Impianti Informatici

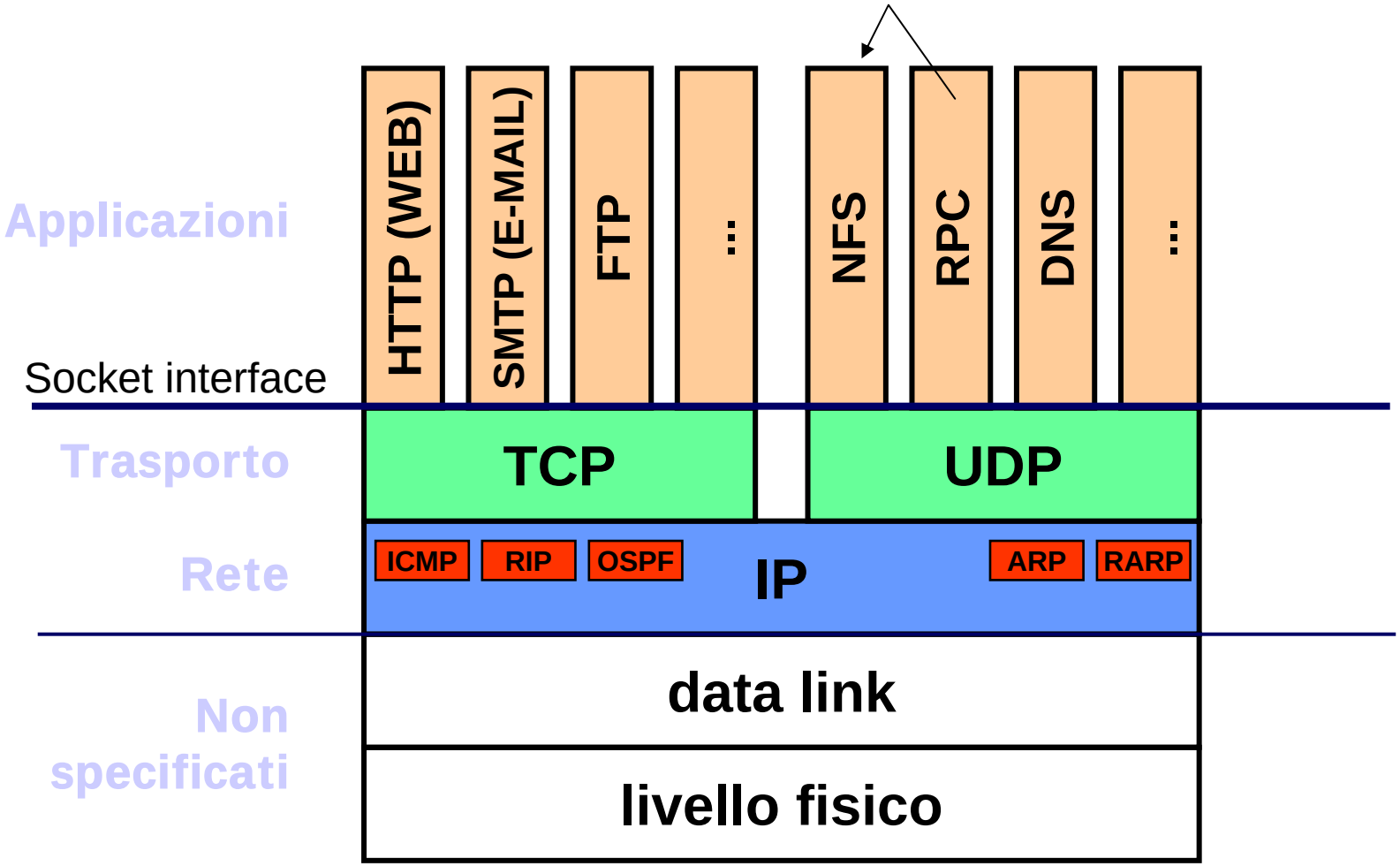
POLITECNICO DI MILANO





Protocolli applicativi

I protocolli applicativi



Paradigma client/server

Server: mette a disposizione servizi (Pagine Web, Posta elettronica,…)

- Uno dei processi che intendono comunicare (processo **server**) deve scegliere una porta e porsi in attesa (listen)

Client: utilizza le risorse offerte dal server

- L'altro processo (il **client**) blocca una porta libera arbitraria all'inizio della trasmissione

World Wide Web

Nato per organizzare l'enorme quantità di documentazione prodotta da migliaia di ricercatori

Specifiche e prima implementazione realizzate dal CERN di Ginevra (1990) per la condivisione di materiale eterogeneo

Web: sistema multimediale ad ipertesto (1992)

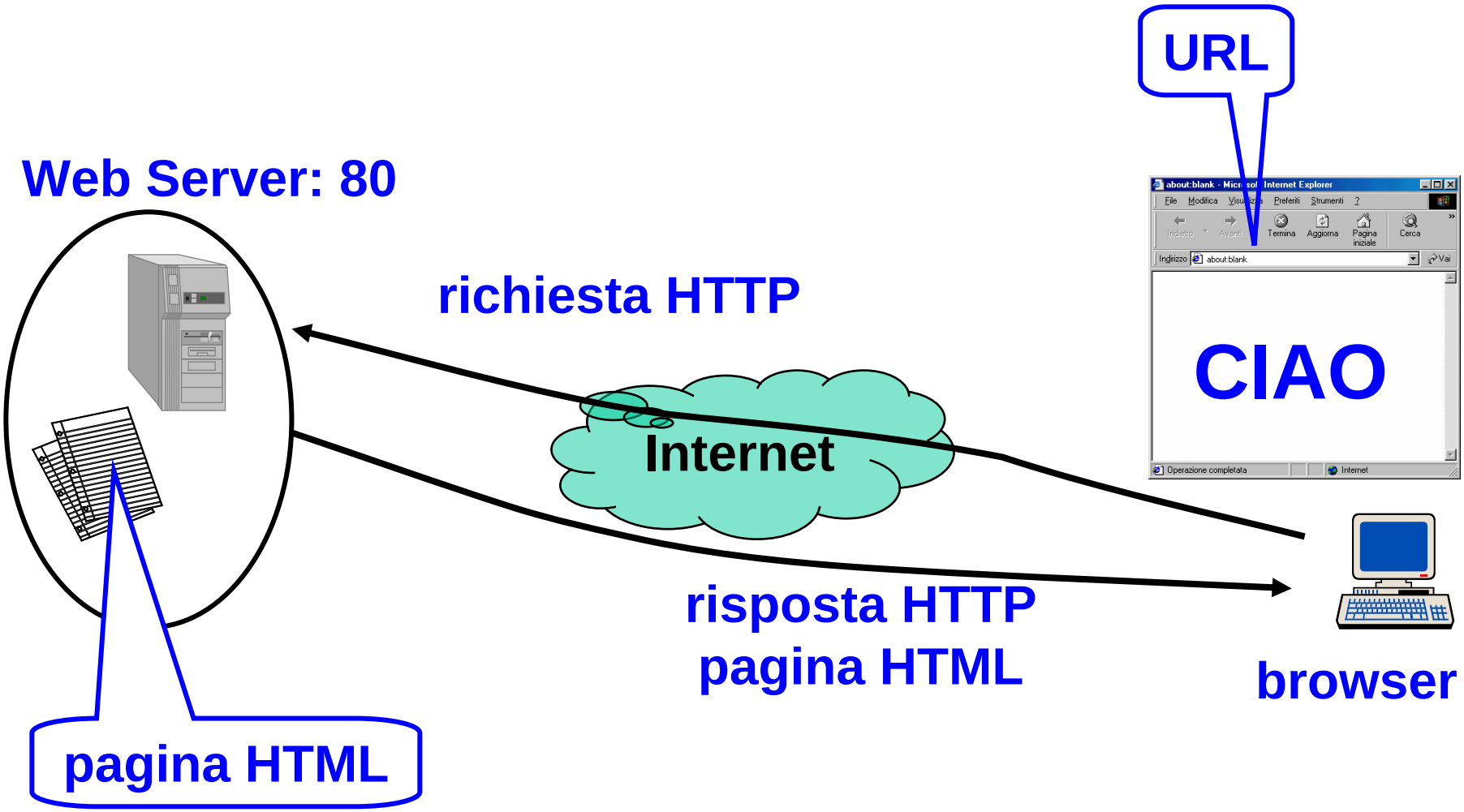
Oggi usato da aziende, enti e privati per vari scopi di comunicazione, ad es:

- Grid computing
- Web semantico

Terminologia:

- HTTP (HyperText Transfer Protocol):
  - Protocollo di livello applicativo per il Web
  - Usa il modello client/server
- URL (Uniform Resource Locator):
  - Identifica un oggetto nella rete e specifica il modo per accedere ad esso
- HTML (HyperText Markup Language)
  - Linguaggio di mark-up

Funzionamento del Web



URL

Schema

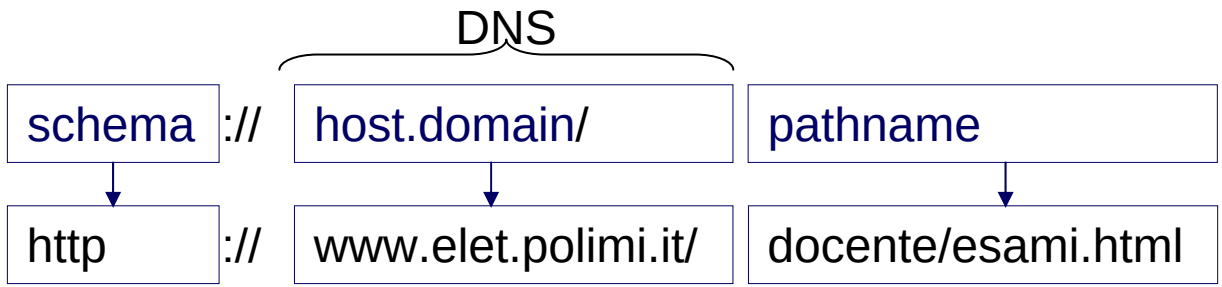
- indica il modo con cui accedere alla risorsa (protocollo da usare).

host.domain

- nodo nel quale risiede la risorsa Web.

pathname

- identifica la risorsa presso il server Web. In particolare, si specifica il cammino all'interno dell'organizzazione del file system dedicata alle risorse Web.



HTTP

È un protocollo di livello applicativo

Permette il reperimento delle risorse Web

Basato sul paradigma client/server

- I client e server Web devono supportare il protocollo HTTP per poter scambiare richieste e risposte (sono perciò anche chiamati client HTTP e server HTTP)

HTTP - Richiesta

Una richiesta HTTP comprende:

- un *metodo*
  - **GET**: richiede il documento specificato nel URL
  - **HEAD**: richiede solo l'informazione *header* relativa al documento
  - **POST**: richiede che il server accetti alcuni dati dal browser, come l'input delle form html
  - **PUT**: upload di dati in arrivo dal client
- un *URL*
- l'identificativo della *versione* del protocollo HTTP
- un insieme di *extension header* (opzionali):
  - Accept: i tipi di file che il browser può accettare
  - Authorization: usato se il browser vuole autenticarsi con il server; contiene informazioni come username e password
  - User-agent: il nome e la versione del browser
  - Host: nome DNS per consentire il *name-based virtual hosting*
- I *dati* della richiesta (opzionale):
  - In caso di POST o PUT il client invia i dati dopo gli *header*
  - In caso di GET o HEAD non ci sono dati da spedire

HTTP - Risposta

La risposta include:

- *Versione* del protocollo HTTP
- *Codice* di stato (3 cifre)
  - 200-299 successo
  - 300-399 redirezione
  - 400-499 errore sul lato client
  - 500-599 errore sul lato server
- *Reason phrase*
- *Header* della risposta
  - Server: nome e versione del server web
  - Date: la data corrente
  - Last-modified: la data di ultima modifica
  - Expires: la data di scadenza del documento
  - Content-length: dimensione in byte dei dati che seguono
- *Dati* della risposta:
  - Dopo gli header viene inserita una linea vuota; tutto ciò che segue sono i dati relativi alla risposta
  - Solitamente si tratta di un file HTML (ma non è imposto dal protocollo)

Se la pagina richiesta, oltre al testo HTML, contiene altri oggetti, ciascuno di essi sarà identificato da un URL differente, per cui è necessario che il browser invii un esplicito messaggio di richiesta per ognuno degli elementi collegati alla pagina.

Caratteristiche del protocollo HTTP

E' un protocollo *stateless*:

- il server non mantiene informazione sulle richieste precedenti del client
- rende difficili le *transazioni*

Versioni:

- HTTP/1.0 (*non persistente*)
  - Il server analizza la richiesta, risponde e chiude la connessione TCP
    - 2 RTT per ricevere ciascun oggetto
    - Ogni oggetto subisce lo "slow start" TCP
- HTTP/1.1 (*persistente*)
  - Sulla stessa connessione TCP: il server analizza una richiesta, risponde, analizza la richiesta successiva,..
  - Il client invia richieste per tutti gli oggetti appena riceve la pagina HTML iniziale.
    - Si hanno meno RTT e slow start
  - Problema: rimangono aperte molte connessioni in attesa che scatti il timeout

FTP

È un protocollo applicativo c/s che si appoggia su **TCP**

Il **F**ile **T**ranfer **P**rotocol viene utilizzato per il trasferimento di file tra macchine con architetture diverse

- I file vengono trattati come file di testo (7 bit per carattere) oppure come file binari (8 bit per carattere). Non viene modificato o tradotto il contenuto dei file.

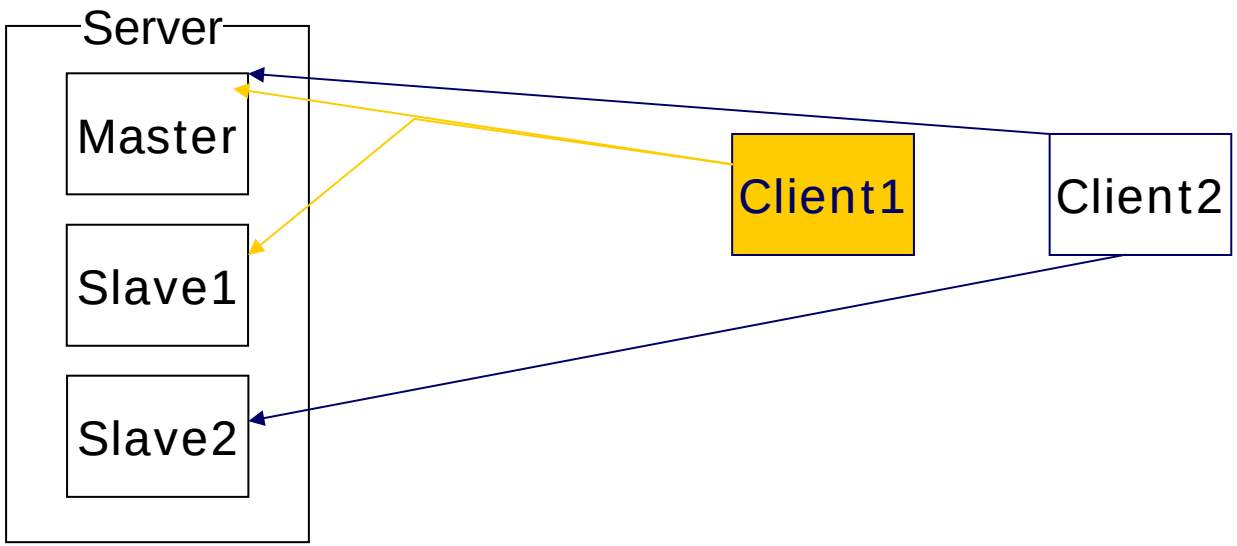
Deve fornire opportune funzioni per:

- Autorizzazioni
- Naming
- Gestire rappresentazioni eterogenee dei dati

Server FTP

La maggior parte degli FTP server permette l'accesso concorrente di client multipli:

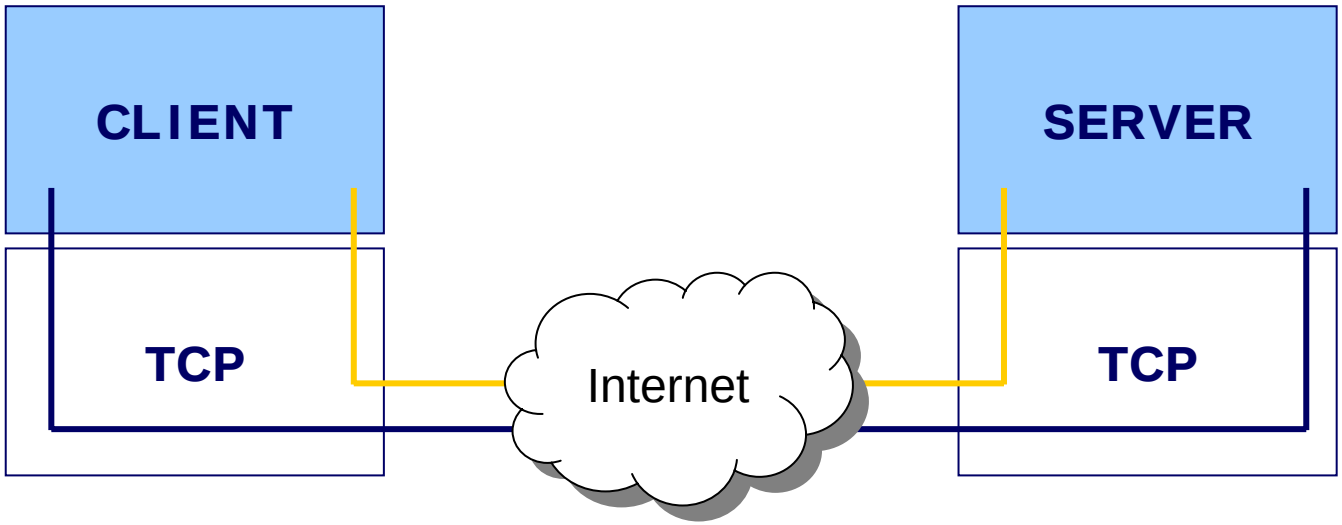
- un processo *master* si occupa di ricevere le connessioni e di creare uno *slave* per ognuna



Connessione di controllo e connessione dati

Il server (+ precisamente gli *slave)* gestiscono

- una connessione di *controllo*
- si avvalgono di un processo addizionale che crea un'ulteriore connessione utilizzata per il *trasferimento* dei dati.



Anonymous FTP

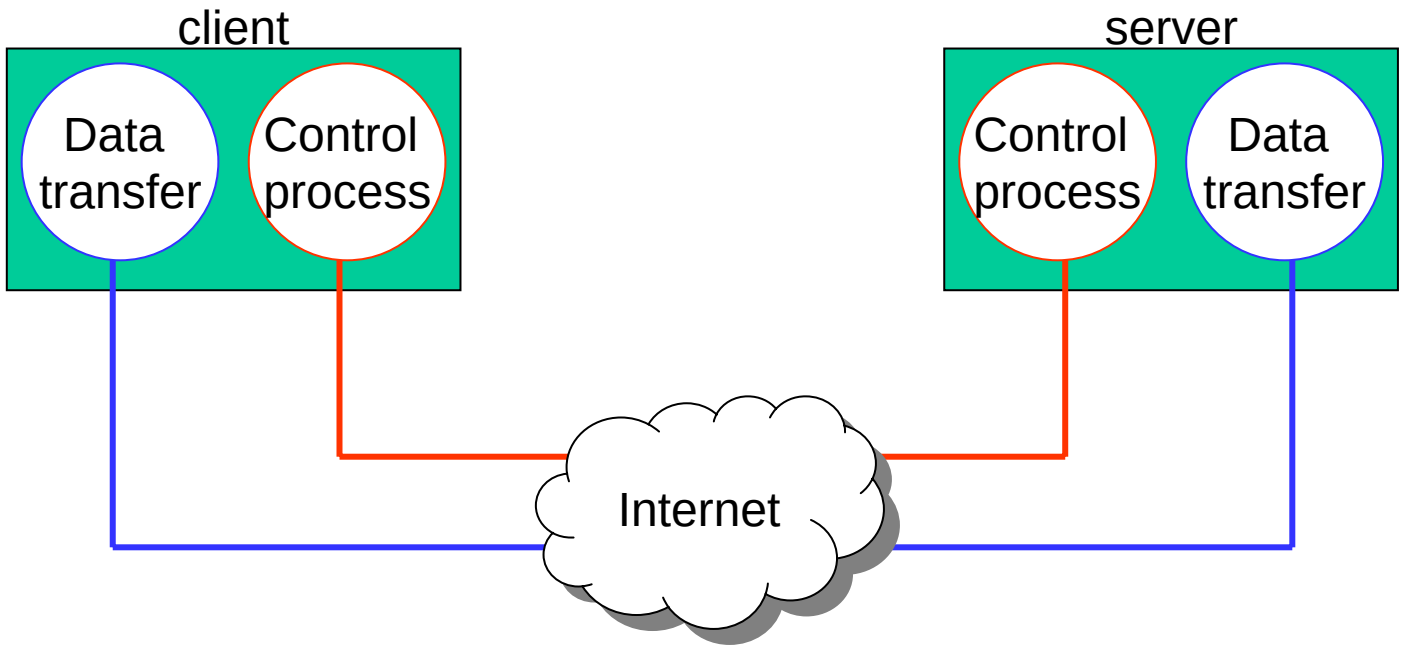
Anonymous FTP

- È il tipo più utilizzato
- Tutti possono accedere e scaricare i file memorizzati tramite un client FTP
- Viene usato principalmente per dare accesso *pubblico* a particolari directory di file

Non-anonymous FTP

- Richiede un accesso mediante *account* (username e password)

FTP



Modalità di connessione:

- Active FTP
- Passive FTP

NB: attivo/passivo è riferito alla connessione di trasferimento dati, lato server

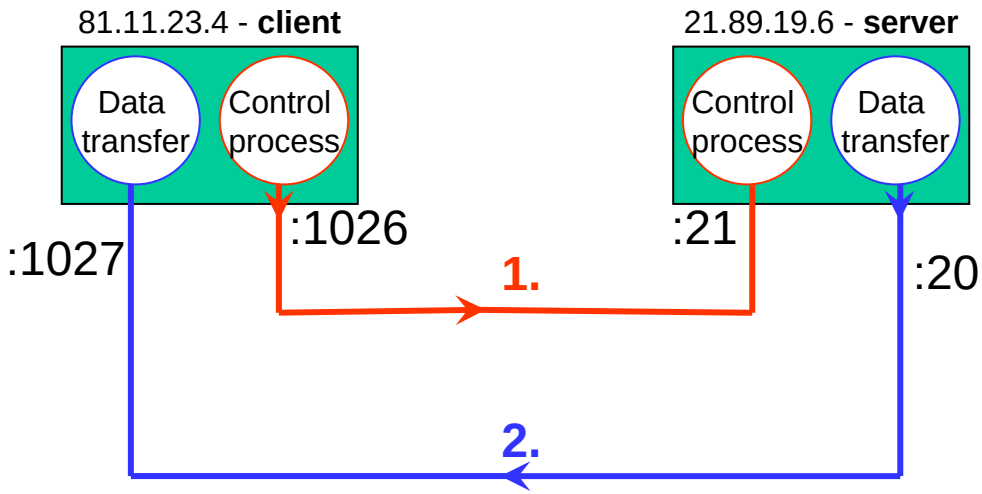
Active FTP

Il client si connette, da una sua generica porta N (che non sia già assegnata), alla porta 21 del server FTP

Il client si mette in attesa alla porta N+1 e comunica il nuovo stato al server

Il server si connette dalla porta 20 alla porta N+1 del client

Active FTP presenta problemi in presenza di Firewall sul lato client

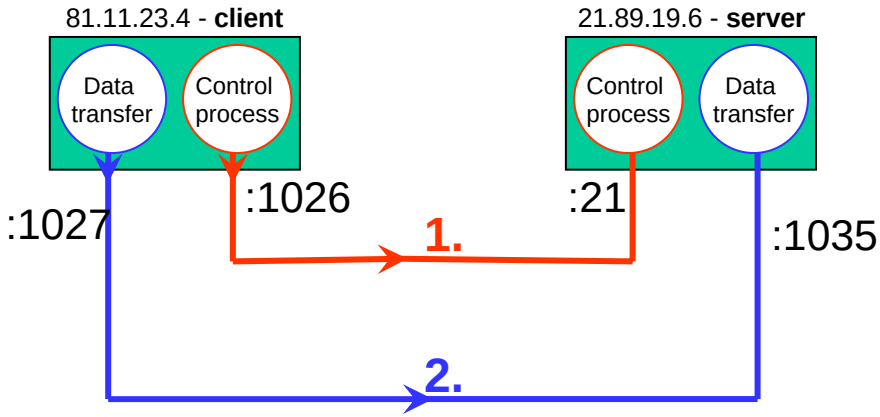


Passive FTP

E’ stato sviluppato per risolvere i problemi di connessione dal server al client

Il client inizializza entrambe le connessioni, sia quella di controllo che quella dati, scegliendo due porte a caso (N e N+1):

- da N instaura una connessione di controllo con la porta 21 del server FTP, comunicando la modalità *passive*
- il server si mette in attesa su una porta P, che comunica al client mediante la connessione di controllo
- il client apre la connessione per il trasferimento dei dati, dalla porta N+1, sulla porta P del server



Osservazioni su Passive FTP

Alcuni client (meno recenti) non supportano la modalità passive

La maggior parte dei web browser attuali possono essere usati anche come client ftp e generalmente supportano solamente passive FTP

Necessita l'apertura di porte oltre la 1024 sul lato server

FTP e NATTING

Problemi con NAT-BOX

- Il messaggio FTP contiene informazioni su indirizzi IP e porte TCP
- La NAT-BOX deve accedere al livello applicativo per poter cambiare IP e porte

Telnet

Telnet è un'applicazione *client/server* nata come *terminale virtuale remoto* basata su NVT:

- Servizio per il login remoto tramite Internet ("to telnet", "telneting")

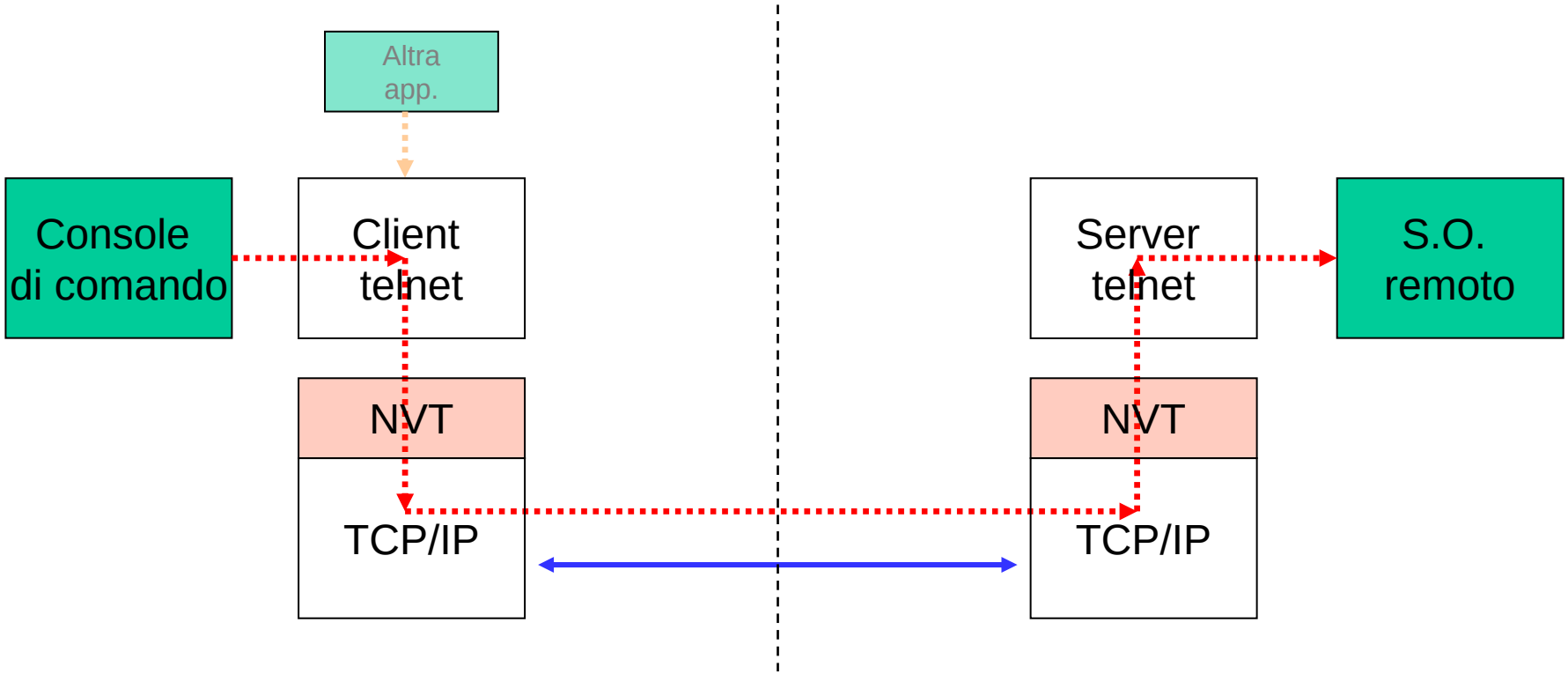
NVT (**N**etwork **V**irtual **T**erminal):

- Utilizza codici a 7 bit per i caratteri
- Fornisce un linguaggio standard per le comunicazioni
- "bi-directional character device" con keyboard/printer

Può ricevere input sia da *tastiera* che da una qualsiasi altra applicazione

- NB: telnet è stato sviluppato 1969, non c'erano problemi di sicurezza (veniva eseguito nella stessa organizzazione,..) → no encrypting, rischio di MITM (man in the middle)
- Oggi sempre più sostituito da ssh

Telnet


Altra app.
Console di comando
Client telnet
Server telnet
S.O. remoto
NVT
NVT
TCP/IP
TCP/IP

Telnet – es. FTP (active)

Client:USER anonymous

Server:331 Guest login ok, send your e-mail address as password.

Client:PASS NcFTP@

Server:230 Logged in anonymously.

Client:PORT 192,168,1,2,7,138

The client wants the server to send to port number 1930 on IP address 192.168.1.2.

Server:200 PORT command successful.

Client:LIST

Server:150 Opening ASCII mode data connection for /bin/ls.

The server now connects out from port 21 to port 1930 on 192.168.1.2.

Server:226 Listing completed

That succeeded, so the data is now sent over the established data connection

Client:QUIT

Server:221 Goodbye.

GET HTTP 1.0

GET / HTTP/1.0

HTTP/1.1 200 OK
Date: Mon, 21 Mar 2005 14:48:51 GMT
Server: Apache/2.0.53 (Win32)
Last-Modified: Wed, 16 Mar 2005 08:52:41 GMT
ETag: "7305-f6-44983720"
Accept-Ranges: bytes
Content-Length: 246
Connection: close
Content-Type: text/html

```
<HTML>
<HEAD><TITLE>MAIN PAGE</Title></HEAD>
<BODY>
  <center>My MAIN page</center> </br> </br>
  <a href="/secondpage.html">Link to SECOND page </a>   </br></br>
  <a href="/formexample.html">Link to sample Input Form </a>
</BODY>
</HTML>
```

GET HTTP 1.1

```
GET / HTTP/1.1
Host: server.com
Accept: text/html

HTTP/1.1 200 OK
Date: Mon, 21 Mar 2005 14:56:51 GMT
Server: Apache/2.0.53 (Win32)
Last-Modified: Wed, 16 Mar 2005 08:52:41 GMT
ETag: "7305-f6-44983720"
Accept-Ranges: bytes
Content-Length: 246
Content-Type: text/html

<HTML>
<HEAD><TITLE>MAIN PAGE</Title></HEAD>
<BODY>
  <center>My MAIN page</center> </br> </br>
  <a href="/secondpage.html">Link to SECOND
      page </a>   </br></br>
  <a href="/formexample.html">Link to sample Input
      Form </a>
</BODY>
</HTML>
```

```
GET /secondpage.html HTTP/1.1
Host: server.com
Accept: text/html

HTTP/1.1 200 OK
Date: Mon, 21 Mar 2005 14:57:15 GMT
Server: Apache/2.0.53 (Win32)
Last-Modified: Tue, 15 Mar 2005 08:08:18 GMT
ETag: "733a-57-88020bbf"
Accept-Ranges: bytes
Content-Length: 87
Content-Type: text/html

<HTML>
<HEAD><TITLE>Second Page</Title></HEAD>
<BODY>My SECOND page</BODY>
</HTML>
```