

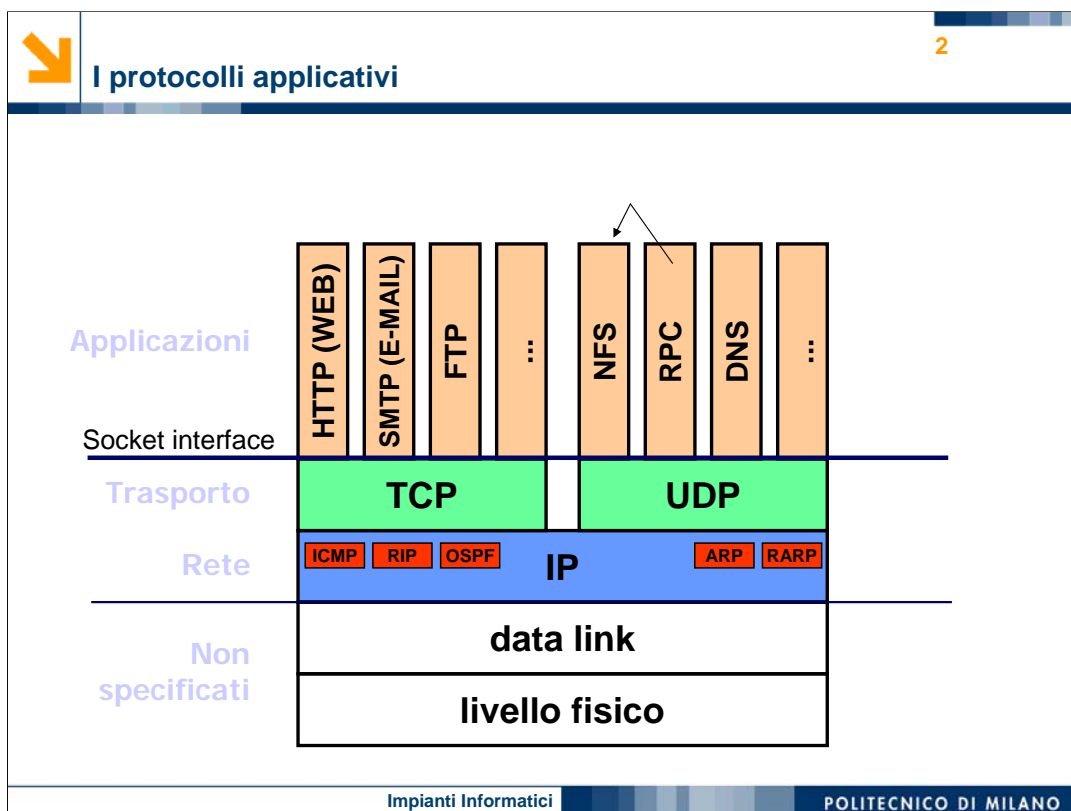
Protocolli per il Web

Impianti Informatici

 POLITECNICO DI MILANO



Protocolli applicativi



I protocolli applicativi rappresentano il più alto livello di astrazione nel modello ISO-OSI.

- La suite TCP/IP utilizzata in Internet definisce i protocolli fino al livello di trasporto. Ci sono quindi una serie di applicazioni che si appoggiano ad esso. Alcune di esse utilizzano protocolli proprietari e
- comunicano con il livello TCP o UDP mediante la socket interface. Altre applicazioni si avvalgono di protocolli standard, quali possono essere,
- L'http per il web
- L'SMTP per le email
- L'FTP per il trasferimento di file
- Il DNS per la risoluzione dei nomi di domini
- E numerosi altri protocolli



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

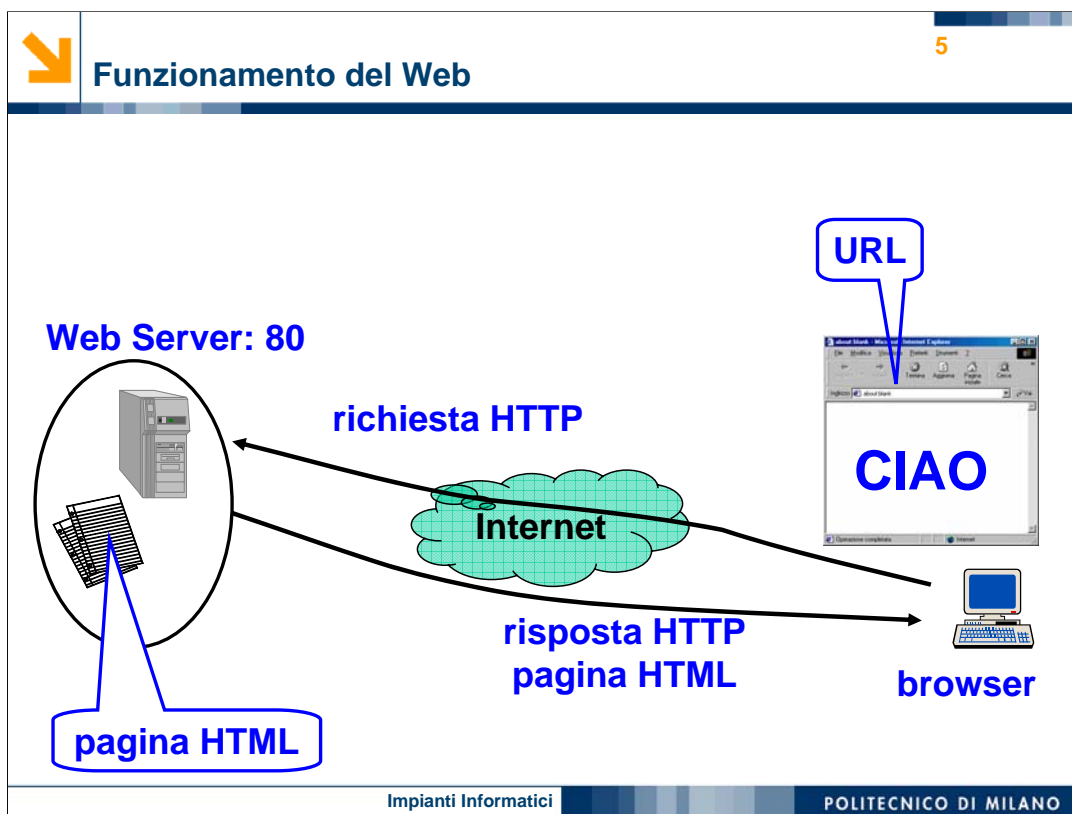
Una tipologia ben nota ed utilizzata di tali applicazioni è quella basata sul modello client/server. In esso ci sono due entità logiche fondamentali, che sono appunto il client ed il server, con diverse funzionalità. La divisione è logica e non necessariamente fisica: i due moduli possono risiedere sullo stessa macchina.

- La funzione logica del server è quella di offrire dei servizi. In termini pratici il server
- Si pone in ascolto su una determinata porta, in attesa un altro processo inizi la comunicazione
- Il client è il modulo che utilizza le risorse del server. È il componente che
- Apre attivamente la connessione, identificata come di consueto da una coppia di socket.

	<div style="text-align: right;">4</div> <h2 style="margin: 0;">World Wide Web</h2>
<p>Nato per organizzare l'enorme quantità di documentazione prodotta da migliaia di ricercatori</p> <p>Specifiche e prima implementazione realizzate dal CERN di Ginevra (1990) per la condivisione di materiale eterogeneo</p> <p>Web: sistema multimediale ad ipertesto</p> <p>Oggi usato da aziende, enti e privati per vari scopi di comunicazione</p> <p>Terminologia:</p> <ul style="list-style-type: none"> ▪ HTTP (HyperText Transfer Protocol): <ul style="list-style-type: none"> • Protocollo di livello applicativo per il Web • Usa il modello client/server ▪ URL (Uniform Resource Locator): <ul style="list-style-type: none"> • Identifica un oggetto nella rete e specifica il modo per accedere ad esso ▪ HTML (HyperText Markup Language) <ul style="list-style-type: none"> • Linguaggio di mark-up 	
<div style="display: flex; justify-content: space-between;"> Impianti Informatici POLITECNICO DI MILANO </div>	


Molte applicazioni per il Web sono basate sul paradigma client/server.

- Il Web è nato con l'obiettivo di organizzare e gestire il materiale sempre + crescente prodotto dai migliaia di ricercatori
- La sua origine deriva da un progetto degli anni 60 dell'ARPANET, ridenominata in seguito ARPA, un'agenzia del dipartimento della difesa statunitense, il cui scopo era di collegare fra loro sistemi remoti. Il termine Internet nasce nel 1982, come contrazione di Internetworking.
- Nel 1992 è stato quindi definito il web, uno dei programmi più usati su Internet, definito come sistema multimediale ad ipertesto, con tecnologia client/server, per la diffusione di informazioni, per lo più di carattere testuale.
- Il Web è oggi il più potente strumento di comunicazione, utilizzato tanto da privati quanto da aziende
- Le specifiche del Cern precisano l'uso
- dell'HTTP come protocollo di livello applicativo. L'identificazione di una risorsa nella rete
- avviene tramite l'URL. Contemporaneamente al web è stato anche specificato il linguaggio da utilizzare per la scrittura delle pagine.
- Tale linguaggio è l'HTML



Il web è una tipica applicazione client/server: il classico scenario vede,

- dal lato client, la presenza di un browser
- E dal lato server un server web, in ascolto sulla porta 80. L'utente identifica la risorsa,
- che può essere una pagina HTML, mediante
- L'URL. Il browser interpreta l'URL e invia una richiesta al server web
- Utilizzando il protocollo appropriato: l'HTTP. Il server preleva la pagina e
- la invia tramite lo stesso protocollo HTTP al browser
- Dove viene interpretata e presentata sullo schermo



URL

6

Schema

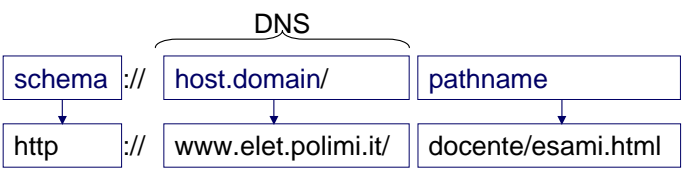
- indica il modo con cui accedere alla risorsa (protocollo da usare).

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.



Impianti Informatici

POLITECNICO DI MILANO

L'URL permette di identificare ed accedere ad una risorsa sulla rete.

1. In sostanza è una stringa, composta da diverse parti:
2. Lo schema indica la modalità di accesso alla risorsa, cioè quale protocollo bisogna usare per interagire con il server che controlla la risorsa. Il metodo di accesso più comune è HTTP, ovvero il protocollo nativo per il recupero di risorse Web.
3. L'host ed il dominio individuano il nodo in cui risiede la risorsa. Il nome viene solitamente risolto in un indirizzo IP da un server DNS.
4. Il path name individua la risorsa all'interno della macchina, completo di percorso nel file system del sistema.



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)

L'HTTP è

- Il protocollo applicativo che
- consente il reperimento di risorse su Internet
- Il suo funzionamento è basato sul paradigma client/server. Affinché la comunicazione tra client e server, tipicamente un browser ed un server web, possa avvenire è necessario
- che entrambi i moduli supportino il protocollo HTTP



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**: sostituisce il contenuto di un documento del server con 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: l'indirizzo IP e la porta
- 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

Il protocollo HTTP è basato su messaggi di richiesta e di risposta.

- Una richiesta HTTP deve necessariamente comprendere
 - Il metodo, ovvero il tipo di richiesta. Può essere una
 - GET di richiesta di una determinata risorsa, oppure
 - HEAD per richiedere unicamente l'header,
 - POST per inviare dei dati, o ancora
 - il metodo PUT
 - Occorre specificare l'URL della risorsa richiesta,
 - e la versione del protocollo HTTP. È inoltre possibile specificare
 - uno o più header che contengono informazioni aggiuntive, come ad esempio
 - i tipi di dato che il browser è in grado di visualizzare
 - Dati di autenticazione
 - il tipo di software utilizzato dal client oppure
 - Dati relativi all'host richiedente.
- Successivamente a questi campi si inseriscono
- gli eventuali dati della richiesta. Essi sono necessari
 - unicamente nei casi dei metodo POST o PUT



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.

A seguito di una richiesta, vi è un

1. messaggio di risposta, che include
2. La versione del protocollo HTTP
3. Un codice rappresentante lo stato dell'operazione richiesta; è un numero di 3 cifre che può indicare
4. Successo
5. Oppure redirezione della richiesta, ad esempio se la pagina non è fisicamente presente nel server. È un messaggio di avvertimento per il client in termini di sicurezza.
6. Il codice di stato può denotare un errore dal lato client, ad esempio in caso di sintassi errata
7. Dal lato server, ad esempio nel caso in cui non sia presente la risorsa indicata nell'URL. Assieme al codice di stato viene anche
8. Inviata la reason phrase, una breve descrizione a parole dell'esito della richiesta.
9. Altri header riscontrabili in una risposta possono essere:
10. Il tipo del server web
11. La data
12. La data dell'ultima modifica delle risorse inviate
13. La scadenza del documento
14. E la sua dimensione.
15. A seguito degli header, dopo una linea vuota, si trovano i dati della risposta.
16. Ad esempio una pagina HTML. Se tale pagina contiene anche altri oggetti
17. È necessario che il browser invii una richiesta per ciascuno di essi specificando il relativo URL




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

1. L'HTTP è un protocollo stateless
2. Cioè il server non memorizza le informazioni riguardanti le precedenti richieste effettuate dallo stesso client.
3. Questo complica la gestione delle transazioni, come ad esempio nel caso di un sistema di acquisto online.
4. Le due versioni esistenti di HTTP sono la 1.0 e la 1.1
5. La prima non è persistente, quindi per ogni richiesta
6. Occorre aprire una connessione TCP col server, il quale analizza la richiesta, risponde e poi chiude la connessione, penalizzando in tal modo le prestazioni in caso di pagine composte da molti oggetti.
7. La versione 1.1 è invece persistente, quindi
8. All'interno di una stessa connessione TCP è possibile richiedere più risorse.
9. Nel caso di una pagina HTML si ha perciò un miglioramento delle prestazioni.
10. Lo svantaggio è che il server chiude le connessioni dopo un tempo prefissato, col rischio di mantenerne aperte molteplici in attesa che scatti il timeout.

 **FTP**

11

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

Il **File Transfer Protocol** 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

Impianti Informatici

POLITECNICO DI MILANO

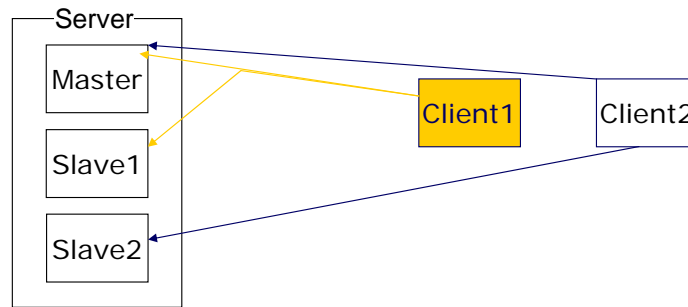
Uno dei protocolli applicativi più diffusi che

1. si appoggia sullo stack TCP/IP è l'FTP.
2. Esso viene utilizzato per il trasferimento dei file tra sistemi eterogenei
3. Il file viene inviato senza essere né modificato né tradotto: esistono due modalità di trasferimento, una con codifica a 7 bit come file di testo, ed una a 8 bit come file binari.
4. Tra le funzionalità richieste vi è la gestione dei
5. Diritti di accesso ai file,
6. Di problematiche di naming e di
7. Gestione di rappresentazioni eterogenee di dati.



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

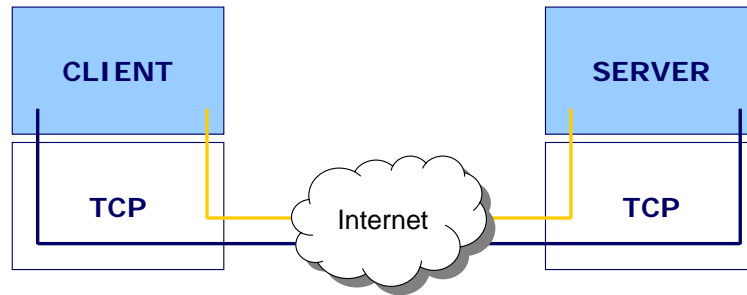


1. La maggior parte delle implementazioni di server FTP, consente l'accesso concorrente a client multipli tramite
2. un processo master in ascolto su una determinata porta che si occupa di ricevere le connessioni in ingresso rimanendo in ascolto su una determinata porta.
3. Per ogni connessione crea un processo slave, e
4. redirige la connessione del client su tale processo,
5. così da poter accogliere nuove richieste



Il server, ed in particolare gli *slave* gestiscono

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



1. I server, ed in particolare gli *slave*, gestiscono due connessioni TCP separate
2. Una per il controllo e la configurazione del trasferimento, su una determinata porta
3. Una per il trasferimento vero e proprio dei dati su una porta differente



Anonymous FTP

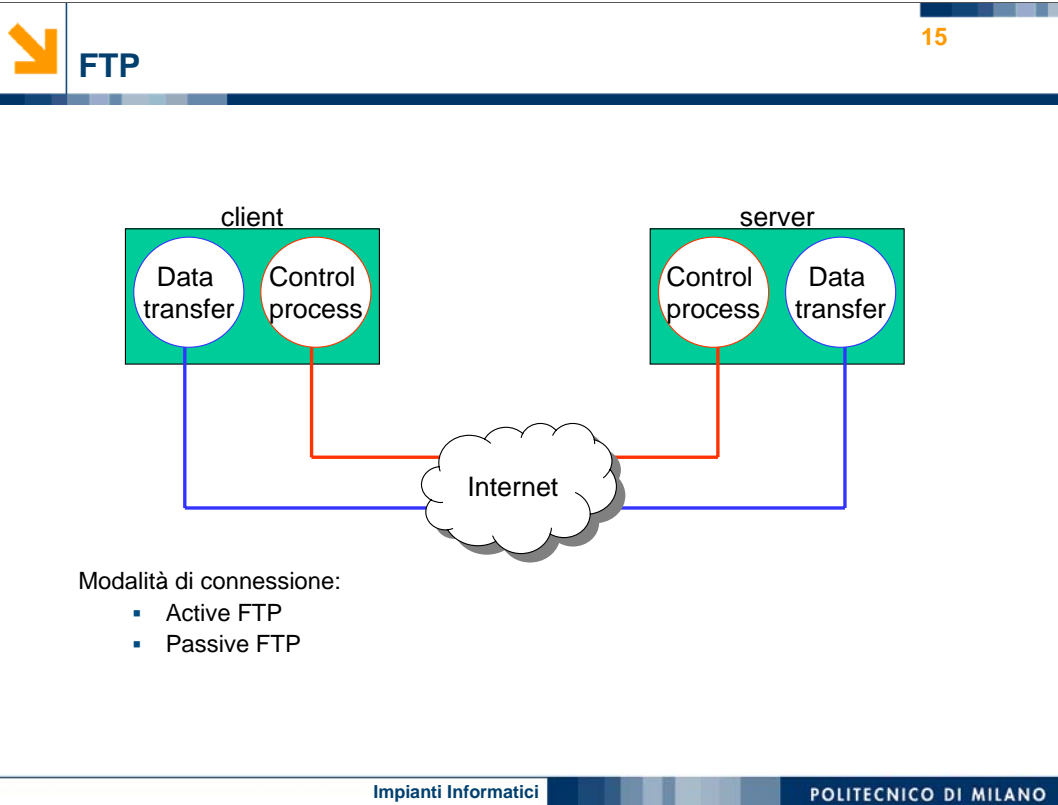
- È il tipo più utilizzato
- Tutti possono accedere e scaricare i file memorizzati tramite un client FTP
- Viene usato principalmente per dare accesso *pubblico* a particolari directory di file

Non-anonymous FTP

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

A seconda che sia necessario un'autorizzazione o meno all'accesso di una risorsa si parla di

1. FTP anonimo oppure non-anonimo:
2. Il primo è quello più diffuso, in cui
3. Ognuno può accedere ai dati, tipicamente pubblici, del server
4. L'alternativa è l'FTP non anonimo
5. Che richiede l'accesso mediante username e password, per proteggere risorse dal contenuto privato



Esistono due modalità di connessione tra client e server,

1. l'FTP attivo e l'FTP passivo, a seconda che sia il server o il client ad aprire la connessione riservata al trasferimento

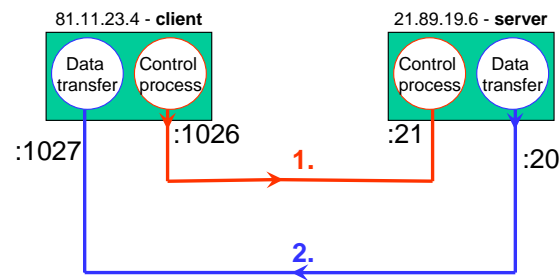


Il client si connette, da una sua generica porta N (che non sia già assegnata), alla porta 21 del server FTP

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



In modalità attiva,

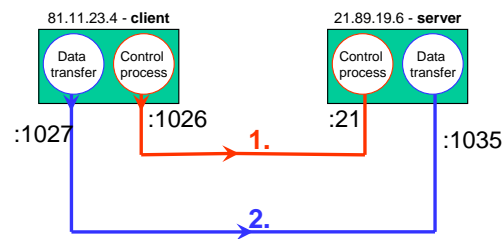
1. il server si trova in ascolto sulla porta 21
2. Il client apre attivamente una connessione da una sua generica porta, poniamo la 1026, alla porta 21 del server
3. Si mette quindi in attesa su una sua porta libera, supponiamo la 1027, comunicando al server tale porta
4. Questi apre attivamente la connessione dalla sua porta 20,
5. Alla porta 1027 del client
6. La presenza dei firewall sul lato client porta alla necessità di una soluzione alternativa, dato che con questa modalità non sarebbe possibile l'apertura della connessione di trasferimento dati.



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



1. La soluzione ai problemi di firewall lato client è l'FTP modalità passiva
2. È il client ad inizializzare entrambe le connessioni
3. Da una sua porta libera apre la connessione di controllo sulla porta 21 del server FTP, comunicando che sta operando in modalità passiva
4. Il server comunica al client la porta su cui si è messo in attesa
5. Il client apre attivamente la connessione per il trasferimento dei dati, da una sua porta libera alla porta comunicatagli dal server



Alcuni client non supportano la modalità passiva

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

1. Non tutti i client supportano la modalità passiva, soprattutto quelli più obsoleti, dato che il protocollo prevedeva originariamente la sola modalità attiva
2. Tuttavia la maggiorparte dei browser oggi in commercio possono essere usati come client ftp e la modalità supportata è quasi sempre quella passiva
3. Uno dei problemi più evidenti del passive ftp è la necessità di aprire porte oltre la 1024 sul lato server, con relativa complicazione nella gestione eventuali firewall



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

Il protocollo FTP, così come altri, necessita di un particolare trattamento

1. nel caso si stiano utilizzando funzioni di NATTING, occorre considerare il fatto
2. Che i messaggi scambiati dalla connessione di controllo FTP contengono a livello applicativo informazioni su indirizzi ip e porte tcp.
3. Questo significa che la NAT box deve implementare anche il protocollo FTP, un protocollo applicativo, in modo da poter traslare correttamente le informazioni



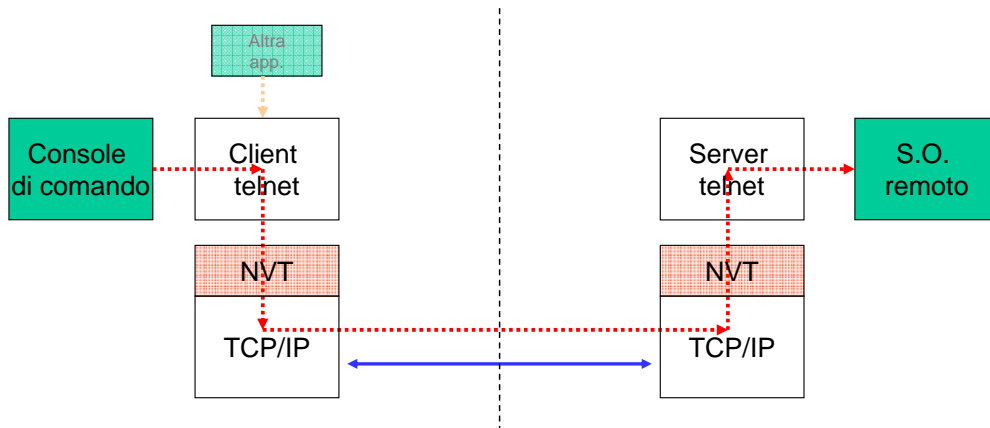
Telnet è un'applicazione *client/server* nata come *terminale virtuale remoto* basata su NVT:

- Servizio per il login remoto tramite Internet

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

- Utilizza codici a 7 bit per i caratteri
- Fornisce un linguaggio standard per le comunicazioni

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





Telnet – es. FTP (active)

22

Client:USER anonymous

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

Client:PASS NcFTP@

Server:230 Logged in anonymously.

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

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

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

23

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
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>
```