Differenze tra comunicazione attiva e passiva FTP

FTP in modalità attiva lavora in questo modo:

- Il client inizia una connessione (generalmente sulla porta 21) verso il server e comunica la porta in cui si pone in ascolto (generalmente una porta random decisa dal client) per lo scambio dei dati;
- Il server conferma la connessione (nel caso validando le credenziali)
- Il server apre una connessione verso il client sulla porta comunicata
- Il client conferma la connessione
- Inizia la trasmissione dei dati

FTP in modalità passiva lavora in questo modo:

- Il client inizia una connessione (generalmente sulla porta 21) verso il server;
- Il server conferma la connessione (nel caso validando le credenziali) e comunica la porta in cui si pone in ascolto (generalmente una porta random in un preciso range impostato lato server) per lo scambio dei dati;
- Il client apre una connessione verso il server sulla porta comunicata x la trasmissione dei dati
- Il server conferma la connessione
- Inizia il trasferimento dei dati

Esempio di una connessione attiva

```
<><> A connessione aperta e autenticazione eseguita <><>
---> PORT 192,168,150,80,14,178 *
200 PORT command successful.
---> LIST
150 Opening ASCII mode data connection for file list.
drwx------   3 slacker    users         104 Jul 27 01:45 public_html
226 Transfer complete.
```

Esempio di una comunicazione passiva

```
<><> A connessione aperta e autenticazione eseguita <><>
---> PASV
227 Entering Passive Mode (192,168,150,90,195,149)*.
---> LIST
150 Opening ASCII mode data connection for file list
drwx------   3 slacker    users         104 Jul 27 01:45 public_html
226 Transfer complete.
```

*Indicazione dell'ip e della relativa porta aperta sul server: 192.168.150.90:(195 * 256 + 149 $_{\text{da fare la somma}}$)

Interattività FTP

Inizialmente FTP era una interfaccia a riga di comando, dove l'utente specificava il server remoto, effettuava l'accesso, operava una serie di richieste e chiudeva la connessione.
Attualmente, quasi tutto l'utilizzo di FTP avviene tramite browser. L'utente inserisce l'URL o clicca un link, il browser usa FTP per contattare il server remoto e ottenere la lista di file, e dunque l'utente seleziona il file per il download

Telnet Options

Vedi appendice appunti parte 2

File di zona e resource record

Il file di zona è il file di testo in cui sono scritte le informazioni inerenti la raggiungibilità dei server e servizi associati ad un dominio e informa su quali siano i server che hanno il compito di mantenere queste informazioni aggiornate. Esso ha il compito di informare Internet su dove si trovino i servizi associati ad un dominio affinchè siano raggiungibili.
Il file di zona corrisponde ad un database contenuto all'interno di un name server, ed è composto da uno o più resource record.
Quando un resolver sottopone un nome di dominio ad un name server, saranno proprio i resource record associati a tale nome ad essere tornati.
Un resource record è una quintupla composta da
Domain_name: indica il dominio a cui si riferisce il record
Time_to_Live: indicatore della stabilità del record, il cui valore sarà più alto quando l'informazione è stabile, mentre sarà più basso per le info temporanee (importante per l'uso della cache)
Class: codice che identifica l'area di interesse del record (usualmente "IN", per "internet")
Type: Tipo di record (SOA, A, CNAME, NS…)
Value: Valore associato al record (può essere un numero, un nome di dominio o una stringa ASCII.
Grazie alle informazioni nei resource record sarà possibile risolvere il nome di dominio (che potrebbe risolversi in un IP, in una indicazione a chiedere ad un altro name server, all'indirizzamento verso un altro nome di dominio). Esso contiene anche un record (tipo SOA) che contiene tutte le informazioni in merito all'origine dell'autorità, il seriale, flag e timeout utilizzati per mantenere coerenza tra server primari e secondari.

Comunicazione POP3

Vedi appendice 3 appunti parte 2

Paradigma Client Server

Il paradigma client/server permette di centralizzare risorse e funzionalità su un certo nodo fisico (o su un insieme di nodi) in modo del tutto indipendente dagli accessi alle medesime risorse e funzionalità, che possono avvenire da un numero qualunque di altri nodi periferici.
Ciò prevede uno scenario dove vi siano dispositivi in grado richiedere servizi, e che vi sia un dispositivo in grado di fornire servizi.
Sarà quindi il client a rivestire il ruolo attivo (invierà le richieste), mentre il server rivestirà un ruolo passivo (accoglie le richieste, fornisce risposte). Per svolgere questo ruolo, il server dovrà rimanere in ascolto

delle possibili richieste pervenibili dai client, e prontamente operare in modo da fornire una risposta qualora si presenti una richiesta.
Esistono diversi protocolli di uso comune che implementano comunicazioni basate su questo paradigma, ma di sicuro il più utilizzato è HTTP, alla base dell'attuale WWW, dove un client, tramite delle apposite HTTP request richiederà i servizi ad un server, ed esso provvederà a rispondere tramite una HTTP response, contenente un codice di stato (ad indicare il risultato della richiesta) e dati a supporto.

Tipologia Record DNS

Pagina 586 del libro

Fetch and store

Paradigma che prevede l'utilizzo di solo due comandi (fetch e store) che prevedono rispettivamente di leggere il valore di un dato o di memorizzare un valore in un dato.
Questo paradigma prevede dunque che ogni oggetto abbia un nome unico ed un valore che può essere ottenuto o memorizzato tramite i due relativi comandi.
SNMP implementa questo paradigma (in quanto non definisce un insieme di comandi da utilizzare per svolgere le sue funzioni, ma esegue le sue operazioni tramite lettura e setting di valori).
Nel concreto, SNMP applica il paradigma a più di soli due comandi (get-request, get-next-request, set-request…)

Differenze tra socket TCP e UDP

- Per TCP è necessario definire come tipo di socket SOCK_STREAM, mentre per UDP si dovrà impostare SOCK_DGRAM (secondo parametro della funzione socket())
- A lato server, per TCP è necessario effettuare il binding (associazione ad un socket di una porta), mentre per UDP sarà necessario solo qualora ci sia necessità di ricevere. Se si deve solo spedire messaggi, non serve.
- A lato server, per TCP è necessario utilizzare la primitiva listen per modificare lo stato del socket in LISTEN e indicare al kernel in numero massimo di connessioni simultaneo accodabili per quel socket. UDP non prevede ciò.
- A lato server, per TCP a lato server è necessario utilizzare la primitiva accept per mettere il server in attesa di connessioni TCP in entrata. UDP non prevede ciò.
- A lato client, per TCP sarà necessario utilizzare la primitiva connect per effettuare la connessione al socket TCP del server. Per UDP è possibile utilizzare questa primitiva, ma la semantica cambia (si limiterà a memorizzare localmente le informazioni del destinatario per l'invio dei messaggi)
- Per l'invio di messaggi TCP utilizza la primitiva send, mentre UDP utilizza la primitiva sendto
- Per la ricezione di messaggi TCP utilizza la privita recv, mentre UDP utilizza la primitiva recvfrom
- TCP garantisce la ricezione dei messaggi in ordine, UDP non garantisce la ricezione né tantomeno la ricezione dei messaggi

Differenza tra DNS iterativo e ricorsivo

Nella risoluzione iterativa, il client invia una query al Local Name Server, esso verifica se il nome può essere convertito rispondendo al client con l'indirizzo IP corrispondente, altrimenti si limita a comunicargli il nome del server che secondo lui è in grado di risolvere il nome. Successivamente, il client ripete la procedura con il server DNS fornitogli. Di conseguenza, quando un client adotta questo tipo di risoluzione obbliga il server DNS a fornire una risposta, che sia anche non autorevole (come il name server a cui rivolgersi).

Nella Risoluzione DNS di tipo Ricorsiva, il client aspetta dal server DNS contattato la risposta alla sua richiesta. Il server DNS se è responsabile del dominio, risolve l'indirizzo altrimenti trasmette la richiesta ad un server DNS di livello superiore e aspetta la risposta per il client. Quindi il server DNS ricorsivo si impegna a

rispondere sempre e comunque con la risoluzione del nome, appellandosi lui stesso ad altri DNS, eseguendo però delle query iterative. Dunque, il resolver avrà sempre una risposta autorevole alla sua interrogazione.

Caching HTTP

I browser memorizzano le pagine nella cache per risparmiare banda.
Essi mantengono una memorizzazione temporanea (cache) per le pagine recenti, e quando è richiesta una pagina, il browser controlla per vedere se è già nella cache.

- Se non c'è, genera la richiesta GET. Quando arriva il messaggio di risposta, il browser visualizza la pagina e la memorizza nella cache (assieme alle informazioni nell'header).
- Se la pagina è presente in cache, il browser invia la GET con l'header If-Modified-Since relativo ai dati della pagina. Se il codice tornato è 200, la pagina è stata aggiornata, il browser visualizza la pagina e la salva in cache, altrimenti il codice tornato sarà 304 e verrà visualizzata la versione memorizzata in cache.

Direttive Cache-Control
Cache-Control: must-revalidate → Una volta che la risorsa diventa vecchia, la cache non dovrà usare la sua copia senza aver validato la sua copia con successo sul server di origine
Cache-Control: no-cache → Il browser può salvare il cache la response, ma questa dovrà sempre passare attraverso una validazione dal server di origine
Cache-Control: immutable → La risorsa non cambierà nel tempo
Cache-Control: no-store → la cache non deve memorizzare nulla sulla richiesta del client o sulla risposta del server
Cache-Control: no-transform → non si devono verificare trasformazioni o conversioni per la risposta (Content-Encoding, Content-Range e Content-Type non devono essere modificati da un proxy).
Cache-Control: public → La risorsa può essere salvata da qualsiasi cache
Cache-Control: private → La risorsa può essere salvata solo nella cache del browser
Cache-Control: proxy-revalidate → Funziona come must-revalidate ma solo per le cache condivise (proxy). Ignorato dalle cache private

Esempi HTTP

Richiesta e risposta a GET condizionale

```
GET /~avf/ HTTP/1.1
Host: www.myserver.unimi.it
If-Modified-Since: Tue, 30 Aug 2005 14:00:00 GMT
-------------------------------------------------------------------------------
HTTP/1.1 304 Not Modified
Date: Tue, 06 Jan 2006 14:08:58 GMT
Server: HP Apache-based Web Server/1.3.27 (Unix) mod_perl/1.27 PHP/4.2.2
ETag: "10b3e-1000-431452ef"

Connection closed by foreign host
```

Richiesta e risposta HEAD

```
HEAD /~avf/index.html HTTP/1.1
Host: www.myserver.unimi.it
-------------------------------------------------------------------------------
HTTP/1.1 200 OK
Date: Tue, 06 Jan 2006 14:23:24 GMT
Server: HP Apache-based Web Server/1.3.27 (Unix) mod_perl/1.27 PHP/4.2.2
Last-Modified: Tue, 30 Aug 2005 12:37:03 GMT
ETag: "10b3e-1000-431452ef"
Accept-Ranges: bytes
Content-Length: 4096
Content-Type: text/html

Connection closed by foreign host.
```

Cookies

Quando il server vuole salvare lo stato di una sessione utilizza i cookie. Nella response invia

```
Set-cookie: CUSTOMER=Dave_Reed; PATH=/; EXPIRES=Thursday, 29-Jan-04 12:00:00
```

Quando l'utente ritorna all'URL, il browser manda al server i dati del cookie come parte della sua richiesta

```
Cookie : CUSTOMER=Dave_Reed
```

404 File non trovato

```
HTTP/1.1 404 Not Found
Date: Tue, 06 Jan 2006 13:40:52 GMT
Server: HP Apache-based Web Server/1.3.27 (Unix) mod_perl/1.27 PHP/4.2.2
Transfer-Encoding: chunked
Content-Type: text/html

a5
<html>
<head>
<title>404 Not Found</title>
<link rel="stylesheet" href="./fonts.css" type="text/css">
</head>

<body bgcolor="#FFFFFF" text="#000000">

  <h1>Page Not Found</h1>
        <p class="text">The requested URL
        /~avf/foo.html
        was not found on this server.</P>
</body>
</html>

0

Connection closed by foreign host.
```

Richiesta e risposta dove è specificata una directory (e viene cercato il file denominato index.html)

```
GET /~avf/ HTTP/1.1
Host: www.myserver.unimi.it
-------------------------------------------------------------------------
HTTP/1.1 200 OK
Date: Tue, 06 Jan 2006 11:45:09 GMT
Server: HP Apache-based Web Server/1.3.27 (Unix) mod_perl/1.27 PHP/4.2.2
Last-Modified: Tue, 30 Dec 2005 12:37:03 GMT
ETag: "10b3e-1000-431452ef"
Accept-Ranges: bytes
Content-Length: 4096
Content-Type: text/html

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">>
<head>
…
</head>
<body>

 <h1>Homepage</h1>

…

</body>
</html>
Connection closed by foreign host.
```

Richiesta e risposta dove viene indicata directory senza "/" finale (il browser deve fare due richieste)

```
GET /~avf HTTP/1.1
Host: www.myserver.unimi.it
------------------------------------------------------------------------------
HTTP/1.1 301 Moved Permanently
Date: Tue, 06 Jan 2006 13:49:15 GMT
Server: HP Apache-based Web Server/1.3.27 (Unix) mod_perl/1.27 PHP/4.2.2
Location: http:// www.myserver.unimi.it /~avf/
Transfer-Encoding: chunked
Content-Type: text/html; charset=iso-8859-1

148
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML><HEAD>
<TITLE>301 Moved Permanently</TITLE>
</HEAD><BODY>
<H1>Moved Permanently</H1>
The document has moved <A HREF="http://www.myserver.unimi.it/~avf/">here</A>.<P>
<HR>
<ADDRESS>HP Apache-based Web Server/1.3.27 Server at www.myserver.unimi.it Port 80</ADDRESS>
</BODY></HTML>

0

Connection closed by foreign host.
```