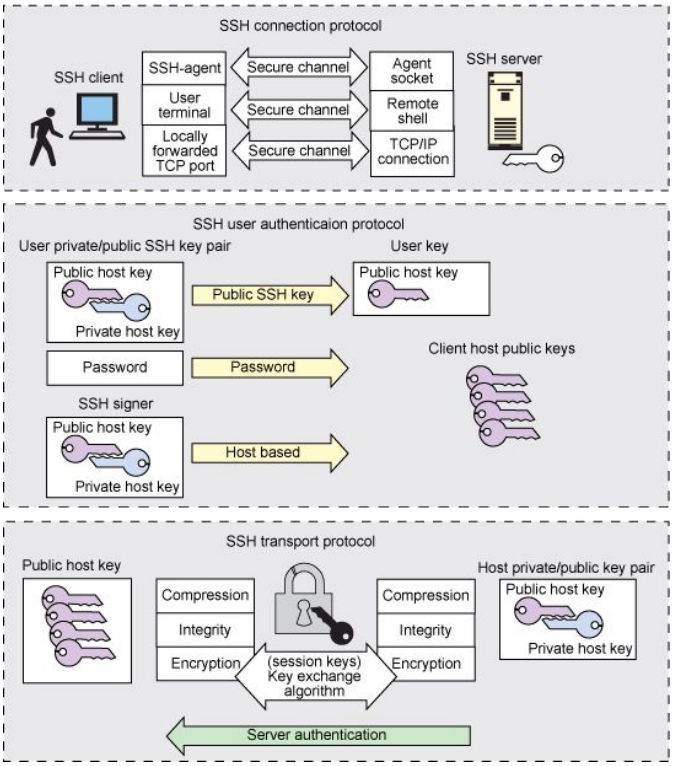
SSH protocol logical layers

SSH ha 3 componenti (layer) principali:

Transport Layer Protocol: consente l'autenticazione, la privacy e l'integrità del server con una perfetta forward secrecy. Può fornire compressione opzionale e viene eseguito su una connessione TCP.

User Authentication Protocol: autentica il Client sul Server e viene eseguito sul livello di trasporto.

Connection protocol: esegue il multiplexing del tunnel crittografato su numerosi canali logici, al di sopra dello User Authentication Protocol.



Courtesy: https://developer.ibm.com/articles/au-sshsecurity/

SSH implementa un sistema piuttosto sofisticato di gestione delle chiavi e dei relativi agenti.

Supporto ad autenticazione a chiave pubblica (es. come su GitHub)

Supporto a stoccaggio sicuro delle chiavi (es. su Yubikey) e a multi-factor authentication

SSH Agent

SSH Agent

I Client SSH in genere vengono eseguiti per la durata di una sessione di accesso remoto e sono configurati per cercare la chiave privata dell'utente in un file nella directory home dell'utente (ad esempio, .ssh/id_rsa).

Per una maggiore sicurezza, è comune archiviare la chiave privata in una forma crittografata, in cui la chiave di crittografia viene calcolata da una passphrase che l'utente ha memorizzato. Poiché la digitazione della passphrase può essere noiosa, molti utenti preferirebbero inserirla solo una volta per sessione di accesso locale.

Pertanto, gli utenti eseguono un programma chiamato ssh-agent che viene eseguito oltre la durata di una sessione di accesso locale, memorizza le chiavi non crittografate in memoria e comunica con i client SSH utilizzando una socket Unix.

Trasferimento file (ftp)

ftp (file transfer protocol) stabilisce un collegamento con una macchina remota per il trasferimento (upload e download) di file. Come per telnet, ci sono problemi di **sicurezza**, legati alla trasmissione in chiaro delle password. Possibile uso di FTP over SSL o di alternative sicure come **scp** (trasferimento di file su canale ssh).

- Vari **programmi applicativi**, da linea di comando o con interfaccia grafica.
- **Protocollo di comunicazione** TCP
- **Protocollo ftp** con comandi (4 caratteri ASCII a 7 bit) e risposte (numeri a 3 cifre). Esempi di comandi protocollo ftp:
  STOR local-file trasferisce un file locale sulla macchina remota
  RETR remote-file trasferisce un file remoto sul disco locale

Utente utilizza varie interfacce (ad es. linea comandi, con put (esegue STOR), get (esegue RETR), mget e mput, help, dir, ls, cd, lcd, ...

ftp usa due connessioni per ogni collegamento client/server, una di CONTROLLO (su porta 21) e una di DATI (su porta 20). Per questo, in alcuni testi si dice che le informazioni di controllo sono trasmesse fuori out-of-band (ma è situazione diversa da rlogin…).

Il Server mantiene lo **stato** della sessione del Client.

Comandi Principali

USER: Username
PASS: Password.
PORT: Per entrare in modalità attiva
PASV: Per entrare in modalità passiva
QUIT: Log out
ABOR: Cancella comando precedente
PWD: Stampa directory corrente sul Server
CWD: Cambia directory corrente lato Server
CDUP: Vai alla directory padre lato Server (cd ..).
DELE: Cancella file
LIST: Ottieni lista dei file
NLIST: Ottieni lista dei file ma senza attributi
MKD: Crea una directory sul Server
RMD: Cancella una directory sul Server
RNTO: Rinomina file sul Server
RETR: Scarica file dal Server
STOR: Carica file sul Server
APPE: Come STOR ma in modalità append se file esiste
… (si veda https://en.wikipedia.org/wiki/List_of_FTP_commands)

Notate che questi comandi, definiti all'interno del protocollo applicativo, cambiano lo stato dell'interazione tra Client e Server. **FTP è un protocollo stateful.**

FTP – Codifica risposte

Le **risposte** sono codificate con 3 cifre, e un testo descrittivo:

La prima cifra codifica le interazioni
1xx Positive Preliminary reply
2xx Positive Completion reply
3xx Positive Intermediate reply
4xx Transient Negative Completion reply (il comando può essere ripetuto)
5xx Permanent Negative Completion reply
6xx Protected reply

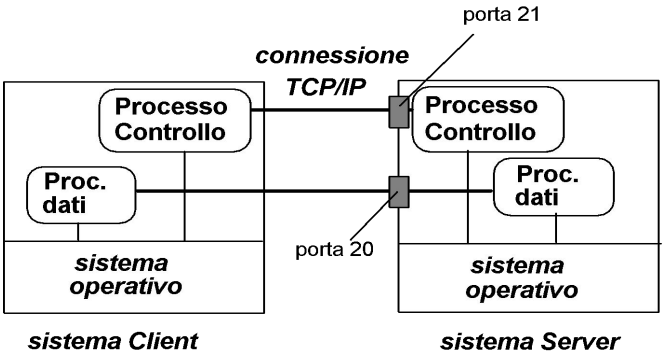
La seconda cifra codifica le risposte
x0x Sintassi
x1x Informazione
x2x Connessione
x3x Autenticazione e accounting
x4x Non specificato
x5x File system

La terza cifra specifica più precisamente

Codici con semantica ben definita, per permettere ai programmi Client di capire per quale motivo un comando eventualmente non è andato a buon fine.

Troveremo uno schema simile in SMTP (E-Mail) e HTTP (Web).

Implementazione FTP



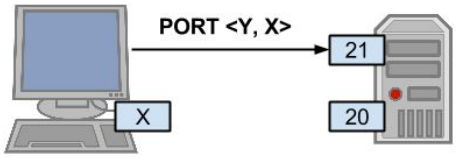
Un processo **master** del server attende connessioni (processo **ftpd**, demone di ftp) e si crea uno **slave** per ciascuna connessione.

Possibili diverse implementazioni slave: singolo o multi processo, ma sempre su diverse connessioni TCP (controllo e dati).

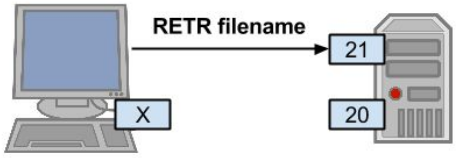
Caso singolo: un processo gestisce le due connessioni controllo e dati.
Caso multi: un processo gestisce la connessione dati, uno quella di controllo.

Chi esegue le active e passive open sulle socket?

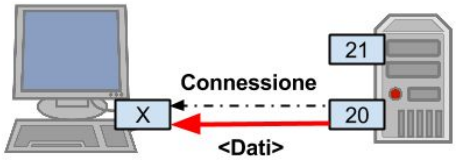
FTP Attivo



Per prima cosa, il client contatta il server sulla porta 21 (comandi) e gli comunica il proprio indirizzo IP Y e il numero X della propria porta dati.

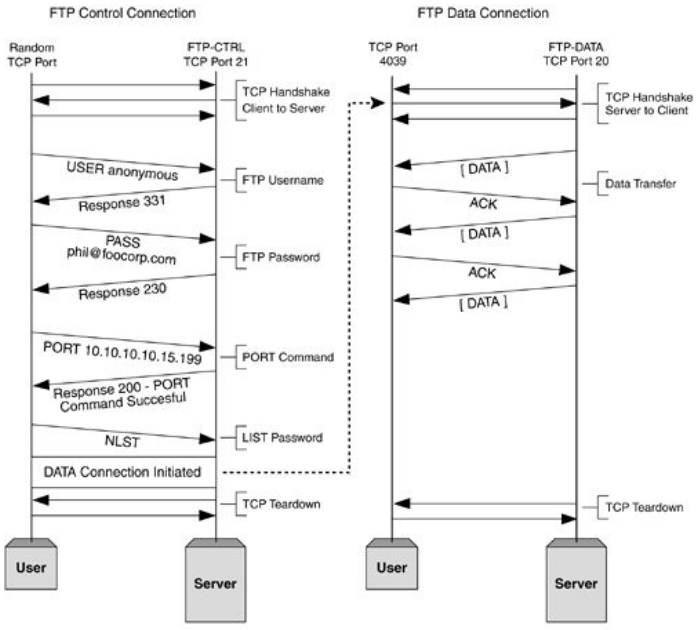


Dopodiché, Il client può comunicare con il server sulla porta comandi per richiedere il trasferimento di file da server a client (RETR) o da client a server (STOR).



In risposta a ciascun comando di trasferimento, il server apre una connessione alla porta X (dati) del client e trasmette (RETR) o riceve (STOR) il contenuto del file richiesto dal client.

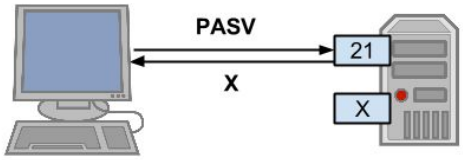
FTP Attivo



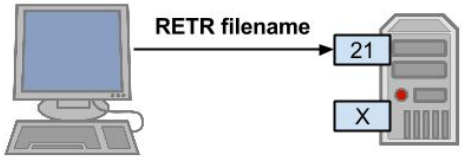
In an active FTP connection, the client will use the Control connection to tell the server, via a PORT command, which IP address and TCP port it should establish the Data connection to. The server then opens a Data connection to that IP address and port using the well-known Port 20 as the source.

Courtesy: M. Syme, P. Goldie, “Optimizing Network Performance with Content Switching”, Pearson, 2004.

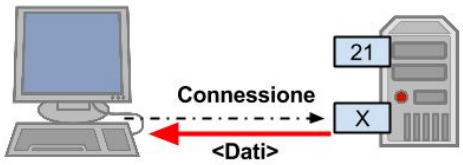
FTP Passivo



Per prima cosa, il client contatta il server sulla porta 21 (comandi) e gli chiede di entrare in modalità passiva. In risposta, il server apre una porta dati X (numero > 1023 scelto casualmente e a run-time) e comunica tale valore al client.

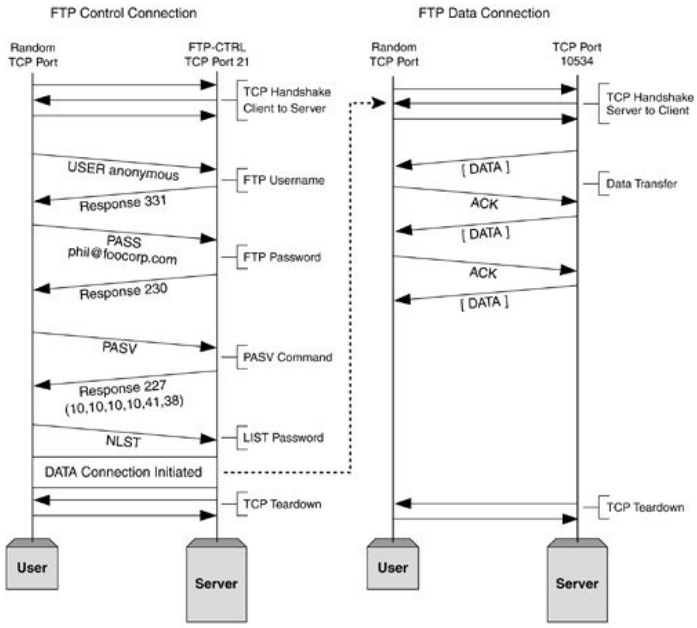


Dopodiché, Il client può comunicare con il server sulla porta comandi per richiedere il trasferimento di file da server a client (RETR) o da client a server (STOR).



In risposta a ciascun comando di trasferimento, il server si mette in ascolto sulla porta X (dati) per trasmettere (RETR) o ricevere (STOR) il contenuto del file richiesto dal client.

FTP Passivo



In an Passive FTP session, the client will use the Control connection to request from the server, via a PASV command, which IP address and TCP port it should establish the Data connection to. The client then opens a Data connection to that IP address and port using a random TCP port as the source.

Courtesy: M. Syme, P. Goldie, “Optimizing Network Performance with Content Switching”, Pearson, 2004.

Problemi di FTP

FTP presenta numerosi problemi, al punto che potrebbe forse essere indicato come esempio di come ***NON*** progettare servizi Internet.

In FTP attivo il Client comunica il proprio indirizzo IPv4 al Server come argomento del comando PORT, un errore di design davvero eclatante. Infatti, scrivere un indirizzo di livello 3 in un protocollo di livello 7 rappresenta una gravissima (!!!) violazione del principio di layering alla base del modello ISO/OSI con serie conseguenze per il protocollo applicativo. In seguito all'arrivo di IPv6, si è dovuto estendere il protocollo con due comandi (EPRT ed EPSV) che sostituissero i comandi PORT e PASV.

Inoltre, la violazione del layering causa problemi nel caso (praticamente certo) in cui il Client si trovi in un ambiente di rete in cui è vengono usati indirizzi IPv4 privati e NAT, poiché il Server riceve dal Client un IP privato che ovviamente non è raggiungibile.

Problemi di FTP

In FTP passivo, il Server deve essere raggiungibile su un range di porte piuttosto esteso.

Questo crea molti problemi alla configurazione del firewall, in quanto non si può restringere il traffico in ingresso al Server a un insieme limitato di porte – come invece richiederebbero le basilari best practice in ambito sicurezza.

Se un utente sbadato o un attaccante aprisse un server insicuro su una delle porte nel range utilizzato da FTP non ci sarebbe alcuna protezione contro eventuali attacchi da parte del firewall.

In generale, FTP rappresenta un perfetto case study su come ***NON*** progettare protocolli applicativi.

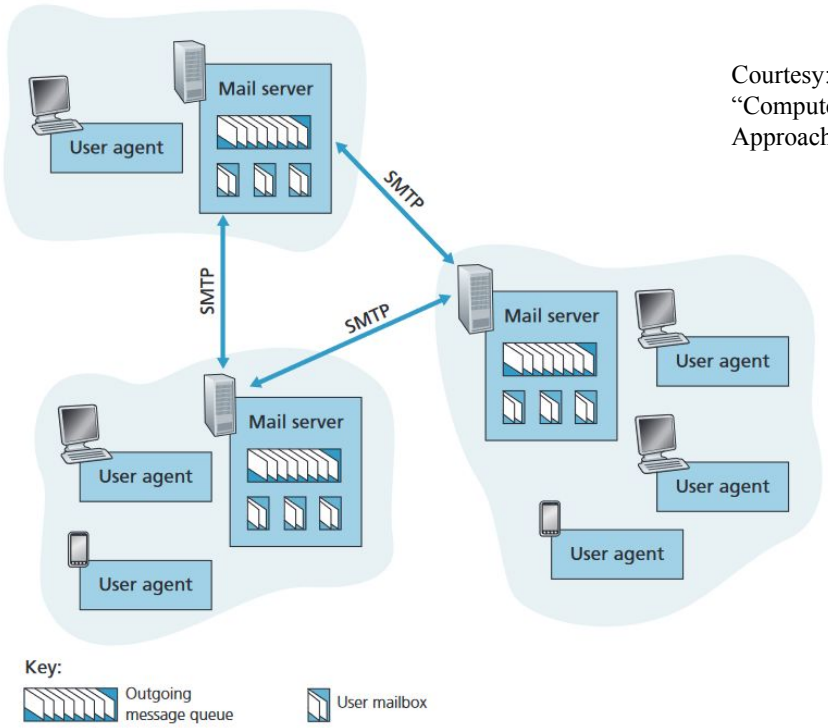
La posta elettronica

Il servizio di posta elettronica permette lo scambio di e-mail tra utenti in Internet. È un servizio **asincrono** (a differenza di telnet ed ftp), anche se si appoggia su TCP (porta 25).

Componenti del sistema di posta:

- **Programmi applicativi** – Thunderbird, Eudora, Outlook, etc.
- **Protocollo applicativo** o **di trasferimento** – SMTP
- **Protocolli di accesso alle caselle** di posta – POP3, IMAP
- **Protocolli** di specifica del **formato dei dati** – RFC822, MIME, etc.
- **Protocollo di comunicazione** – TCP

La posta elettronica



Courtesy: J. Kurose, K. Ross, "Computer Networking: A Top-Down Approach" (8th Ed.), Pearson, 2021.

Architettura del servizio di posta elettronica

I componenti principali sono:

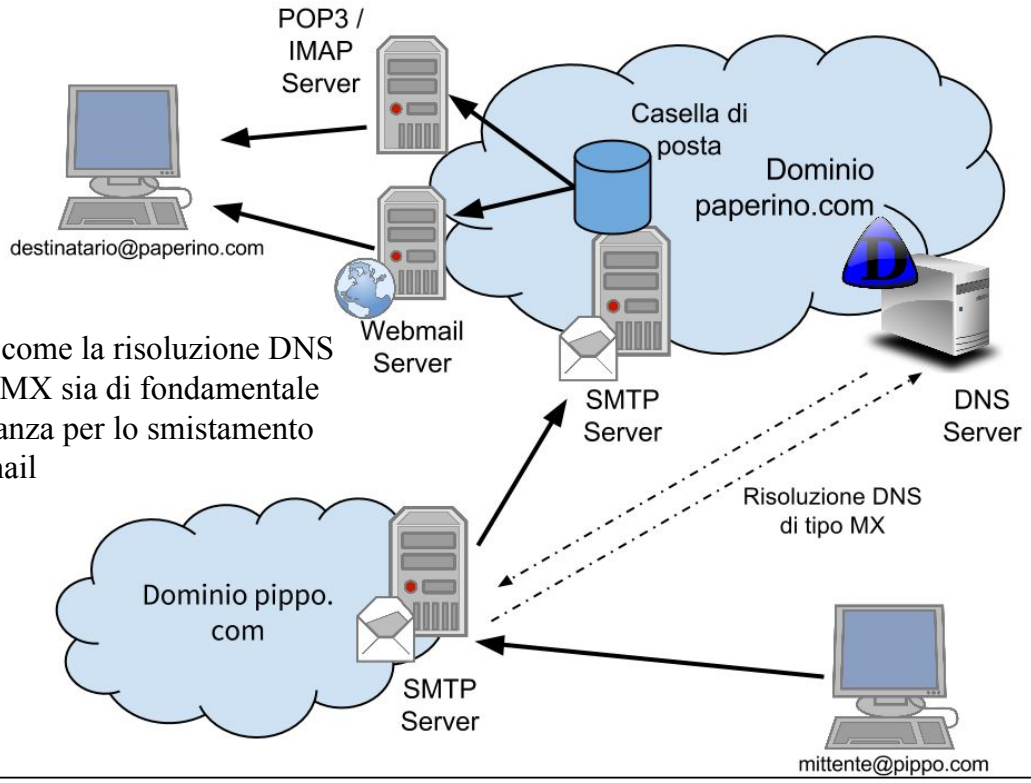
- **User Agent (UA),** che si interfacciano con l'utente
- **Mail Transfer Agent (MTA),** i server di posta

E-mail parte da UA mittente e arriva a UA destinazione attraverso la rete dei MTA. Utilizzo del protocollo di trasferimento **SMTP** per la consegna della posta alla casella del destinatario (**modello push**):

- MTA usano SMTP
- UA in trasmissione usa SMTP

Utilizzo di protocolli specifici, come POP3 o IMAP (o, ancor meglio, delle loro versioni «sicure» POP3s e IMAPS), per la ricezione della posta, ovverosia per il trasferimento della posta dalla casella di posta dell'utente allo UA che ne permette la consultazione (**modello pull**).

Architettura del sistema di posta elettronica



Si noti come la risoluzione DNS di tipo MX sia di fondamentale importanza per lo smistamento della mail

Posta elettronica e DNS

Il funzionamento del sistema di posta elettronica dipende dal supporto offerto dalla risoluzione di nomi di tipo **Mail eXchange (MX)** del DNS.

In particolare, attraverso la risoluzione di nomi di tipo MX, il DNS permette a un Server SMTP di scoprire **i nomi logici dei Server SMTP di un determinato dominio** (ad esempio il dominio di un indirizzo di destinazione di un'e-mail).

Senza il supporto alla risoluzione MX, sarebbe impossibile scoprire quale sia il Server SMTP di un particolare dominio.

Si noti che i Server SMTP di un dominio sono tipicamente replicati. Più precisamente, abbiamo **almeno 2 Server: 1 master e 1+ slave**.

Esempio risoluzione MX per dominio unife.it

```
kepler ~ dig -t mx unife.it
;;...

;; ANSWER SECTION:
unife.it.         60    IN    MX    10 aspmx3.googlemail.com.
unife.it.         60    IN    MX    1 aspmx.l.google.com.
unife.it.         60    IN    MX    5 alt1.aspmx.l.google.com.
unife.it.         60    IN    MX    5 alt2.aspmx.l.google.com.
unife.it.         60    IN    MX    10 aspmx2.googlemail.com.

;; ADDITIONAL SECTION:
aspmx.l.google.com.    29    IN    A    74.125.195.27
aspmx.l.google.com.    29    IN    AAAA    2607:f8b0:400e:c03::1b
alt1.aspmx.l.google.com. 29    IN    A    74.125.129.27
alt1.aspmx.l.google.com. 29    IN    AAAA    2607:f8b0:4001:c15::1a
alt2.aspmx.l.google.com. 29    IN    A    173.194.219.26
alt2.aspmx.l.google.com. 29    IN    AAAA    2607:f8b0:4002:c03::1a
```

Formato dei messaggi (RFC 822)

Un messaggio, un e-mail, è composto da un **header** e da un **body**.

Header. L'intestazione è composta da una serie di righe, ognuna composta da un tipo:valore, tra i quali:

`From:` indirizzo mittente
`To:` mailbox destinatario (anche più destinatari)
`Date:` data di spedizione
`Subject:` soggetto del messaggio
`Cc:` copia destinatari
`Replay-To:` indirizzo per la risposta
`Message-Id:` identificatore unico del messaggio

Body. Il corpo del messaggio è il testo dell'e-mail, in formato ASCII

Indirizzi di posta elettronica:
destinatario come username e nome provider `mauro.tortonesi@unife.it`

pseudonimi (aliases) e **mail forwarding**

Formato dei messaggi (MIME)

La RFC 822 standardizza messaggi di solo testo. **MIME** estende RFC 822 per consentire invio di dati multimediali (audio, immagini, video, documenti word, etc.), *che devono essere transcodificati in un formato US-ASCII compatibile*.

MIME ha 3 parti fondamentali:

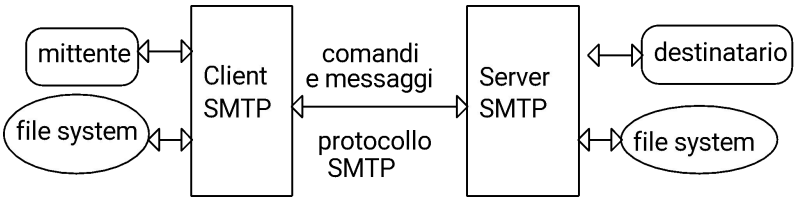
1) Righe di intestazione (integrano header di RFC822) per definire il tipo di dato.
   `MIME-version:... Content-Description:...`
   `Content_Type:... Content-Transfer-Encoding:...`

2) Definizione dei tipi di dati contenuti (Content-Type).
   `image/jpeg, text/plain, application/postscript, application/msword`

3) Indica la transcodifica (Content-Transfer-Encoding) usata per i vari dati, in modo che possano essere trasmessi in una e-mail (che usa SOLO caratteri ASCII). Conversioni di formati binari in ASCII (ad esempio, tramite codifica *base64*; per conversioni da testo UTF-8 ad ASCII spesso si usa codifica *quoted-printable*).

Protocollo applicativo SMTP (RFC 2821)

Standard per il trasferimento della mail, con messaggi codificati tra client e server.



Comandi cliente Risposte server (Esempio**)**

sender `MAIL FROM: nome mittente`
receiver `250 OK`

sender `RCPT TO: nome destinatario`
receiver `250 OK` abilitato

sender `DATA`
`linee di testo del messaggio`
sender `< cr-lf>.< cr-lf>` fine messaggio
receiver `250 OK'`

COMANDI: parole, di caratteri ascii
RISPOSTE: codice di 3 cifre e testo

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

Protocollo SMTP – Codifica comandi e risposte

I **comandi** sono stringhe ASCII, tra cui: HELO, MAIL, RCPT, DATA, QUIT

Le **risposte** sono codificate con 3 cifre, e un testo descrittivo:

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)

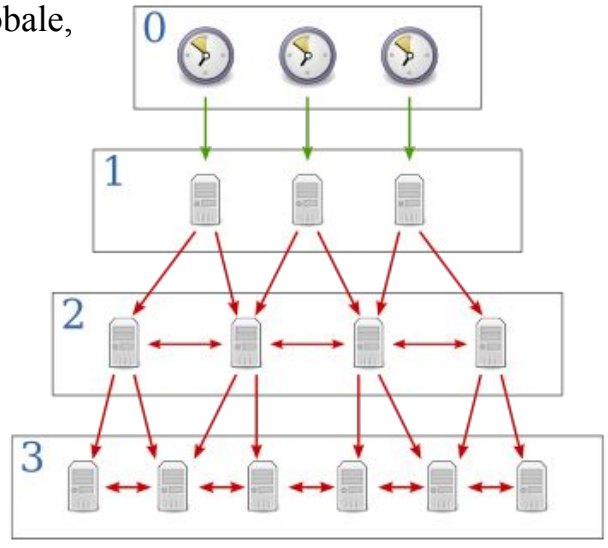
La terza cifra specifica più precisamente

Esempio di dialogo

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
S: QUIT
R: 221 Closing connection

SINCRONIZZAZIONE CLOCK
Network Time Protocol (NTP)

NTP permette di sincronizzare il clock tra una molteplicità di nodi su scala globale, con accuratezza di pochi millisecondi (uso di Precision Time Protocol, PTP, invece per applicazioni che richiedono accuratezza sub msec, su reti locali)



Organizzazione gerarchica di server su diversi livelli (strata), con pochi server di livello più alto sincronizzati con orologi precisissimi (es. cesio) e "ufficiali" e bassi carichi di servizio, e molti server di livello più basso che sopportano carichi più elevati.

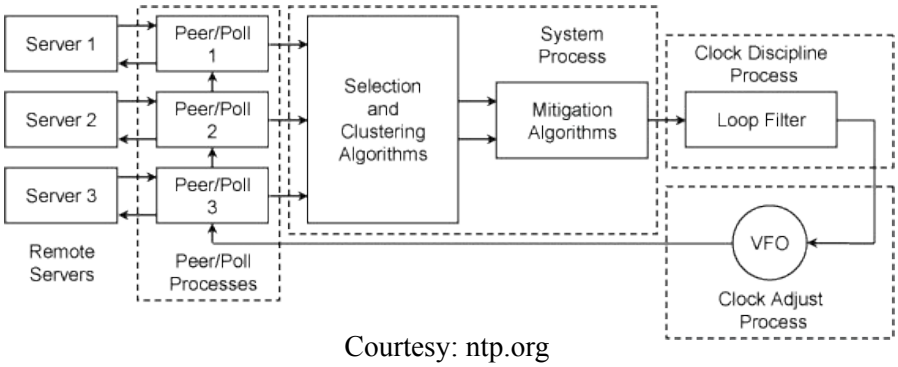
Network Time Protocol (NTP)

NTP usa transazioni di richiesta clock, in cui un Client richiede l'ora corrente da un Server, inserendo il proprio tempo con la richiesta. Il server aggiunge il suo tempo al pacchetto di dati e lo ritrasmette al client.

Quando riceve il pacchetto, il Client può ricavare due informazioni essenziali: il tempo di riferimento al Server e il tempo trascorso, misurato dall'orologio locale, affinché un segnale passi dal Client al Server e viceversa. Iterazioni ripetute di questa procedura consentono al Client di rimuovere gli effetti del jitter di rete e quindi di ottenere un valore stabile per il ritardo tra l'orologio locale e lo standard dell'orologio di riferimento sul Server.

Questo valore può quindi essere utilizzato per regolare l'orologio locale in modo che sia sincronizzato con il Server. Ulteriori iterazioni di questo scambio di protocollo possono consentire al client locale di correggere continuamente l'orologio locale per indirizzare lo sfasamento dell'orologio locale.

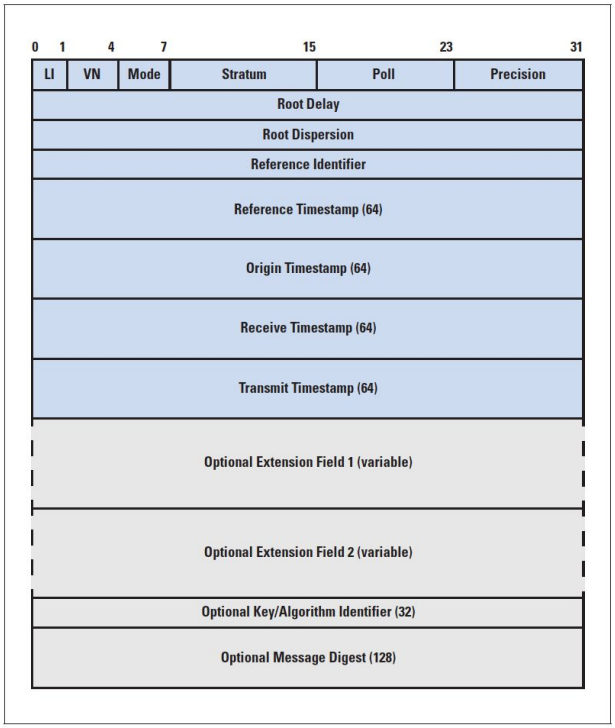
Network Time Protocol (NTP)



Courtesy: ntp.org

Sincronizzazione con molti Server allo stesso tempo e sintesi delle informazioni ricevute con procedure ben definite e resistenti alla perdita di pacchetti.

Network Time Protocol (NTP)



NTP usa UDP sulla porta 123.

Il server NTP è stateless e risponde a ciascun pacchetto NTP del client ricevuto in modo transazionale semplice aggiungendo campi al pacchetto ricevuto e ritrasmettendo il pacchetto al mittente originale, senza riferimento a precedenti transazioni NTP.

Courtesy: https://labs.apnic.net/?p=462

INSTANT MESSAGING
eXtensible Messaging and Presence Protocol (XMPP)

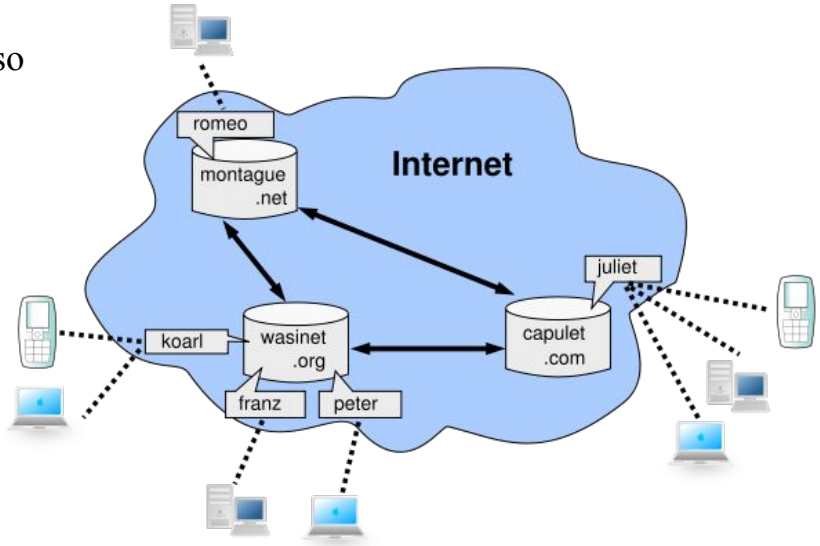
XMPP è un protocollo che permette la comunicazione real-time di messaggi e di informazioni di stato (assente, occupato, ecc.).

I client si registrano presso server locali.

Routing di messaggi e architettura simile a SMTP.

RFC 3920, 3921,…

Porta standard TCP 5222



Applicazioni di XMPP

XMPP è stato pensato per molti tipi di applicazioni:

- Instant messaging
- Group chat
- Gaming
- Controllo di sistemi
- Location-based services
- Middleware e cloud computing
- Data syndication (RSS, aggiornamenti di stato su siti Web 2.0)
- VoIP (Google Voice)
- Servizi di identità

Nonostante XMPP (una volta usato da numerosi servizi come MSN, ICQ/AIM, Skype, Facebook Messenger) sia recentemente "uscito di moda", rappresenta ancora oggi una tecnologia matura e piuttosto interessante: https://xmpp.org/about/myths.html

Protocollo XMPP

Lo scambio di messaggi in XMPP è asincrono. Il client apre una sessione con il server, gli comunica la propria presenza, e poi può inviare e ricevere messaggi.

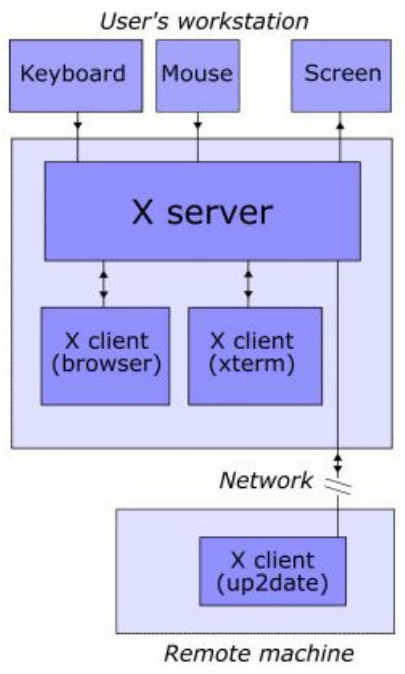
Il protocollo di comunicazione usato da XMPP è basato su XML. Esempio:

```
Client: <stream:stream>
Client: <presence/>
Client: <message from="studente@unife.it"
                 to="mauro.tortonesi@unife.it">
          <body>Quando si terrà l'esame scritto?</body>
        </message>
Server: <message from="mauro.tortonesi@unife.it"
                 to="studente@unife.it">
          <body>Martedì prossimo.</body>
        </message>
Client: </stream:stream>
```

X Window

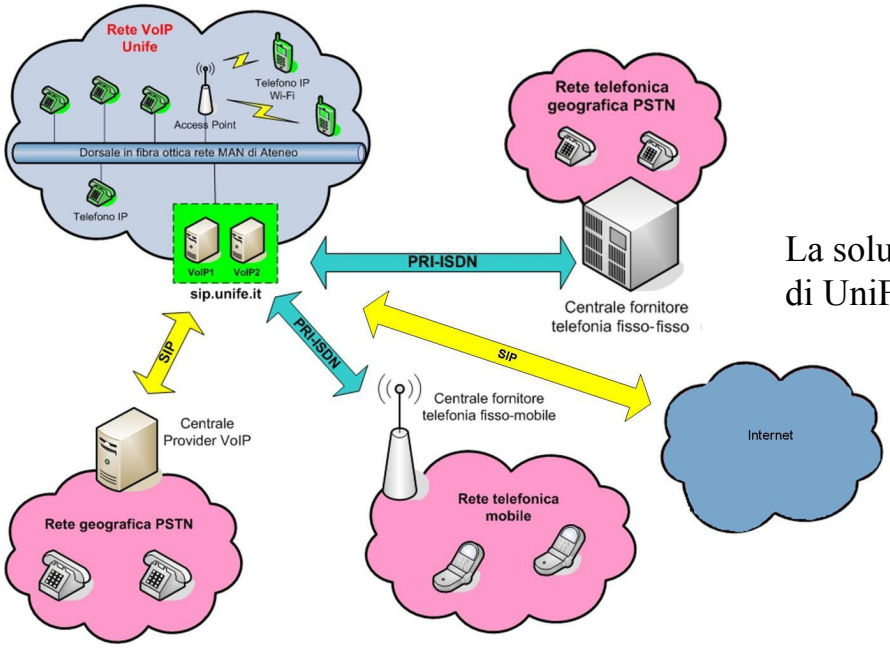
X Window è un sistema per l'accesso in modalità grafica ad applicazioni che risiedono su macchine remote.

Architettura client-server con inversione dei ruoli: il server è la macchina locale e il client è la macchina su cui eseguono le applicazioni remote.



VoIP e Multimedia Streaming

SIP e RTP vengono usati come protocolli di controllo e di trasporto dei dati in applicazioni Voice over IP (VoIP) e di Multimedia Streaming.



La soluzione VoIP di UniFE

Real-Time Transport Protocol (RTP)

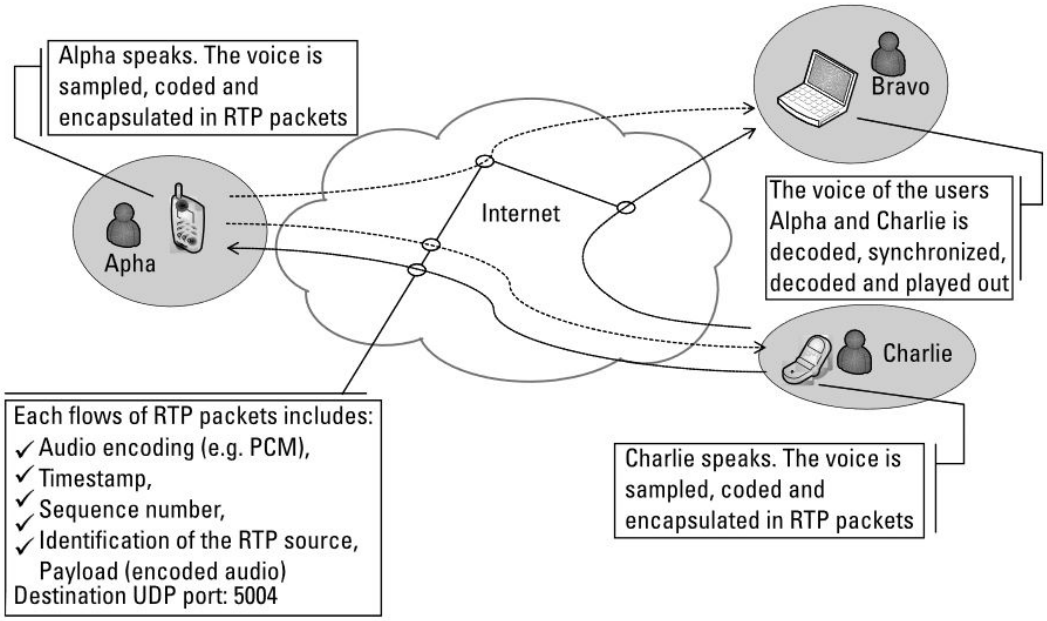
Real-time Transport Protocol (RTP) è un protocollo utilizzato per servizi di comunicazione in tempo reale su Internet, come audio e video interattivi.

RTP fornisce diverse funzioni:

- identificazione del payload type
- numerazione sequenziale
- timestamping
- monitoring
- …

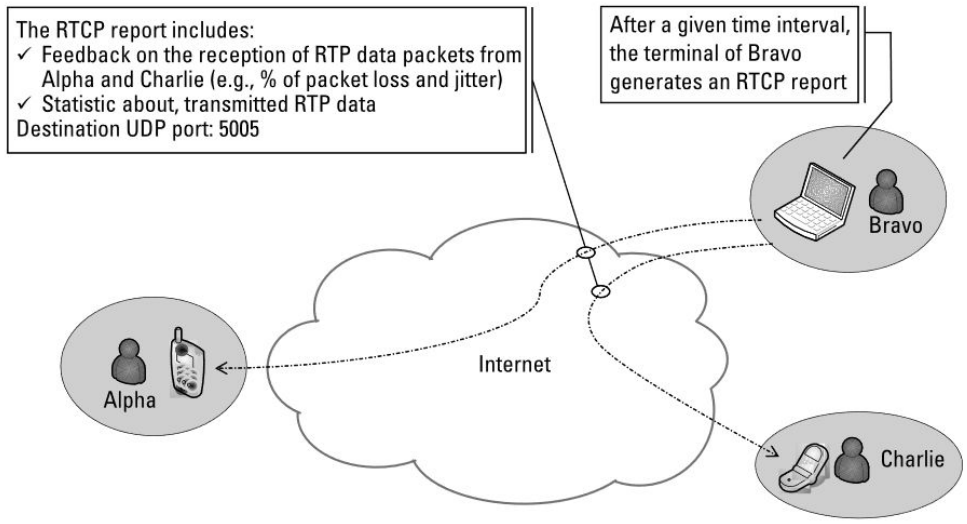
Basato sul protocollo UDP, viene usato con RTP Control Protocol (RTCP) che monitora la qualità del servizio.

Real-Time Transport Protocol (RTP)



Courtesy: Vidal et al., «Multimedia Networking: Technologies, Protocols and Architectures», Artech House, 2019

RTP Control Protocol (RTCP)



Courtesy: Vidal et al., «Multimedia Networking: Technologies, Protocols and Architectures», Artech House, 2019

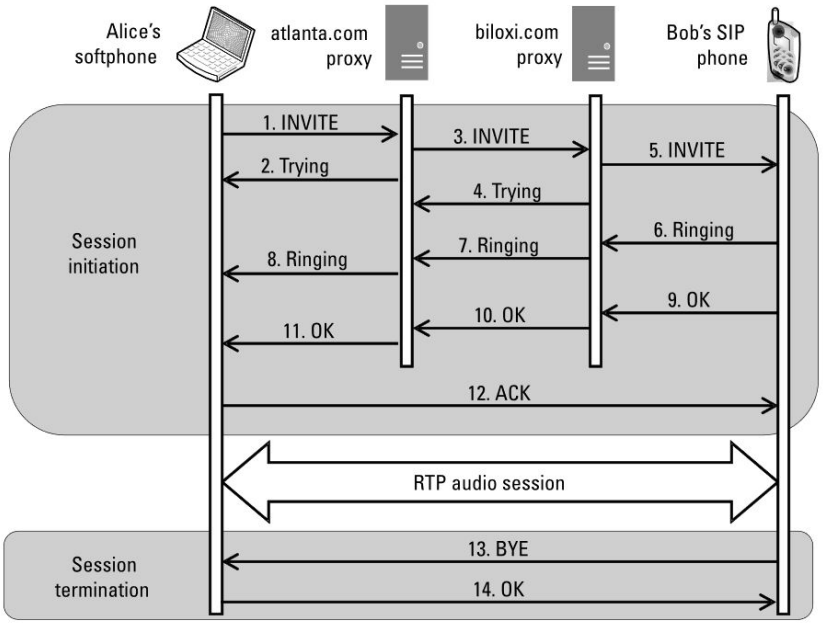
Session Initiation Protocol (SIP)

Session Initiation Protocol (SIP) è un protocollo applicativo usato per creare, modificare, e terminare sessioni interattive di comunicazione tra uno o più partecipanti, come chiamate VoIP, multimedia streaming, e videoconferenze.

SIP instaura o termina chiamate video o vocali e permette di modificare le caratteristiche di chiamate in corso (es. indirizzo IP e porta), di invitare ulteriori partecipanti e aggiungere o cancellare stream multimediali. Tradizionalmente basato su UDP (porta 5060), recentemente viene utilizzato anche su TPC e TLS.

Elementi principali: user agent, proxy server, registrar, redirect server, session border controller, gateway. URI identificano ogni elemento di una rete SIP: `sip:1-555-123-4567@myisp.com`

Session Initiation Protocol (SIP)



Courtesy: Vidal et al., «Multimedia Networking: Technologies, Protocols and Architectures», Artech House, 2019