

## Risposte quesiti teorici

### Quesito n. 3 appello del 12 settembre 2012

Si scriva la risposta ad una richiesta http avente avuto esito negativo in quanto sintatticamente errata, inviata da un Web server di tipo Microsoft-IIS ad una macchina Linux Fedora, in data e ora correnti. Si precisa che la dimensione della risorsa scambiata (una immagine gif) è di 180.92 kB, il suo ultimo aggiornamento risale al 7 settembre 2012 alle 05.20 e il server ha impostato un timeout di 91s sulla connessione con un numero massimo di scambi ammessi pari a 99. Si tralasci il corpo della reply.

### RISOLUZIONE

HTTP/1.1 400 Bad Request

Server: Microsoft-IIS/5.1

Date: Wed, 12 Sep 2012 09:00:00 GMT

Content-Type: image/gif

Content-Length: 185263

Last-Modified: Fri, 07 Sep 2012 05:20:00

Keep-Alive: timeout=91, max=99

Connection: Keep-Alive

### COMMENTO

Lo status code che si riferisce ad un errore sintattico è il 400. L'informazione inerente la macchina Fedora è irrilevante ai fini della composizione del messaggio di response. Anche se non esplicitamente detto, è necessario aggiungere l'header "Connection: Keep-Alive" poiché, per quanto detto nell'RFC 2068, la presenza dell'header "Keep-Alive" obbliga il server ad aggiungere anche l'header "Connection: Keep-Alive" nonostante il protocollo http utilizzato risulta essere 1.1 (con questa versione è implicitamente una connessione persistente quindi, di solito, il connection keep alive non viene esplicitato dai server)

### **Quesito n. 1 appello del 14 novembre 2013**

Costruire il pacchetto http di risposta da parte del server Microsoft IIS relativo al dominio poliba.it al tentativo di accesso al realm riservato /ruta. Si ipotizzi che la replica avvenga in data e ora correnti e in riferimento alla versione 1.0 del protocollo.

#### **RISOLUZIONE**

Nel messaggio di risposta ometto l'entity-body il quale contiene una pagina html di errore auto generata dal server. Utilizzo come metodo di autorizzazione la Basic Access Authentication. Ipotizzo che il client abbia inviato una request con protocollo 1.0 e senza esplicitare l'header "Connection:".

Con Microsoft IIS 5.1 risulta:

HTTP/1.1 401 Access Denied

Server: Microsoft-IIS/5.1

Date: Thu, 14 Nov 2013 15:00:00 GMT

WWW-Authenticate: Basic realm="poliba.it"

Content-Length: 4431

Content-Type: text/html

Con Microsoft IIS 7.5 risulta:

HTTP/1.1 401 Unauthorized

Server: Microsoft-IIS/7.5

Date: Thu, 14 Nov 2013 15:00:00 GMT

WWW-Authenticate: Basic realm="poliba.it"

X-Powered-By: ASP.NET

Content-Length: 1218

Content-Type: text/html

Connection: close

#### **COMMENTO:**

Vista l'ambiguità della traccia, è bene specificare le ipotesi seguite in modo tale da fare comprendere a chi corregge quello su cui si sta ragionando. La traccia richiede la risposta REALE del server quando questo genera uno status code 401. Tale risposta varia leggermente a seconda della versione del server Microsoft IIS, ed è quindi ottimale porre ad esempio più messaggi di risposta. L'informazione riguardante il nome del realm "/ruta" è irrilevante ai fini del messaggio di risposta. E' importante considerare, invece, che la response deve essere fornita in riferimento ad una request con protocollo http 1.0 (NON E' il server che deve rispondere con un messaggio in http 1.0!!) poiché il comportamento dei server varia in funzione di questo fattore. Ad esempio, con Microsoft IIS 5.1 ed una request versione 1.1 è necessario aggiungere l'header "Connection: close" mentre con Microsoft 7.5 ed una request versione 1.1 deve essere eliminato il "Connection: close".

### **Quesito n. 3 appello del 14 novembre 2013**

Si costruisca il file di configurazione vsftpd.conf chiarendo in via preliminare di cosa si tratta e sapendo che il server dovrà ammettere un numero massimo di connessioni client contemporanee pari a 5000 di cui 100 provenienti dalla medesima sorgente e che dovranno essere abilitati tutti gli accessi possibili senza alcun tipo di vincolo.

#### **RISOLUZIONE**

Il file vsftpd.conf è il file testuale di configurazione del server FTP open-source per sistemi Linux chiamato “Very Secure FTP Daemon”. Al suo interno è possibile inserire delle direttive che permettono di specificare le regole comportamentali del server. Di default è presente nella cartella “/etc” o “/etc/vsftpd” a seconda della versione del software considerata.

```
max_clients=5000
max_per_ip=100
anonymous_enable=YES
local_enable=YES
anon_upload_enable=YES
```

#### **COMMENTO:**

La traccia presenta una richiesta generica di abilitazione di tutti gli accessi possibili senza alcun tipo di vincolo. Tale affermazione va considerata in funzione delle sole direttive approntate durante il corso di studi che sono le cinque esposte nella risoluzione. In particolare, il passaggio “dovranno essere abilitati tutti gli accessi possibili” si riferisce al porre a valore vero le direttive “anonymous\_enable” e “local\_enable”, mentre “senza alcun tipo di vincolo” esplicita la richiesta di poter far compiere tutte le operazioni possibili agli utenti abilitati e, nel nostro caso, l'unica direttiva che rientra in questa richiesta è la “anon\_upload\_enable” la quale deve essere posta a valore vero.

Le keyword “YES” e “NO” per le direttive booleane sono quelle accettate di default dal server vsFTPd, tant'è che il file di configurazione standard esplicita tutte le direttive booleane con questi valori. In alternativa, è possibile utilizzare anche i valore “yes”, “no”, “true”, “false”, “TRUE”, “FALSE”. E' importante sottolineare che prima e dopo il carattere “=” delle direttive non vi deve essere alcuno spazio.

#### **Quesito n. 4 appello del 25 luglio 2012**

Si provveda a spiegare, motivando opportunamente la risposta, la rilevanza teorica del modello ISO/OSI ai fini della trattazione dei protocolli di telecomunicazione (e in particolare dei protocolli applicativi) secondo uno schema layerizzato.

#### **RISOLUZIONE**

L'ISO/OSI (International Organization for Standardization / Open System Interconnection) è un modello standardizzato che definisce delle linee guida generali a cui gli host (elaboratori, device di rete etc...) facenti parte di una rete informatica, quindi interconnessi gli uni con gli altri, devono attenersi per poter comunicare tra loro. Esso non definisce in modo specifico i tipi di servizi o protocolli da implementare ma delinea gli obiettivi di una architettura di rete.

Formalmente è costituito da uno schema a layer (livelli) ognuno dei quali svolge determinati task. Ogni livello ottiene delle informazioni dal livello inferiore, le rielabora in relazione dei propri obiettivi e funzionalità e le offre al livello superiore attraverso dei servizi. Questa schematizzazione permette ad un livello superiore di non preoccuparsi delle problematiche svolte dai livelli inferiori e di concentrarsi sui propri compiti.

Per i protocolli applicativi, ovvero i protocolli che implementano i task afferenti il layer applicativo, questo schema layerizzato permette di focalizzarsi solo sulle logiche di livello utente, ovvero la manipolazione e la elaborazione del contenuto informativo. Tutte le logiche di basso livello, ovvero la gestione dei problemi legati al mezzo trasmissivo, alla strutturazione dei dati, alle modalità con cui far raggiungere i messaggi ai destinatari, al controllo degli errori, non sono contemplate a livello applicativo.

A titolo di esempio, ipotizziamo di dover implementare un server HTTP. Supponiamo che ci sia arrivata una request da un client. Le implementazioni di cui dobbiamo occuparci riguardano il parsing del messaggio, in modo tale da poter estrapolare le intestazioni, e il comportamento del server, il quale deve costruire il messaggio di risposta e inviarlo. Nello sviluppo di queste logiche utente, ignoriamo i problemi di basso livello, cioè assumiamo, ad esempio, che il contenuto informativo della request sia corretto (cioè che gli header arrivati siano gli stessi che sono partiti) o che gli eventuali messaggi inviati dal server arrivino a destinazione e arrivino corretti.

#### **COMMENTO**

La domanda sostanzialmente chiede “perché il modello layerizzato è così importante nel modello ISO/OSI?”. La risposta deve, quindi, concentrarsi su questo trattando in particolare il livello applicazione.

### Quesito n. 2 appello del 14 novembre 2013

Si individuino le possibili incongruenze nel campo FLAGS del seguente pacchetto DNS.  
Preliminarmente, per ciascun campo si indichi il rispettivo significato si costruisca infine la codifica compatta:

1 . . . . . =  
. 000 1 . . . . . =  
. . . . 0 . . . . . =  
. . . . 1 . . . . . =  
. . . . 1 . . . . . =  
. . . . 0 . . . . . =  
. . . . 0 . . . . . =  
. . . . 1 . . . . . =  
. . . . . 0000 =

#### RISOLUZIONE

1 . . . . . = QR(Query/Response) indica che il messaggio è una risposta  
. 000 1 . . . . . = OPCODE(Operation code) indica che la query richiesta è di tipo  
IQUERY (query inversa)  
. . . . 0 . . . . . = AA(Authoritative answer) indica che il server che ha inviato il  
messaggio di risposta non è autoritativo  
. . . . 1 . . . . . = TC(Truncation) indica che questo messaggio è stato troncato poiché  
supera i 512 byte di lunghezza  
. . . . 1 . . . . . = RD(Recursion desired) indica che la query richiesta deve essere  
risolta ricorsivamente  
. . . . 0 . . . . . = RA(Recursion available) indica che il server non è in grado di  
eseguire query ricorsive  
. . . . 0 . . . . . = Z è un bit di padding posto sempre a zero  
. . . . 1 . . . . . = AD(authentic data) indica che i dati forniti dal server sono autenticati  
. . . . . 0000 = RCODE(response code) rappresenta il codice zero, ovvero “no error”.  
Indica che non ci sono stati errori di alcun tipo del portare a  
compimento la risoluzione

Tenuto conto che questo messaggio di risposta a query inversa è inviato dal DNS Resolver ad un client, le incongruenze sono le seguenti:

Campo AA: La risposta ad una query inversa deve risultare autoritativa poiché è il DNS Resolver stesso a fornirla! Difatti una volta trasformata in PTR e ricevuta la risposta dal server autoritativo della zona in “in-addr.arpa” è poi lui a restituirla direttamente al client come query inversa.

Campi RD ed RA: Se RD è a 1, RA deve essere anche ad uno o, in alternativa, RCODE deve essere posto a 1001 (codice 5 – Refused).

Campo AD: Poichè la query inversa è restituita dal DNS Resolver, non può mai essere applicato un meccanismo di autenticazione (il quale prevede l'interazione tra più SERVER DNS) quindi questo flag deve essere posto a zero.

Il valore del campo FLAGS senza incongruenze e seguendo la logica sopra descritta risulta:

1 0 0 0   1 1 0 1   1 0 0 0   0 0 0 0

In ultimo, calcoliamo la forma compatta del campo FLAGS fornito dalla traccia:

Binario:        1 0 0 0   1 0 1 1   0 0 1 0   0 0 0 0

Esadecimale:     8            B            2            0

Forma compatta: 0x8B20

## COMMENTO

Descritto il significato di ogni valore dei bit presenti ed esposta la forma compatta che è semplicemente una conversione da binario a esadecimale, ciò che è importante esplicitare è la logica con cui si sono determinate le incongruenze tra i vari campi.

## RISPOSTA VECCHIA, CORRETTA MA ESULA DAL CORSO (RUTA VUOLE QUELLA SOPRA)

La prima incongruenza riguarda il campo RCODE. Nell'RFC3425 è esplicitamente detto che il valore 1 per il campo OPCODE è OBSOLETE ed ogni server che riceve un messaggio con tale valore deve restituire una risposta con RCODE = 0100 (errore 4 - Not Implemented). Assumendo, quindi, che questo messaggio sia un messaggio di errore e che pertanto non abbia con sé alcuna risoluzione (campi ANSWER, AUTHORITY, ADDITIONAL sono vuoti) una seconda incongruenza è legata al campo AD poiché se la risoluzione non è stata eseguita non è possibile che sia stato indicato che le risposte sono state autenticate. (AD va posto a zero)

In ultimo, è possibile ipotizzare una ulteriore incongruenza nel campo TC: difatti se questo è un messaggio di risposta di errore, sono presenti solo i valori delle intestazioni che in totale occupano 96 bit (12 byte). Di conseguenza il TC dovrebbe essere a zero.

Le strade da seguire possono essere diverse e non devono necessariamente coincidere con quella sopra esposta. Ad esempio, possiamo seguire questa seconda risoluzione.

Invertiamo la logica iniziale e assumiamo che il campo incongruente non sia RCODE ma risulti OPCODE. Questo significa che per eliminare l'incongruenza pongo OPCODE = 0000. Fatto questo, l'incongruenza successiva può essere trovata nel campo RA (Recursion available) poiché, essendo posta a zero, il server dovrebbe generare un RCODE=1001 (codice 5 – Refused). Questo è dovuto al fatto che chi ha inviato la query richiede esplicitamente una risoluzione ricorsiva (RD=1) ma il server non è in grado di eseguirla. Per risolvere tale incongruenza, si può pensare di porre RD=0 in modo tale da evitare la generazione dell'errore. Con questa scelta i campi TC e AD risultano congruenti poiché il messaggio di risposta include dei valori in ANSWER, AUTHORITY, ADDITIONAL e, pertanto, può essere stato troncato e le risposte possono essere state autenticate.

#### **Quesito n. 4 appello del 14 novembre 2013**

Si scriva il listato di una sessione SMTP nella quale l'utente ruta accreditato presso il server deemail.poliba.it richieda la trasmissione in data e ora correnti di una mail dalla capienza di 128KB con oggetto "Risultati Esame" all'utente [prova@fastwebnet.it](mailto:prova@fastwebnet.it) accreditato presso il mail server mail.fastwebnet.it. Alla mail risulta allegata una immagine jpg.

#### **RISOLUZIONE**

Denoto con "C" (client) i comandi inseriti dall'utente e con "S" (server) i messaggi di risposta del server SMTP del ricevente.

S: 220 mail.fastwebnet.it ESMTP Service (7.0.027) ready

C: HELO deemail.poliba.it

S: 250-mail.fastwebnet.it

C: MAIL FROM:<ruta@poliba.it> SIZE=131072

S: 250 MAIL FROM:<ruta@poliba.it> OK

C: RCPT TO:<prova@fastwebnet.it>

S: 250 RCPT TO:<prova@fastwebnet.it> OK

C: DATA

S: 354 Start mail input; end with <CRLF>.<CRLF>

C: Message-ID: <messageid@poliba.it>

Date: Thu, 14 Nov 2013 17:00:00 +0200

MIME-Version: 1.0

From: Ruta <ruta@poliba.it>

To: prova@fastwebnet.it

Subject: Risultati Esame

Content-Type: image/jpeg

.

S: 250 <messageid> Mail accepted

C: QUIT

Con "messageid" indico un codice identificativo assegnato dal server SMTP del ricevente a quel particolare messaggio.

#### **COMMENTO**

La traccia richiede una "SESSIONE", ovvero l'insieme dei comandi digitati dall'utente e le risposte del server SMTP con il quale si sta comunicando. Tutte le informazioni esplicitate nella traccia sono utili al fine della risoluzione. In particolare, la data, il soggetto e il tipo di file in allegato vanno specificati nell'intestazione del messaggio, mentre la dimensione è specificata con keyword "SIZE"

assieme al comando "MAIL TO". Da sottolineare è il carattere "." che indica al server la fine del messaggio.

### **Quesito n. 3 appello 25 luglio 2012**

Si applichi l'algoritmo RLE alla codifica della seguente immagine RGB, quantificando in percentuale l'eventuale compressione ottenuta:

```
2233DD443333
AAA443333333
22222BBBB444
111111111111
00AAAAA66CCC
55555222222
44455222222
```

### **RISOLUZIONE**

```
2#23#2D#24#23#4
A#34#23#7
2#5B#44#3
1#12
0#2A#56#2C#3
5#62#6
4#35#32#6
```

Caratteri iniziali: 84

Caratteri finali: 64

Percentuale di compressione:  $[(84-64) \times 100] / 84 = 23,8 \%$

### **COMMENTO**

Anche se la traccia specifica che il contenuto mostrato è una immagine RGB, l'algoritmo va applicato come se fosse una stringa, quindi individuando i caratteri che si ripetono e sostituendoli con carattere + "#" (carattere di escape) + numero di ripetizioni. La percentuale di compressione indica di quanti caratteri si è ridotto il contenuto esprimendolo in percentuale rispetto al totale. Quindi  $84-64 = 20$  sono i caratteri "risparmiati" che rappresentano il 23,8 % di tutti gli 84 caratteri. NOTA: In RLE, i simboli non ripetuti devono comunque essere codificati con la sintassi standard ovvero: Simbolo + "#" + 1 (Esempio 33ABB → 3#2A#1B#2)



### **Quesito n. 1 appello del 25 luglio 2012**

Si spieghi il significato e contenuto della seguente replica http:

```
HTTP/1.1 206 Partial content
Date: Wed, 15 Nov 1995 06:25:24 GMT
Last-Modified: Wed, 15 Nov 1995 04:58:08 GMT
Content-Range: bytes 21010-47021/47022
Content-Length: 26012
Content-Type: image/gif
```

### **RISOLUZIONE**

Il contenuto esposto è un messaggio di risposta (response) inviato da un server HTTP. Indica che è stata accettata la request di un client che richiedeva tramite metodo GET o HEAD (probabilmente quest'ultimo visto la mancanza del body) di recuperare solo una parte di una certa risorsa di tipo immagine gif, specificando soltanto un singolo range nell'intestazione "Range:". Analizziamo nella sua interezza il messaggio.

La prima riga è la "Response-Line" la quale contiene la versione del protocollo (HTTP/1.1), lo status code (206 appartenente alla classe "Successful") e la frase descrittiva di quest'ultimo ("partial content").

La seconda riga è una intestazione che indica la data in cui il messaggio è stato inviato dal server, nel caso particolare mercoledì 15 novembre 1995 alle ore 6 e 25 circa.

La terza riga è una intestazione che indica la data di ultima modifica del file di cui si sta recuperando il contenuto parziale, nello specifico mercoledì 15 novembre 1995 alle ore 4 e 58 circa.

La quarta riga è una intestazione che il server inserisce per indicare dove si collocano i byte trasmessi nell'entity-body rispetto alla risorsa completa, in questo caso tra il byte 21010 e il 47021. E' inoltre indicata la lunghezza totale della risorsa di cui si sta inviando solo un contenuto parziale, in questo caso 47022 byte.

La quinta riga è una intestazione che indica la lunghezza dell'entity-body e in questo caso coincide proprio con la lunghezza del contenuto parziale.

L'ultima riga è una intestazione indicante il tipo della risorsa di cui si sta inviando il contenuto parziale.

### **COMMENTO:**

La traccia è chiara. Si noti semplicemente che quanto presentato in questo quesito è reperibile nell'RFC 2616 a pagina 123 ed è completamente spiegato (in inglese).

## Quesito n. 2 del 25 luglio 2012

Si chiarisca la seguente sinossi spiegando in via preliminare a cosa fa riferimento:

```
set-cookie      =      "Set-Cookie:" cookies
cookies         =      1#cookie
cookie          =      NAME "=" VALUE *("; " cookie-av)
NAME            =      attr
VALUE           =      value
cookie-av       =      "Comment" "=" value
                  |      "Domain" "=" value
                  |      "Max-Age" "=" value
                  |      "Path" "=" value
                  |      "Secure"
                  |      "Version" "=" 1*DIGIT
```

## RISOLUZIONE

La sinossi presentata costituisce le regole sintattiche dell'header "Set-Cookie". Questo è una intestazione che i server HTTP inseriscono nelle loro response quando vogliono instaurare una sessione con un client attraverso l'utilizzo di Cookie. L'header in questione è definito nell'RFC 2109. Analizziamola nel dettaglio.

La sinossi indica che l'intestazione deve essere inserita con il field name "Set-Cookie:" seguito dal cookie vero e proprio. La seconda riga specifica che il numero di cookie che si possono specificare è pari a uno, difatti se si vogliono settare più cookie in una stessa response è necessario ripetere l'header "Set-Cookie" più volte nella response. Le righe successive definiscono la sintassi con cui ogni informazione di stato del cookie deve essere scritta, in particolare si deve seguire la sintassi NOME = VALORE seguito dal carattere ";". In ultimo, sono esplicitati i parametri aggiuntivi relativi alla particolare coppia NOME/VALORE definita. Anche questi devono essere separati dal carattere ";". I parametri sono i seguenti:

**Comment** → E' un parametro opzionale che include una descrizione sulla coppia NOME/VALORE a cui fanno riferimento. In particolare, il server potrebbe comunicare al client che si tratta di una informazione sensibile in modo tale da permettere a quest'ultimo di scegliere se continuare la sessione o meno

**Domain** → Indica il dominio a cui il cookie deve far riferimento

**Path** → Indica L'URL a cui il cookie fa riferimento

**Max-Age** → Indica il tempo di vita del cookie. Scaduto questo il client deve scartarlo.

**Secure** → Non ha valori. Se presente, indica al client che la sessione va proseguita con un metodo di sicurezza sicuro come ad esempio HTTPS

**Version** → Indica la versione a quale il cookie fa riferimento. Quella dell'RFC 2109 è la versione 1.0

Un esempio di utilizzo è il seguente:

```
Set-Cookie: Utente=Davide; Domain=www.davide.it; Path=/account; Max-Age=3600; Secure
```

## COMMENTO

La traccia è chiara. Si noti che questa sinossi si trova nell'RFC 2109 a pagina 4.

### **Quesito n. 2 del 25 luglio 2012**

Si scrivano le direttive del file di configurazione vsftpd.conf corrispondenti alle seguenti impostazioni:

- 150000 connessioni possibili di cui 1000 provenienti dalla medesima macchina;
- abilitazione dell'accesso per gli utenti locali;
- disabilitazione dell'accesso per gli utenti non autenticati.

### **RISOLUZIONE**

Le direttive da aggiungere nel file di testo vsftpd.conf, presente nella cartella “/etc” o “/etc/vsftpd” su sistemi Linux, sono le seguenti:

```
max_clients=150000  
max_per_ip=1000  
local_enable=YES  
anonymous_enable=NO
```

### **COMMENTO**

Per approfondire la tipologia di esercizio, guarda il quesito n. 3 del 14 novembre 2013.

## Quesito n. 1 del 12 settembre 2012

Si spieghi il significato e contenuto del seguente interscambio client/server http:

```
1. User Agent -> Server
    POST /acme/login HTTP/1.1
    [form data]
2.  Server -> User Agent
    HTTP/1.1 200 OK
    Set-Cookie: Customer="WILE_E_COYOTE"; Version="1"; Path="/acme"
3.  User Agent -> Server
    POST /acme/pickitem HTTP/1.1
    Cookie: $Version="1"; Customer="WILE_E_COYOTE"; $Path="/acme"
    [form data]
4.  Server -> User Agent
    HTTP/1.1 200 OK
    Set-Cookie: Part_Number="Rocket_Launcher_0001"; Version="1";
    Path="/acme"
5.  User Agent -> Server
    POST /acme/shipping HTTP/1.1
    Cookie: $Version="1";
    Customer="WILE_E_COYOTE"; $Path="/acme";
    Part_Number="Rocket_Launcher_0001"; $Path="/acme"
    [form data]
6.  Server -> User Agent
    HTTP/1.1 200 OK
    Set-Cookie: Shipping="FedEx"; Version="1"; Path="/acme"
7.  User Agent -> Server
    POST /acme/process HTTP/1.1
    Cookie: $Version="1";
    Customer="WILE_E_COYOTE"; $Path="/acme";
    Part_Number="Rocket_Launcher_0001"; $Path="/acme";
    Shipping="FedEx"; $Path="/acme"
    [form data]
8.  Server -> User Agent
    HTTP/1.1 200 OK
```

## RISOLUZIONE

Il contenuto mostrato rappresenta una sessione di comunicazione tra client e server, costituita da diverse request e response, in cui è presente l'utilizzo di cookie. Analizziamo nel dettaglio ogni messaggio:

1. Inizialmente lo user agent non possiede alcun cookie. Invia una richiesta con metodo POST il cui entity-body è costituito dalle informazioni inserite dall'utente in una form html. Essendo L'URL "acme/login", possiamo ipotizzare che i dati inviati siano le credenziali di accesso ad un account registrato sul server.
2. Il server risponde con un esito positivo, settando il cookie "Customer=WILE\_E\_COYOTE" e specificando che questo fa riferimento all'URL "/acme". In questo modo il server sta memorizzando lo stato per cui l'utente ha effettuato il login.
3. L'user agent invia nuovi dati all'URL "/acme/pickitem" il quale ci fa intuire che il contenuto dell'entity-body è rappresentato da un oggetto in vendita su un e-shop. Con questo passaggio, si sta chiedendo al server di aggiungere tale oggetto al carrello. Inoltre il client setta invia il Cookie prima settato dal server per notificargli che ha già effettuato il login
4. Il server risponde positivamente alla richiesta di aggiunta nel carrello ed invia un nuovo Cookie "Part\_Number=Rocket\_Launcher\_0001", sempre riferito all'URL "/acme" per memorizzare nella sessione che l'utente ha un oggetto nel carrello

5. A questo punto il client invia altri dati all'URL “/acme/shipping” in cui presumibilmente ha inserito i dati relativi alla spedizione dell'oggetto. Setta, inoltre, i due Cookie precedentemente inviati dal server per ricordargli di essere loggato e di avere un oggetto nel carrello
6. Il server accetta i dati sulla spedizione e setta il Cookie “Shipping=FedEx” per memorizzare il fatto che i dati sulla spedizioni sono stati inviati dall'utente
7. L'user agent invia le ultime informazioni all'URL “/acme/process” che, presumibilmente, sono rappresentate da un elemento form html di tipo “Button” che permette all'utente di confermare l'acquisto. Invia attraverso l'intestazione “Cookie” tutti i Cookie che fino a quel momento si sono scambiati Client e Server in modo tale che quest'ultimo abbia tutte le informazioni per concludere la transazione di acquisto con successo
8. Il server invia un messaggio di conferma dell'ordine di acquisto al client

L'intercambio client/server, pertanto, rappresenta una sessione in cui un utente effettua un login su un negozio online, sceglie un oggetto, inserisce i dati sulla spedizione e conferma l'ordine.

#### COMMENTO:

La traccia richiede di spiegare il significato dei messaggi HTTP, il quale è descritto nelle prime righe della risoluzione e nelle ultime due, e il contenuto di ogni messaggio, descritto passo per passo nell'elenco. La risposta si basa sostanzialmente sull'indizio fornito dalle Request-URL le quali con i loro nomi permettono di comprendere che tipo di operazione l'utente sta compiendo. E' da sottolineare che tutto il contenuto presente nella traccia è reperibile nell'RFC 2109 a pagina 12 e 13 con annessa spiegazione (in inglese).

#### **Quesito n. 4 appello del 12 settembre 2012**

Si provveda a spiegare in modo conciso ma compendioso il meccanismo della Digest Access Authentication identificando con chiarezza i suoi punti di forza ma anche le problematiche ad essa connesse.

#### **RISOLUZIONE**

Il Digest Access Authentication è un metodo di autenticazione utilizzato nel protocollo HTTP. Tale metodo permette ad un utente che vuole accedere ad una risorsa localizzata in un'area riservata di un server, chiamata "realm", di autenticarsi. E' una tecnica di tipo "challenge-response", ovvero l'autenticazione avviene attraverso la formulazione di una "challenge" da parte del server e di una successiva "response" del client con le informazioni utili all'autenticazione. Essa è costituita principalmente da tre passi:

1. Il client invia una richiesta di accesso ad una risorsa appartenente ad un'area riservata senza averne l'autorizzazione
2. Il server formula un "challenge" con il quale notifica al client che quella risorsa appartiene ad un area riservata e che è necessario inviare le credenziali di accesso. Nel caso della Digest, il server HTTP invia una intestazione aggiuntiva di questo tipo:  
`WWW-Authenticate: Digest realm="nomerealm",  
nonce="codicenonce", opaque="codiceopaque"`  
dove al posto di "nomerealm", "codicenonce" e "codiceopaque" il server genera degli opportuni valori. In particolare, il parametro "realm" permette di identificare il nome dell'area riservata alla quale si accede, il "nonce" è una stringa di codifica base64 il cui scopo è ridurre gli attacchi tipo "reply" e "man-in-the-middle" e "opaque" è una stringa generata dal server che deve essere restituita dal client in tutti i messaggi successivi per indicare che quei messaggi fanno tutti parte dello stesso dominio di validità.
3. Il client risponde con una "reponse" in cui invia le credenziali per quel particolare realm. Per fare questo, aggiunge tra le sue intestazioni la seguente:  
`Authorization: Digest username="codiceusername",  
realm="nomerealm", uri="urlrequest", nonce="codicenonce",  
response="codiceresponse", opaque="codiceopaque"`  
in cui tutti i parametri già presenti nel challenge vengono inviati uguali, "codiceusername" è l'username inserita dall'utente, "urlrequest" è l'URL della risorsa che si sta richiedendo e "codiceresponse" è una stringa codificata in MD5 (32 caratteri) i cui valori di input includono realm, username e password.

Una volta che il server ha ricevuto il "response", può confrontarlo con una stringa da lui generata con i parametri corretti in suo possesso. Se le due stringhe coincidono, l'autenticazione è riuscita.

Il suo principale punto di forza è che è possibile verificare un utente senza ricorrere a tecniche di autenticazione più elaborate, permettendo di inviare password crittografate e, quindi, difficili da carpire.

Di contro, si misurano due principali problemi. Il primo è che i messaggi http sono inviati in chiaro e quindi possono essere soggetti ad attacchi replay e man-in-the-middle, il secondo è che il metodo non prevede nessuna impostazione di sicurezza iniziale con la quale far arrivare la password al server per la prima volta: è quella che potremmo definire la "registrazione" delle credenziali presso il server.

#### **COMMENTO**

La domanda è chiara. Tutte le informazioni si possono reperire nell'RFC 1035.

### **Quesito n. 1 appello del 30 novembre 2012**

Si provveda a spiegare il significato dei seguenti header http chiarendo, per ciascuno di essi, se fa parte di un pacchetto di richiesta o di risposta:

Host:

If-Match:

Proxy-Authorization:

### **RISOLUZIONE**

Host:

E' una intestazione presente solo nelle request. A partire da HTTP 1.1 è divenuta obbligatoria. Specifica il nome di dominio e la porta TCP del server su cui è presente la risorsa che si sta richiedendo. Se la porta TCP è assente si assume quella standard (80 per HTTP). E' indispensabile per implementare il virtual hosting, ovvero fare in modo che più domini possano fare riferimento ad una stessa macchina (o ad un pool di macchine).

If-Match:

E' una intestazione presente solo nelle request. E' un header condizionale. Il valore specificato è un "Etag", ovvero un codice assegnato dal server che permette di identificare lo stato di una certa risorsa. La condizione prevista afferma che se la risorsa richiesta ha un Etag uguale a uno di quelli specificati come valore di questa intestazione, allora il server deve accettare la request inviata. In caso contrario, il server deve inviare uno status code 412 (Precondition failed).

Proxy-Authorization:

E' una intestazione presente solo nelle request. Viene aggiunta quando un proxy risponde ad una request con uno status code 407 (Proxy Authentication required). Svolge la stessa funzione dell'intestazione "Authorization" con la differenza che il messaggio è diretto agli eventuali proxy presenti tra client e server. Se sono presenti più proxy, la direttiva viene recepita dal primo della catena che ha richiesto l'autenticazione. Permette, quindi, di autenticarsi presso un proxy che richiede autenticazione inviando le credenziali di accesso.

### **COMMENTO**

La traccia è chiara. Tutte le informazioni utili possono essere reperite nell'RFC 2616.

## **Quesito n. 2 appello del 30 novembre 2012**

Si scriva il listato di una sessione SMTP nella quale l'utente m.ruta accreditato presso il server poliba.it richieda la trasmissione in data e ora attuali di una mail della capienza di 12KB con oggetto "Prova SMTP" all'utente [studente@libero.it](mailto:studente@libero.it) accreditato presso il server msmtplibero.it

### **RISOLUZIONE**

Identifico con "C:" (client) i comandi inviati dall'utente e con "S:" le risposte del server.

S: 220 msmtplibero.it ESMTP Server (7.0.025) ready

C: HALO poliba.it

S: 250-msmpt.libero.it

C: MAIL FROM:<m.ruta@poliba.it> SIZE=12288

S: 250 MAIL FROM:<m.ruta@poliba.it> OK

C: RCPT TO:<studente@libero.it>

S: 250 RCPT TO:<studente@libero.it> OK

C: DATA

S: 354 Start main input; end with <CRLF>.<CRLF>

C: From: Michele Ruta <m.ruta@libero.it>

To: studente@libero.it

Subject: Prova SMTP

Data: Fri, 30 Nov 2012 08:00:00 +0200

.

S: 250 <mainID> Mail accepted

C: QUIT

Nota che "mailID" è un ID assegnato dal server a quel messaggio.

### **COMMENTO**

Vedi quesito n. 4 del 14 novembre 2013.



## Quesito n. 4 appello del 30 novembre 2012

Si provveda a decifrare il pacchetto seguente:

### Answers

```
www.google.it: type CNAME, class IN, TTL
3dlh1m19s, cname www.google.com
```

```
www.google.com: type CNAME, class IN, TTL 8m18s,
cname www.l.google.com
```

```
www.l.google.com: type A, class IN, TTL 23s, addr
66.249.85.104
```

```
www.l.google.com: type A, class IN, TTL 23s, addr
66.249.85.99
```

### Authoritative nameservers

```
l.google.com: type NS, class IN, ns b.l.google.com
```

```
l.google.com: type NS, class IN, ns c.l.google.com
```

```
l.google.com: type NS, class IN, ns d.l.google.com
```

```
l.google.com: type NS, class IN, ns a.l.google.com
```

### Additional records

```
b.l.google.com: type A, class IN, addr 64.233.179.9
```

```
c.l.google.com: type A, class IN, addr 64.233.161.9
```

```
d.l.google.com: type A, class IN, addr 64.233.183.9
```

```
a.l.google.com: type A, class IN, addr 216.239.53.9
```

## RISOLUZIONE

Il messaggio proposto è una parte del pacchetto fornito da un DNS resolver ad un client il quale aveva richiesto di effettuare una query standard sul domain name “www.google.it”. In questa parte di pacchetto sono illustrate le risposte ottenute dal DNS resolver, i server autoritativi coinvolti e le risposte aggiuntive.

Le answer mostrano che il DNS resolver, nel processo di risoluzione di quell'indirizzo, ha ricevuto inizialmente una risposta con un resource record di tipo CNAME poiché il nome di dominio fornito dall'utente “www.google.it” è in realtà un alias del dominio “www.google.com”. Il DNS resolver ha quindi tentato di provvedere alla risoluzione di quest'ultimo ma anche in questo caso ha ricevuto una risposta con record CNAME poiché “www.google.com” è in realtà un alias del nome di dominio “www.l.google.it”. A questo punto la richiesta di risoluzione di quest'altro indirizzo va a buon fine e il DNS resolver riceve in risposta due record di tipo A i quali contengono i due indirizzi IP associati al dominio “www.google.it”. Il fatto che siano associati più indirizzi IP a quello stesso dominio significa che sul dominio in questione si sta applicando la tecnica di “load balancing” grazie alla quale le richieste a quel dominio sono ripartite tra diversi server.

La sezione authoritative mostra i server DNS che risultano autoritativi per la zona del dominio che si è chiesto di risolvere, ovvero “l.google.com”.

Infine la sezione additional fornisce una corrispondenza tra nome di dominio dei vari server autoritativi e il loro indirizzo IP.

COMMENTO: La traccia è chiara. Il pacchetto proposto è presente nelle slide del corso.

### **Quesito n. 3 appello del 30 novembre 2012**

Qual è la tipologia di file supportati dal protocollo FTP? E quali le modalità trasmissive ammesse? Per ciascuna categoria si chiariscano le peculiarità ed applicazioni tipiche.

#### **RISOLUZIONE**

In primo luogo è importante sottolineare che l'RFC 959 contenente le indicazioni del protocollo FTP crea una differenza fra “byte logico” e “byte di trasmissione”. Il byte logico indica un byte il cui numero di bit significativi (i bit utilizzati per il contenuto informativo) possono differire dal byte di trasmissione che è invece costituito sempre da 8 bit significativi. Detto questo, i file type (o data type) supportati dal protocollo FTP sono sostanzialmente quattro:

1. ASCII Type – E' utilizzato solo per il trasferimento di file di testo ed è il valore di default se non diversamente specificato in sede di configurazione. Usa un byte logico di 8 bit.
2. EBCDIC (Extended binary coded decimal interchange code) – Rappresenta il tipo di file utilizzati dai mainframe IBM. Era conveniente usarlo per scambiare file testuali quando entrambi gli interlocutori conoscevano questa rappresentazione dei dati. Ad oggi è in disuso.
3. IMAGE type o Binary type – E' un tipo di file in cui le informazioni sono considerate organizzate in byte contigui. E' utilizzato per i file binari e deve essere implementata in tutti i client/server FTP. Il suo byte logico è di 8 bit.
4. Local type – E' un tipo di file che prevede l'utilizzo di un byte logico in cui viene definito dall'utente il numero di bit significativi. E' utilizzato per la creazione di un linguaggio “proprietario” di invio delle informazioni tra un client e un server. In questa maniera un terzo che non conosce il byte logico della comunicazione non può intercettare i dati trasmessi. I bit non significativi vanno rimpiazzati con bit di padding.

Le modalità di trasmissione, in cui non si fa più distinzione fra byte logico ma c'è solo il byte di trasmissione, sono tre:

1. STREAM mode (o streaming) – I dati vengono inviati come flusso di byte. E' utilizzabile per tutti i tipi di file sopra definiti e con i tipi di data structures file e record. Nel caso il data structures sia file, la fine della trasmissione (EOF) avviene con la chiusura della comunicazione da parte del mittente.
2. BLOCK mode – E' un tipo di modalità in cui i dati vengono inviati a blocchi, ognuno dotato di una propria intestazione che definisce informazioni aggiuntive di ogni blocco. Le due intestazioni principali sono “count” che indica la lunghezza del blocco e “description code” che indica la funzione che il blocco inviato assume nel processo di trasferimento. Alcuni di questi codici sono:
  - 128=il blocco è la fine di un record(EOR)
  - 64=il blocco è la fine di un file (EOF)
  - 32=probabili errori nel blocco
  - 16 = il blocco è un “restart marker”
3. COMPRESS mode → Modalità in cui i dati vengono trasmessi applicando l'algoritmo di compressione RLE(run length encoding) al fine di rendere efficiente il trasferimento dei dati.

#### **COMMENTO**

Quanto detto è presente nell'RFC959. Da notare che con “tipologia di file supportati” intende in realtà i DATA type descritti nell'RFC.

## Quesito n.1 appello del 27 settembre 2012

In corrispondenza di ciascuna direttiva di caching fornire la opportuna spiegazione:

“no-cache”

“max-age”

“must-revalidate”

Si scriva anche una richiesta HEAD http 1.1 con scopi di validazione cache sulla risorsa di tipo immagine jpeg avente URI <http://sisinflab.poliba.it>.

## RISOLUZIONE

Le direttive elencate fanno tutte riferimento all'intestazione “Cache-Control”. I solo significati sono i seguenti:

**“no-cache”** → E' una direttiva che indica a tutti i meccanismi di caching attivi durante la comunicazione che la risorsa richiesta non deve mai essere restituita dalla propria cache ma deve essere recuperata dal server di origine

**“max-age”** – > Indica il tempo in secondi per cui la risorsa deve essere ritenuta “fresca”. Oltre questo tempo, la risorsa scade e deve essere rivalidata.

**“must-revalidate”** → E' una direttiva che può specificare solo il server in una response. Comunica ai meccanismi di caching che una volta che una risorsa è scaduta deve essere rivalidata prima di essere fornita a chi la richiede. Se durante la fase di rivalidazione il server di origine non risponde, il server fornisce una risposta 504 (Gateway time out).

La richiesta di validazione da parte di un client è la seguente:

HEAD / HTTP/1.1

Host: sisinflab.poliba.it

Accept: image/jpeg

## COMMENTO

La richiesta di descrizione delle direttive è chiara. Per la richiesta di validazione, è bene ricordare che in una cache possono essere presenti più versioni di una risorsa che fa riferimento ad un determinato URL. In questo caso, in corrispondenza del dominio principale “/” possono essere presenti risorse in cache di diversa natura. E' bene, quindi, specificare il tag “Accept” in modo tale che la cache reperisca la risorsa in cache corrispondente al valore assunto da questa intestazione. Inoltre NON DEVE essere inserito un “Cache-Control: max-age=0” poiché questo fa parte dei metodi di caching successivi al metodo di validazione con head: inserirlo, significherebbe mischiare una metodologia più recente in una più vecchia!

## Quesito n. 2 appello 27 settembre 2012

Si chiarisca il significato del seguente scambio client/server http:

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: Digest
realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
opaque="5ccc069c403ebaf9f0171e9517f40e41"

[...]
Authorization: Digest username="Mufasa",
realm="testrealm@host.com",
nonce="dcd98b7102dd2f0e8b11d0f600bfb0c093",
uri="/dir/index.html",
response="6629fae49393a05397450978507c4ef1",
opaque="5ccc069c403ebaf9f0171e9517f40e41"
```

### RISOLUZIONE

I messaggi presentati rappresentano un request ed un response http. La prima è una response da parte di un server, status code 401, con cui sta comunicando al client che la risorsa richiesta appartiene ad un'area riservata e che, quindi, deve autenticarsi. A tal fine, applica il metodo di autenticazione “Digest Access Authentication” ed invia, tramite l'ausilio dell'intestazione “WWW-Authenticate” un challenge con i seguenti parametri:

**realm** → stringa rappresentante il nome dell'area riservata. Ha lo scopo di rendere noto all'utente del client quali credenziali deve digitare. Viene, inoltre, codificata nel parametro “response” che successivamente invierà il client e che includerà anche le credenziali codificate.

**nonce** → è una stringa ottenuta dalla codifica in base64 di alcuni valori quali timestamp e Etag della risorsa. Permette di prevenire gli attacchi di tipo “reply” e “man-in-the-middle”.

**opaque** → è una stringa generata dal server. Deve essere inclusa in tutti i messaggi che fanno riferimento allo stesso dominio di validazione

Il secondo messaggio è rappresentato dal “response” da parte del client al “challenge” inviato dal server. L'user agent ha chiesto le credenziali di accesso all'utente ed ha provveduto a codificarle, assieme ad altri valori, tramite l'algoritmo di crittografia MD5. Il risultato di questa codifica viene inserito nel parametro “response” dell'intestazione “Authorization” presente nella request del client. Oltre a questo, sono presenti gli stessi parametri arrivati con il challenge del server e che sono rispediti tali e quali più i seguenti:

**username** → è la stringa inserita dall'utente indicante lo username

**uri** → coincide con la Request-URL presente nella Request-Line. Tale valore è ripetuto anche in questo parametro poiché talvolta i server proxy possono modificare la Request-URL

Una volta che il challenge ha ricevuto il contenuto del parametro “response”, può ricalcolarlo anche lui con i valori a sua disposizione, tra cui la user e password corrette, e vedere se c'è corrispondenza (ciò avviene tramite una divisione polinomiale. Se il resto è zero sono uguali). In caso affermativo, il client è autorizzato ad accedere alla risorsa.

### COMMENTO

La traccia è chiara. Si noti che i messaggi presenti nella traccia possono essere reperiti a pagina 18 dell'RFC 2616 con tanto di spiegazione (in inglese).

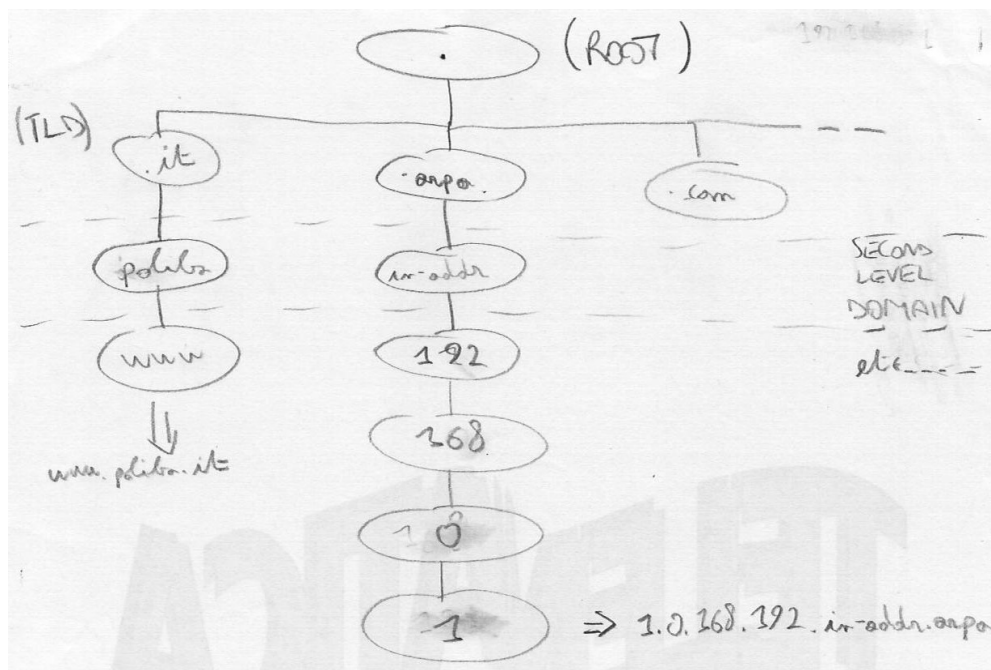
### Quesito n. 3 appello del 27 settembre 2012

Si spieghi l'organizzazione logica dello spazio dei nomi per la risoluzione degli indirizzi internet. In particolare, si schematizzi l'albero DNS chiarendo il ruolo di ciascun livello della gerarchia.

#### RISOLUZIONE

Il DNS (Domain name system) è un servizio il cui scopo è associare (in gergo “risolvere”) un nome di dominio (indirizzo logico) ad un certo indirizzo IP (indirizzo fisico) in modo tale da poter fornire agli utenti che hanno intenzione di accedere a delle risorse presenti in Internet un metodo mnemonicamente più semplice da seguire. La realizzazione delle risoluzioni sfrutta il concetto di “spazio dei nomi”. Per “spazio dei nomi” (o namespace) si intende uno schema organizzativo logico che permette di rappresentare tutti i possibili nomi di dominio assoluti (Full Qualified Domain Name). Un nome di dominio, come definito nell'RFC 1035, è costituito da un insieme di labels, le quali sono separate da un punto (dot). Tale definizione permette di considerare lo spazio di nomi come uno schema gerarchico rappresentabile attraverso un “albero delle gerarchie”, in cui è presente un nodo radice, chiamato “root”, ed una serie di nodi figli, ognuno dei quali identifica una “zona” e sviluppa, a sua volta, altre ramificazioni con altre “zone”. A mano a mano che si scende nella gerarchia, si passa da un livello superiore ad uno inferiore ed è possibile identificare ogni dominio in funzione del proprio livello (top level domain, dominio di secondo livello etc...)

Questa organizzazione permette di individuare, percorrendo le radici di questo albero fino alle foglie, la zona facente riferimento al FQDN che si vuole risolvere. Ogni zona, quindi, è identificata da tutte le label di un nome di dominio che si sono “attraversate” per raggiungere un certo nodo di questa gerarchia. Le varie zone di questa gerarchia sono gestite da dei server DNS ognuno dei quali gestisce un database costituito da RR (resource record) i quali contengono le corrispondenze tra un certo nome di dominio e il loro indirizzo fisico. La schematizzazione dell'albero dei DNS è la seguente:



Ad ogni livello della gerarchia corrisponde una “zona”, gestita da un server DNS autoritativo. Ruolo di questo server è fornire la risoluzione di un nome di dominio se la sua zona corrisponde al FQDN che si sta resolvendo oppure fornire l'indirizzo di un altro server autoritativo, il quale gestisce una zona localizzata ad un livello inferiore della gerarchia. Quindi per risolvere il dominio “www.poliba.it”, il server DNS della zona “.it” fornisce l'indirizzo del server DNS della zona “poliba.it” che a sua volta fornisce l'associazione domain name / indirizzo IP.

#### Quesito n. 4 appello 27 settembre 2012

Si descrivano le caratteristiche del peer wire protocol in BitTorrent, chiarendo la sequenza di interscambio, la tipologia e il formato dei messaggi. Si utilizzi una schematizzazione.

#### RISOLUZIONE

Il peer wire protocol rappresenta un protocollo di comunicazione utilizzato in BitTorrent la cui finalità è facilitare la comunicazione tra i peer e lo scambio di chunk. Si basa sullo scambio di messaggi tra peer allo scopo di mantenere informazioni sulla stato del trasferimento dei dati. Ogni peer mantiene delle informazioni su tutti gli altri peer che stanno condividendo una certa risorsa. Ognuno di questi può trovarsi in uno dei seguenti stati:

**Choked:** Il client corrente, se messo in questo stato da un peer remoto, non può inviare alcuna tipo di richiesta a quel peer remoto poiché questo le scarta.

**Interested:** Il peer corrente, quando messo in questo stato da un peer remoto, significa che possiede qualcosa a cui un peer remoto è interessato. Se il client corrente ha messo il peer remoto in stato di “choked”, il peer remoto potrà cominciare ad effettuare una richiesta non appena sarà messo in stato di “unchoked”.

La gestione di questi stati e delle richieste / trasferimenti di chunk avviene tramite dei messaggi. Il primo messaggio che due peer si inviano per iniziare la comunicazione è quello di “Handshake”, il cui formato può essere schematizzato come segue:

`<pstrlen><pstr><reserved><info_hash><peer_id>`

dove:

`<pstrlen>` → indica la lunghezza del successivo campo `<pstr>`

`<pstr>` → è una stringa indicante il protocollo utilizzato. Nel caso di BitTorrent 1.0 si ha “BitTorrent Protocol”

`<reserved>` → campo di 8 byte riservato non utilizzato

`<info_hash>` → è una stringa di 20 byte rappresentate la codifica SHA1 del campo “info” presente nel file .torrent

`<peer_id>` → sono 20 byte rappresentanti l'id del client

Tutti i messaggi successivi hanno il seguente formato:

`<length_prefix><message_id><payload>`

in cui:

`<length_prefix>` → indica la lunghezza del campo payload

`<message_id>` → indica l'id del messaggio. A seconda dell'id, il messaggio ha un significato diverso. I più importanti sono:

- id=0 – choke = mette il peer in stato choke
- id=1 – unchoke = mette il peer in stato unchoke
- id=2 – interested = mette il peer in stato interested
- id=3 – not interested = mette il peer in stato non interested

id=5 – bitfield = è inviato subito dopo l'handshake e comunica al server quali chunk il peer ha a disposizione. Se il peer non ha chunk, non deve essere mandato.

id=6 – request = richiede un chunk al peer

id=7 – piece = rappresenta il contenuto informativo vero e proprio. È un sottoinsieme di byte facenti parte un chunk richiesto

id=8 – cancel = annulla il trasferimento di uno dei chunk prima richiesti

<payload> → rappresenta il contenuto informativo. A seconda del message\_id può contenere anche dei sottocampi che specificano alcune informazioni aggiuntive.

## COMMENTO

La traccia è chiara. E' possibile specificare ulteriormente il <payload> di alcuni message\_id particolari:

**id=6** (request) ha un payload così formato:

<index><begin><length>

dove “index” indica l'indice del chunk che si vuole ricevere, “begin” specifica l'offset del chunk rispetto al file completo e “length” la lunghezza del chunk.

**Id=7** (piece) ha un payload così formato:

<index><begin><block>

dove “index” indica l'indice del chunk al quale i dati trasferiti fanno riferimento, “begin” l'offset del blocco dei dati che si sta inviando rispetto al chunk e “block” i dati veri e propri

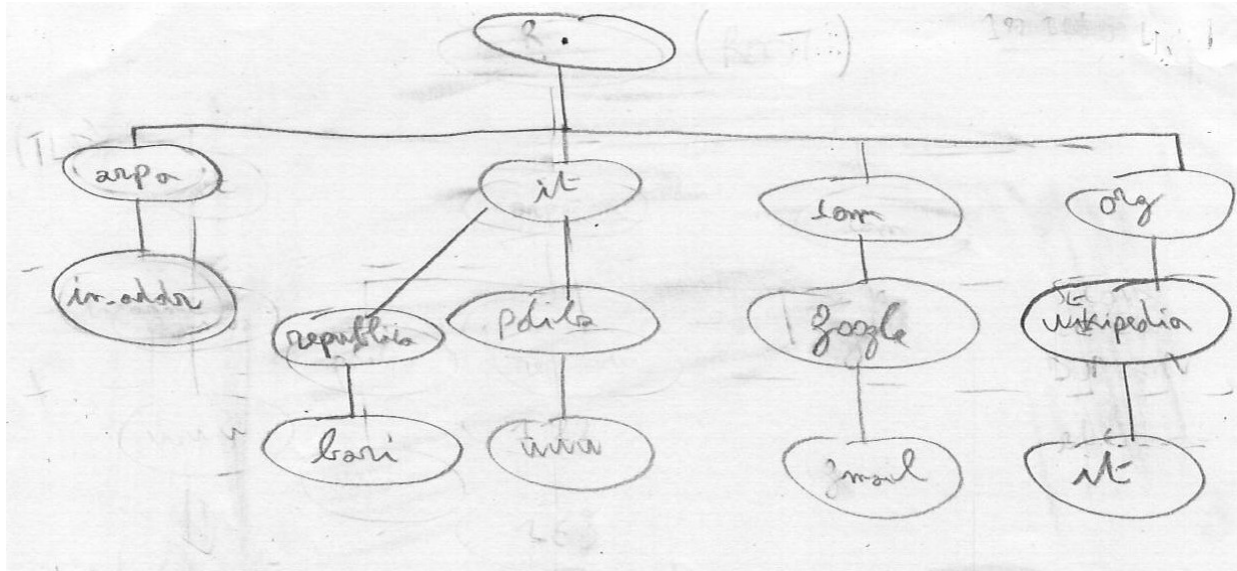
**Id=8** (cancel) ha lo stesso payload del request.

Esiste, inoltre, un altro tipo di messaggio chiamato “Keep-Alive” che segue il formato <len=0000> (senza message\_id e payload). E' un messaggio che viene inviato per tenere in vita la connessione con un peer poiché alcuni di questi, quando non ricevono messaggi per un certo periodo, chiudono la connessione.

### Quesito n. 1 appello 29 novembre 2010

Si schematizzi l'organizzazione dello spazio dei nomi citando il maggior numero di domini di cui si è a conoscenza posizionandoli opportunamente all'interno dello schema. Si provveda inoltre a raggruppare in maniera idonea i domini schematizzati.

### RISOLUZIONE



I domini possono essere raggruppati come segue:

#### DOMINI DI PRIMO LIVELLO

.arpa, .it, .com, .org

#### DOMINI DI SECONDO LIVELLO

in-addr.arpa, repubblica.it, poliba.it, google.com, wikipedia.org

#### DOMINI DI TERZO LIVELLO

bari.repubblica.it, www.poliba.it, mail.google.com, it.wikipedia.org

#### COMMENTO

La traccia è chiara. La risoluzione è stata fatta su un numero molto limitato di domini.



## Quesito n. 2 appello del 29 novembre 2010

Si descriva la struttura di un file .torrent precisando per ciascuno dei campi il significato e la funzionalità rivestita del protocollo BitTorrent.

### RISOLUZIONE

Il file .torrent è un dizionario bencoding contenenti le seguenti chiavi:

**info** → (obbligatorio) è un dizionario che contiene tutte le informazioni sul file

**announce** → (obbligatorio) è l'announce URL del tracker che il client utilizza per inviare request HTTP

**creation date** → indica la data di creazione di questo file torrent

**comment** → stringa di commenti inserita dall'autore del file

**created by** → nome e versione del programma usato per generare questo file

Il dizionario “info”, a sua volta, contiene le chiavi:

**length** → lunghezza del file in byte (solo quando il file da scaricare è uno solo)

**md5sum** → (presente solo se il file da scaricare è 1 solo) è una stringa di 32 caratteri esadecimali ottenuti applicando l'hashing MD5 a tutto il file. Utilizzato per il controllo del file scaricato.

**name** → definisce il nome del file quando il file da scaricare è unico e la directory in cui il file / i file sono salvati per più file.

**piece length** → lunghezza del singolo chunk in byte

**pieces** → è una stringa costituita dalla concatenazione delle fingerprint in SHA1, lunghe 20 byte, di tutti i chunk in cui il file/ i file sono stati suddivisi. Serve per verificare la correttezza dei singoli chunk.

**files** → è una lista di dizionari. Ci sono tanti elementi quanti sono i file da scaricare. Ogni elemento è un dizionario costituito dalle chiavi “Path” che indica il nome o la directory di un particolare file e “length” che indica la grandezza del file.

### COMMENTO

Il quesito è chiaro. Molte informazioni possono essere anche reperite da wikipedia.

### **Quesito n. 3 del 29 novembre 2010**

Si provveda a decifrare il seguente pacchetto, specificando in via preliminare di cosa si tratta:

```
GET /colucci/ HTTP/1.1
Accept: */*
Accept-Language: it, en-us
Accept-Encoding: gzip
User-Agent: Mozilla/4.0 (compatible; MSIE 6.0; Windows NT 5.1; SV1)
Host: cirp.poliba.it
Connection: Keep-Alive
```

### **RISOLUZIONE**

Il pacchetto mostrato è un REQUEST HTTP inviata da un client ad un server HTTP. Analizziamo nel dettaglio ciò che abbiamo.

La prima riga è la Request-Line, la quale esplicita il metodo (GET) la Request-URL (/colucci/) e la versione del protocollo (HTTP/1.1).

La seconda riga è l'intestazione "Accept" la quale permette di specificare al server quali tipo di MIME type il client accetta per la risorsa richiesta. In questo caso, "\*" indica che accetta qualsiasi tipo di MIME type.

La terza riga è l'intestazione "Accept-Language" e indica che il client accetta come linguaggio umano per la risorsa l'italiano e l'inglese americano senza preferirne uno in particolare.

La quarta riga è un header "Accept-Encoding" ed indica al server che il client accetta solo risorse codificate con l'algoritmo di compressione "gzip"

La quinta specifica delle informazioni sullo user agent che ha inviato la richiesta. In particolare, il browser utilizzato è Internet Explorer versione 6 (MSIE 6.0) ed il sistema operativo su cui questo browser gira è Windows XP (Windows NT 5.1).

La sesta riga specifica il nome di dominio dell'host sul quale la risorsa è presente e al quale si sta inviando la request.

L'ultima riga è l'intestazione "Connection: Keep-Alive" con la quale il client indica che la connessione stabilita non deve essere chiusa ma i prossimi messaggi devono essere inviati nella stessa sessione (connessioni persistente).

### **COMMENTO**

La traccia è chiara. Tutte le informazioni possono essere reperite nell'RFC 2616.

### Quesito n. 1 appello del 3 novembre 2011

Si spieghi a cosa fanno riferimento le direttive seguenti e se ne fornisca la corretta interpretazione:

```
options {  
    directory "/var/named";  
};  
zone "." {  
    type hint;  
    file "root.hints";  
};  
zone "59.204.193.in-addr.arpa" {  
    type master;  
    file "pz/193.204.59";  
};
```

### RISOLUZIONE

Le direttive descritte fanno riferimento al server DNS Bind. Esse sono contenute nel file “named.conf” presente nella directory “/etc” o “/etc/named” su sistemi Linux.

La sezione “options” permette di definire delle direttive generali del server. In particolare, viene definita la directory di lavoro con la direttiva “directory”. Quindi, tutte le path specificate nelle direttive successive, a meno che non siano path assolute, fanno riferimento a questa directory.

Le due sezioni successive sono due zone. La prima zona è la zona root (“.”). La direttiva type di tipo hint indica che il server DNS per la zona root può fungere solo da server DNS di caching. Nel caso in cui non possieda una particolare risoluzione per la zona root, chiede di eseguire la query ad uno dei root server, i cui indirizzi IP sono salvati nel file di configurazione “root.hints” presente nella directory “/var/named/root.hints”.

La seconda zona è una zona per la risoluzione inversa degli indirizzi di tipologia “192.204.49”. Il server risulta un server autoritativo per la zona “59.204.193.in-addr.arpa”. Nel file “/var/named/pz/193.204.59”, quindi, saranno presenti dei Resource Record PTR che permetteranno la risoluzione inversa di tutti gli indirizzi IP del tipo 192.204.49.x

### COMMENTO

La domanda è chiara. E' bene specificare tutte le direttive con chiarezza. E' possibile incappare anche nel tipo di server “forward”, con la direttiva “forwarders {1.1.1.1; 1.1.1.1;}”;

### Quesito n. 3 appello del 3 novembre 2011

Si spieghi brevemente il significato dei seguenti comandi chiarendo in via preliminare a quale protocollo essi fanno riferimento e le possibili repliche (con status code) da parte server:

- HELO <name>
- RCPT TO:<name>
- RSET
- VRFY<recipient>
- NOOP

### RISOLUZIONE

I comandi illustrati sono comandi che si riferiscono al protocollo SMTP. Il significato è il seguente:

- HELO <name>  
Permette al client di farsi identificare dal server. Al posto di “<name>” si specifica il nome di dominio del client o l'Ip del client scritto tra parentesi quadre. Esempio di risposta server:

**250-smtp.esempio.it**

dove smtp.esempio.it è il nome di dominio del server.

- RCPT TO:<name>  
Permette di specificare a chi è indirizzato il messaggio di posta elettronica. Al posto di “name” mantenendo il maggiore e minore si specifica l'indirizzo email del destinatario. Esempio risposta:

**250 RCPT TO:<name> OK**

- RSET  
Permette di annullare le informazioni inserite fino al momento attuale. Ogni informazione inserita con altri comandi viene annullata. Esempio risposta:

**250 OK**

- VRFY <recipient>  
Verifica che l'email specificata esiste sul server. Esempio risposta:

**250 Michel Ruta m.ruta@poliba.it**

- NOOP  
Forza il server a rispondere con uno status code 250. Server per verificare se il server è ancora in ascolto. Esempio risposta server:

**250 OK**

#### Quesito n. 4 appello del 3 novembre 2011

Si osservi la seguente GET http spiegando in dettaglio di cosa si tratta:

```
http://some.tracker.com:987/announce?  
info_hash=17562354223434568990&peer_id=AEFGHIMNOPJKGAGSIDTARST&ip=193.204.59.227  
&port=7552&downloaded=2341&left=765&event=stopped
```

#### RISOLUZIONE

L'URL proposto rappresenta una richiesta HTTP GET effettuata da parte di un client che vuole comunicare con un Tracker BitTorrent, ovvero si tratta di un Tracker Announce. In questa URL la parte iniziale "http://some.tracker.com:987/announce" rappresenta l'URL del Tracker al quale si sta inviando la request http. Gli altri sono parametri il cui significato è il seguente:

**info\_hash** → è una stringa di 20 byte ottenuta dalla codifica SHA1 del campo "info" presente nel file .torrent a cui questa richiesta fa riferimento

**peer\_id** → stringa di 20 byte. rappresenta l'id del peer che si sta collegando con il Tracker

**ip** → è l'indirizzo ip del client

**port** → è la porta utilizzata dal client per comunicare

**downloaded** → indica i byte già scaricati della risorsa

**left** → indica i byte mancanti della risorsa

**event** → specifica lo status del peer. In questo caso, indica al Tracker che ha interrotto il download.

#### COMMENTO

Oltre ai parametri specificati nella risoluzione, è possibile passare al Tracker anche il campo "Uploaded" che indica i byte inviati in upload. Inoltre i due altri status che il client può specificare nel campo "event" sono "started" per indicare che il download è in corso e "completed" per indicare che è terminato. A questa GET, il Tracker risponde con un documento di tipo MIME "text/plain" contenente un dizionario bencoded con le seguenti chiavi:

**failure reason** = indica un problema nella condivisione della risorsa e specifica di che tipo di problema si tratta

**interval** → indica il tempo in secondi che il client deve attendere prima di inviare una nuova richiesta

**tracker\_id** → indica l'ID del tracker

**complete** → numero di peer con il file completo (seeders)

**incomplete** → numero di peers con il file incompleto (leechers)

**peers** → lista contenente tanti elementi quanti sono i peer. Ogni elemento della lista è un dizionario avente le chiavi "peer\_id", "ip" e "port"

## Quesito n. 1 appello del 6 febbraio 2012

Decifrare i seguenti messaggi chiarendo in via preliminare di cosa si tratta:

```
GET /wiki/Pagina_principale HTTP/1.1
Connection: Keep-Alive
User-Agent: Mozilla/10.0 (compatible; Konqueror/6.2; Linux) (KHTML, like Gecko)
Accept: text/html, image/jpeg, image/png, text/*, image/*, */*
Accept-Encoding: x-gzip, x-deflate, gzip, deflate, identity
Accept-Charset: iso-8859-1, utf-8;q=0.5, *;q=0.5
Accept-Language: en
Host: it.wikipedia.org
```

```
HTTP/1.0 200 OK
Date: Mon, 06 Feb 2012 16:03:31 GMT
Server: Apache/1.3.29 (Unix) PHP/4.3.4
X-Powered-By: PHP/4.3.4
Vary: Accept-Encoding, Cookie
Cache-Control: private, s-max-age=0, max-age=0, must-revalidate
Content-Language: it
Content-Type: text/html; charset=utf-8
Age: 7673
X-Cache: HIT from wikipedia.org
Connection: close
```

## RISOLUZIONE

I messaggi proposti sono un request ed una response HTTP inviate rispettivamente da un client ed un server. Il client sta richiedendo di accedere ad una delle pagine web del sito “Wikipwdia” e il server ha accettato questa richiesta. Analizziamo nel dettaglio la richiesta HTTP:

La prima riga è la “Request-Line”, la quale specifica il metodo (GET) l'URL della risorsa (“/wiki/Pagina\_principale”) e la versione del protocollo (HTTP/1.1).

La seconda riga indica che il client vuole utilizzare una connessione di tipo persistente, infatti è specificato il token “Keep-Alive” con il quale indica al server di mantenere la connessione attiva per i messaggi successivi.

La terza riga specifica il browser utilizzato dal client (Konqueror/6.2), il sistema operativo da cui è pervenuta la richiesta (Linux) e alcune info aggiuntive sul browser (Gecko è il layout engine).

La quarta riga è un'intestazione di tipo accept. In questo caso, viene specificato che il client accetta risorse con MIME type text/html, image/jpeg, image/png. In mancanza del tipo “text/html”, è disposto ad accettare una risorsa di tipo text di qualsiasi formato (text/\*). Allo stesso modo, in mancanza sia di immagini di tipo gif che di tipo jpeg, è disposto ad accettare immagini di qualsiasi formato (image/\*). Solo in mancanza di risorse di tipo “text” e “image” è disposto ad accettare qualsiasi tipo di risorsa (\*/\*).

La riga cinque è un header accept. Viene specificato che il client può accettare le compressioni di tipo gzip (x-gzip e gzip), deflate (x-deflate e deflate) così come può anche accettare una risorsa non compressa (identity).

La sesta riga è un header accept che indica i charset accettati dal client. In particolare, il client preferisce il charset iso-8859-1 ma in mancanza di questo accetta anche il tipo utf-8 nonostante abbia una preferenza minore (q=0.5). Infine se non sono disponibili entrambi è disposto ad accettare qualsiasi tipo di codifica (\*).

La settima riga indica che il linguaggio umano accettato dal client è inglese generico (us).

L'ultima riga specifica l'host su cui è presente la risorsa richiesta e a cui la request è diretta.

Provvediamo ad analizzare la response del server:

La prima riga è la Response-Line, in cui vengono specificati la versione del protocollo (HTTP/1.0), lo status code (200) e la frase descrittiva dello stato (OK). Si evince che la request inviata dal server è stata accettata.

La seconda riga indica la data in cui il messaggio di response è stato inviato. In particolare, tale data è lunedì 6 febbraio 2012 alle ore 16:03 circa.

La terza riga specifica il server che ha generato la response, in questo caso Apache/1.3.29 su sistema Unix su cui è installato l'engine PHP.

La quarta riga è una intestazione non standard che specifica la tecnologia per le pagine dinamiche installata sul server. In questo caso PHP/4.3.4.

La quinta riga indica che la response inviata è stata recuperata dalla cache del server e che questa era disponibile in diverse versioni. Il criterio di selezione per la scelta della versione da inviare è stato il valore delle intestazioni "Accept-Encoding" e "Cookie" inviate dal client.

La sesta riga è una intestazione utilizzata per il controllo dei meccanismi di caching. Il parametro "private" indica che il contenuto deve essere memorizzato solo in cache private, "s-max-age" vale solo per le cache condivise e definisce il tempo massimo oltre il quale la risorsa è da considerarsi scaduta ed in aggiunta applica una politica di gestione pari a must-revalidate, "max-age" indica solo il tempo di scadenza, "must-revalidate" indica che i meccanismi di cache una volta che la risorsa è scaduta non devono mai fornirla ma subito rivalidarla presso il server di origine.

La settima riga indica il linguaggio umano del contenuto (italiano).

L'ottava riga indica il tipo di contenuto (text/html) e il tipo di charset (utf-8).

La nona riga indica da quanto tempo la risposta inviata è presente nella cache del server.

La decima riga presenta una intestazione non standard, "X-Cache". Ci sta specificando che la risposta è stata recuperata dalla cache interna del server di wikipedia (HIT from wikipedia.org).

L'ultima riga sta comunicando al client che il server vuole chiudere la connessione attuale. Le prossime richieste richiedono l'apertura di una nuova connessione.

## COMMENTO

La traccia è chiara. E' da sottolineare come ci siano delle incongruenze nella risposta del server, come il fatto che l'intestazione Cache-Control non dovrebbe essere usata nei server HTTP/1.0 e che l'intestazione "Content-Language" ci specifica il language tag "it" quando invece la richiesta accettava solo "en". Il server, quindi, avrebbe dovuto rispondere con status code 406 (Not acceptable).

## **Quesito n. 2 appello del 6 febbraio 2012**

Si confrontino in modo schematico i pregi e i difetti delle risoluzioni DNS di tipo iterativo e ricorsivo e si chiarisca quando l'una risulta più efficace dell'altra non prima di avere descritto le caratteristiche fondamentali di ciascuna delle due.

### **RISOLUZIONE**

La risoluzione DNS prevede che un DNS resolver prenda in carico da un client una query DNS. Tale resolver provvederà a contattare altri server DNS, seguendo le gerarchie dell'albero dello spazio dei nomi, per rintracciare il server DNS autoritativo per la zona di riferimento del dominio che si vuole risolvere. Ciò che cambia tra le due risoluzioni è la modalità con cui il DNS resolver interagisce con gli altri server DNS.

Nella risoluzione di tipo iterativo, il resolver contatta direttamente ogni singolo server DNS che fa riferimento ad una certa zona indipendentemente dalla posizione gerarchica. Questo è possibile poiché ogni server DNS restituisce al resolver l'indirizzo del server autoritativo per la zona successiva.

Nella risoluzione di tipo ricorsivo, invece, c'è un vero e proprio “passaggio di consegne”, in cui ogni server DNS chiede al server autoritativo di una zona di contattare lui stesso il server autoritativo della zona successiva. Una volta raggiunto il server autoritativo per la zona del dominio che si vuole risolvere, questo invierà il messaggio di risposta ripercorrendo al contrario tutta la catena dei DNS server coinvolti nella risoluzione.

Per quanto riguarda la risoluzione iterativa, essa risulta essere svantaggiosa per quanto riguarda il DNS resolver, il quale si trova a dover occupare molta banda in quanto è lui l'unico adibito a contattare i vari server autoritativi. Inoltre è lui che si occupa di gestire richieste di contatto e elaborazioni delle risposte, pertanto c'è anche un consumo computazionale importante. Il vantaggio principale è rappresentato dai tempi di completamento della risoluzione che possono in generale essere inferiori a quelli di una risoluzione ricorsiva per il fatto che la risposta non deve percorrere la strada “a ritroso” attraverso tutti i server DNS che sono stati coinvolti nella risoluzione.

Per la risoluzione ricorsiva si hanno dei risultati speculari rispetto alla risoluzione iterativa, ovvero il carico computazionale e la banda sono distribuiti tra tutti i server DNS coinvolti nella risoluzione. Di contro, il tempo impiegato per completare la risoluzione può essere maggiore di quello richiesto nella risoluzione iterativa.

A conti fatti, è preferibile scegliere una risoluzione di tipo iterativo quando si ha a disposizione un DNS resolver dotato di buone prestazioni e molta banda ed in cui il tempo di delay della risposta è determinante. E' preferibile, invece, scegliere una risoluzione di tipo ricorsivo quando il DNS resolver è poco potente e il tempo di delay della risposta non è determinante.

Quindi la risoluzione di tipo iterativo è più efficace della ricorsiva quando il DNS resolver ha molta banda e potenza computazionale, mentre la ricorsiva è più efficace dell'iterativa quando il DNS resolver pecca in termini di banda e di carico computazionale.

### **COMMENTO**

La domanda chiede di descrivere brevemente come sono fatti i due tipi di ricorsione, i loro svantaggi e vantaggi e quando una è migliore dell'altra (risposto nelle ultime righe).



### **Quesito n. 3 appello del 6 febbraio 2012**

Chiarire la differenza esistente tra i protocolli IMAP e POP3 sulla base dei seguenti termini di confronto:

- Accesso alle caselle di posta
- Numero di accessi contemporanei
- Accesso a porzioni MIME di un messaggio e anteprima
- Gestione degli attributi dei messaggi su server
- Ricerca su mail server

### **RISOLUZIONE**

- Accesso alle caselle di posta  
Con POP3, l'accesso alla posta prevede che il client si colleghi al server, scarichi in locale i contenuti e poi si disconnetta. Con IMAP è possibile restare collegati con il server ed effettuare operazioni di elaborazione direttamente online
- Numeri di accessi contemporanei  
Con POP3, solo un utente alla volta può collegarsi ad una determinata casella di posta. In IMAP, più utenti contemporaneamente possono accedere alla stessa casella ed il server si occupa di gestire le richieste di modifica effettuate da tutti gli utenti
- Accesso a porzioni MIME di un messaggio e anteprima  
Nel POP3, il contenuto del messaggio, trasmesso formato MIME, deve essere scaricato nella sua interezza e non è prevista alcuna possibilità di poter avere un'anteprima del messaggio. Con IMAP, è possibile accedere anche ad una parte del messaggio MIME e, quindi, fornire un'anteprima del suo contenuto
- Gestione degli attributi dei messaggi su server  
In IMAP vengono introdotti degli attributi che permettono ad ogni client di tenere traccia delle operazioni effettuate da altri client. Per esempio, è possibile sapere se un messaggio è stato già letto o se ha avuto una risposta. In POP3 non esiste questa funzionalità.
- Ricerca su mail server  
Il protocollo IMAP permette al client di effettuare delle ricerche tra i messaggi attraverso dei criteri. In questo modo possono essere scaricati solo i messaggi di interesse. In POP3 questa funzionalità non è prevista.

### **COMMENTO**

La risposta a questa domanda è reperibile su Wikipedia al link  
[http://it.wikipedia.org/wiki/Internet\\_Message\\_Access\\_Protocol](http://it.wikipedia.org/wiki/Internet_Message_Access_Protocol)

#### **Quesito n. 4 appello del 6 febbraio 2012**

Supponendo di voler perpetrare un attacco di tipo Dos (Denial of Service) ai danni di un server FTP, si descrivano le modalità e le caratteristiche perchè l'attacco stesso risulti il più efficace possibile riuscendo a by-passare (almeno in fase preliminare) i controlli derivanti dal superamento della banda di guardia.

#### **RISOLUZIONE**

Gli attacchi di tipo Dos puntano a portare un servizio al limite delle sue prestazioni in modo tale che esso non possa più essere erogato. I due tipi di protezione principalmente presenti per evitare che un server sia soggetto a questo tipo di attacco sono il Bandwidth control (controllo della banda di guardia) e il filtro Anti-Spoofing.

Il primo provvede a monitorare la banda suddividendola in una “banda di traffico”, solitamente coincidente con i  $\frac{3}{4}$  della banda totale, ed una “banda di guardia”, corrispondente all'ultimo quarto. Una volta che il traffico supera la  $\frac{3}{4}$  ed inizia ad utilizzare la banda di guardia, il sistema suddivide la banda di traffico in altre sottobande, ancora una volta divise in un rapporto  $\frac{3}{4}$  e  $\frac{1}{4}$ . Procedendo con questa logica, è possibile individuare la provenienza del traffico malevolo e scartarlo.

Il secondo tipo di protezione viene implementato nei firewall o nei router di accesso alla rete del server che sta erogando il servizio. Tale filtro controlla la correttezza dei pacchetti arrivati e se questi sono alterati (spoofed) vengono ignorati.

Per perpetrare un attacco ad un server FTP di tipo Dos è in primo luogo necessario bypassare il controllo di guardia. Per rendere difficoltosa l'individuazione dell'indirizzo IP che genera traffico malevolo, il primo passo è generare un attacco proveniente da IP multipli diversificati, ad esempio utilizzando una rete di computer zombie.

Il secondo passo è “bucare” il filtro anti-spoofing per evitare che gli eventuali pacchetti giunti al server vengano identificati. Per fare questo, posso applicare un attacco di tipo “Blind Spoofing”. Questo è un attacco effettuato a livello di protocollo TCP. Avviene nella fase “three-way handshake”. Una macchina malevola invia un flag SYN con IP falsificato al server, il quale invia una risposta ACK+SYN all'IP fasullo specificato dall'attaccante. A questo punto il client malevolo invia subito l'ACK previsto dal three-way handshake sempre specificando l'IP falsificato ed utilizzando tale IP per tutti i pacchetti successivi. In questo modo, il server crede di stare dialogando con un certo IP quando in realtà è una macchina totalmente diversa. Affinchè tale attacco vada a buon fine, però, l'attaccante deve sapere in che modo il server genera i “sequence number” delle sue comunicazioni.

#### **COMMENTO**

Questa stessa domanda è presente in altri appelli con la differenza che l'attacco è da perpetrare nei confronti di un server HTTP invece che FTP. Nonostante questa differenza le modalità di attacco non cambiano.

## **Quesito n. 2 appello del 9 maggio 2012**

Si spieghi in modo conciso ma compendioso il funzionamento della risoluzione inversa DNS. A partire da quale livello la risoluzione torna ad essere operata localmente?

### **RISOLUZIONE**

La risoluzione inversa è un meccanismo il cui obiettivo è di recuperare il nome di dominio associato ad un certo indirizzo IP.

Inizialmente un client contatta un DNS resolver inviando una query inversa. Una volta giunta al resolver, tale query inversa viene rielaborata in una Pointer Query questo perchè con l'RFC 3425 il protocollo legato alla query inversa viene dichiarato obsoleto e viene ritenuto accettabile per la risoluzione inversa la mappatura eseguita sfruttando il dominio "in-addr.arpa". Una volta giunta al DNS resolver, quindi, la query inversa diviene una query standard di tipo PTR.

Il resolver si occupa di rielaborare l'indirizzo IP fornito dal client invertendo i byte che lo compongono e aggiungendoci l'indirizzo delle risoluzioni inverse. Se ad esempio l'IP fornito dal client è "192.168.0.1" il DNS resolver effettuerà una Pointer Query sul nome di dominio "1.0.168.192.in-addr.arpa".

Questa query PTR segue la logica delle normali risoluzioni standard, siano esse ricorsive o iterative, e una volta che questa raggiunge il server DNS autoritativo per la zona "0.168.192.in-addr.arpa" questo server fornisce il resource record di tipo PTR corrispondente al dominio da risolvere al DNS resolver.

A questo punto la risoluzione inversa torna ad essere eseguita localmente (cioè sul server DNS locale che funge da resolver) che rielabora il record PTR e fornisce al client una risposta di tipo inverso così come richiesta.

### **COMMENTO**

E' importante sottolineare la differenza tra POINTER QUERY (o query standard di tipo PTR) e query inversa (o IQUERY). Esse sono due cose diverse, che seguono due protocolli diversi. Quando la traccia ci chiede la risoluzione inversa, essa viene trasformata e gestita dal resolver locale (dove per resolver locale si intende il server DNS al quale chiediamo di risolvere la nostra query) come fosse una query PTR. Pertanto, la query inversa TORNA ad essere operata localmente una volta che il resolver ha ricevuto il messaggio di risposta della POINTER QUERY.

### Quesito n. 3 appello del 9 maggio 2012

Si spieghi a cosa fanno riferimento e cosa significano le direttive seguenti:

```
Listen *80
User michele
Group sisinflab_poliba
ServerAdmin admin@poliba.it
ServerName root.poliba.it
DocumentRoot "/var/www/html"
ErrorLog "/var/log/httpd/error_log"
CustomLog logs/custom_log
```

### RISOLUZIONE

Le direttive proposte sono contenute nel file testuale “httpd.conf” presente nella directory “/etc” su sistemi Linux. E' il file di configurazione del server http “Apache”. Il significato delle direttive è il seguente:

La prima riga indica che il server http apache è in ascolto sulla porta TCP 80 ed accetta connessioni per tutti gli indirizzi IP (carattere “\*”). Se è specificato un indirizzo IP, il server resterà in ascolto solo su l'interfaccia di rete dotata di quell'indirizzo IP (su server che hanno più interfacce di rete). E' una direttiva obbligatoria.

La seconda riga specifica l'utente con cui viene lanciato il demone del server. A seconda dell'utente e dei permessi che questo possiede sul sistema, il server Apache può acceder o meno a determinati file del sistema. In questo caso l'utente è “michele”.

La terza riga indica il gruppo di appartenenza dell'utente specificato in user. In questo caso “sisinflab\_poliba”.

La quarta riga definisce l'indirizzo e-mail dell'amministratore del server. E' bene specificarlo in modo tale che un utente possa contattare l'amministratore in caso di guasti

La quinta riga indica il nome di dominio per il quale questo server accetta richieste. E' possibile, infatti, che più nomi di dominio siano associati alla macchina su cui è avviato questo server. Se arriva una request la cui intestazione “Host:” non coincide con “root.poliba.it”, il server non la accetta.

La sesta riga indica la directory in cui devono essere contenute le risorse (pagine html, immagini etc). E' sostanzialmente la home directory.

La settima riga indica in che file devono essere salvati i messaggi di errore che il server generazione.

L'ultima riga specifica il file di log in cui devono essere memorizzate le request che arrivano al server. E' utilizzato per il tracking di tutto ciò che viene richiesto al server. In questo caso “logs/custom\_log”. Può essere specificata solo una path di tipo relativo, quindi in questo caso il path completo risulta “/usr/local/apache/logs/custom\_log” se il valore di ServerRoot non è stato modificato.

### COMMENTO

L'altra direttiva da ricordare è “DirectoryIndex” che permette di definire i nomi delle risorse che il server deve far riferimento quando arriva una richiesta con URL che termina con “/”.

### Quesito n. 3 appello del 9 maggio 2012

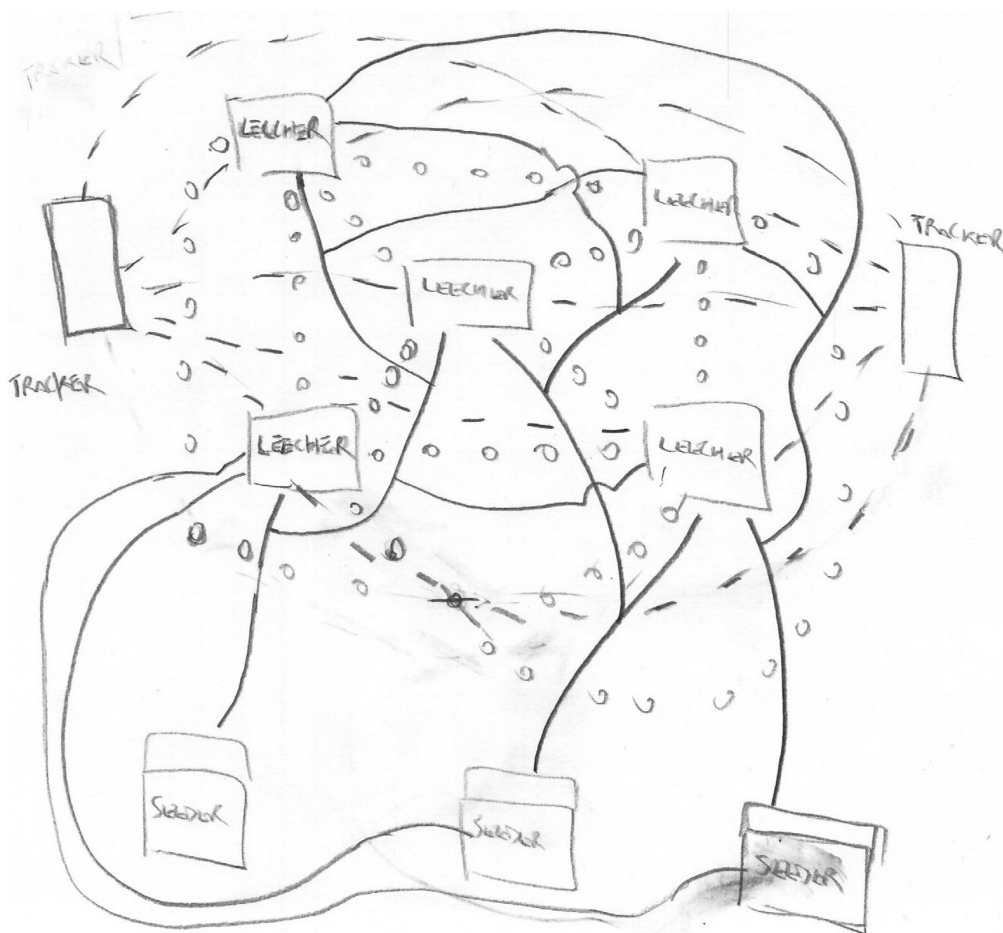
Si provveda a disegnare una rete peer to peer fatta da 10 nodi di cui:

- N. 2 tracker
- N. 3 seeder
- M. 5 leecher

Si faccia attenzione a segnare i collegamenti possibili tra gli attori coinvolti con simbologie diverse a seconda della tipologia. Si commenti lo schema realizzato.

#### RISOLUZIONE

Le linee a pallini indicano le connessioni stabilite tra leechers i quali si scambiano chunk offrendo dati sia in download che in upload. Le linee tratteggiate indicano la comunicazione tra leechers e trackers i quali ricevono informazioni da ogni peer sul loro stato attuale e forniscono a tutti informazioni sulla stato generale dei peers. Infine le linee continue rappresentano le connessioni dei leechers ai seeders i quali offrono la risorsa, ovvero c'è solo download da parte dei leechers.



#### COMMENTO

Spero riusciate a fare uno schema migliore del mio ...

### **Quesito n. 1 appello del 9 luglio 2012**

Si spieghi il significato dei seguenti parametri di una richiesta inoltrata ad un tracker BitTorrent:

info\_hash:

peer\_id:

port:

uploaded:

left:

event:

### **RISOLUZIONE**

**info\_hash:** E' una stringa di 20 byte ottenuta dalla codifica SHA1 del campo “info” contenuto nel file “.torrent” in cui sono contenute tutte le informazioni sulla risorsa che si sta scaricando

**peer\_id:** E' una stringa di 20 byte che indica l'ID del client che ha inoltrato la richiesta al Tracker

**port:** E' il numero di porta su cui il client che ha inoltrato la richiesta al Tracker è in ascolto

**uploaded:** Rappresenta il numero di byte trasferiti in upload da parte del peer che ha inoltrato la richiesta al Tracker

**left:** Rappresenta il numero di di byte rimasti da scaricare al peer che ha inoltrato la richiesta al Tracker

**event:** E' un campo che esplicita lo stato in cui si trova il peer. Se vale “stopped” significa che il peer è in pausa, se vale “started” significa che sta effettuando il download, se è “completed” significa che ha terminato di scaricare la risorsa

### **COMMENTO**

Per ulteriori chiarimenti guarda quesito n. 4 del 3 novembre 2011.

## **Quesito n. 2 appello del 9 luglio 2012**

Codificare secondo bencoding le seguenti sequenze chiarendo in via preliminare di cosa si tratta:

{ "Italia" ⇒ 12, "Spagna" ⇒ "n.p.", "Canada" ⇒ 8 }

[ "Italia2", "Austria", "Olanda", "18", "n.p.", 24 ]

### **RISOLUZIONE**

{ "Italia" ⇒ 12, "Spagna" ⇒ "n.p.", "Canada" ⇒ 8 }

La sequenza è un dizionario. La codifica segue la sintassi:

d<bencoding\_string><bencoding\_elemento>e

In questo caso risulta:

d6:Italiai12e6:Spagna4:n.p.6:Canadai8ee

[ "Italia2", "Austria", "Olanda", "18", "n.p.", 24 ]

La sequenza è una lista. La codifica segue la sintassi:

l<bencoding\_valori>e

In questo caso risulta:

l7:Italia27:Austria6:Olanda2:184:n.p.i24ee

### **COMMENTO**

E' sufficiente un po' di attenzione nell'applicazione delle regole sintattiche di codifica.

### **Quesito n. 3 appello del 25 luglio 2008**

Si traduca il seguente dizionario adoperando la codifica bencoding:

```
{"rossi" ⇒ "23",  
"bianchi" ⇒ "19",  
"verdi" ⇒ "n.a.",  
"rossi2" ⇒ "28",  
"rossi3" ⇒ ["24", "26"],  
"bianchi r." ⇒ "n.p."}
```

### **RISOLUZIONE**

La codifica bencoding di un dizionario segue la seguente sintassi:

`d<bencoded_string_key><bencoded_value>e`

in cui è specificato che ogni chiave deve essere codificata in bencoding come se fosse una stringa.  
La codifica è la seguente:

`d5:rossi2:237:bianchi2:195:verdi4:n.a.6:rossi22:286:rossi312:242:26e10:bianchi r.4:n.p.e`

### **COMMENTO**

Risoluzione immediata. Attenzione a come si codifica la chiave avente per valore una lista!!

`6:rossi312:242:26e`



### **Quesito n. 1 appello del 22 febbraio 2012**

Codificare secondo bencoding le seguenti sequenze chiarendo in via preliminare di cosa si tratta:

{ "Palermo" ⇒ 12, "Roma" ⇒ "n.p.", "L'Aquila" ⇒ 8 }

[ "Palermo", "Roma", "L'Aquila", 12, "n.p.", 8 ]

#### **RISOLUZIONE**

{ "Palermo" ⇒ 12, "Roma" ⇒ "n.p.", "L'Aquila" ⇒ 8 }

La sequenza è un dizionario. La sintassi per la codifica bencoding di un dizionario è la seguente:

d<bencoding\_string\_key><bencoding\_value>e

La codifica risulta:

d7:Palermoi12e4:Roma4:n.p.8:L'Aquilai8ee

[ "Palermo", "Roma", "L'Aquila", 12, "n.p.", 8 ]

La sequenza è una lista. La sintassi di codifica bencoding per una lista è:

l<bencoding\_value>e

La codifica risulta:

l7:Palermo4:Roma8:L'Aquilai12e4:n.p.i8ee

#### **COMMENTO**

Tutto tranquillo, tutto semplice :D

### **Quesito n. 1 appello del 28 febbraio 2013**

Codificare secondo bencoding le seguenti sequenze chiarendo in via preliminare di cosa si tratta:

{ "Napoli"  $\Rightarrow$  36, "Aosta"  $\Rightarrow$  "n.p.", "Napoli"  $\Rightarrow$  8 }

[ "Palermo", "Roma", "L'Aquila", 12, "n.p.", 8 ]

[ "Siracusa", "Caltanissetta", "Teramo", 5487, "n.p.", 8 ]

{ "46"  $\Rightarrow$  36, 72  $\Rightarrow$  "n.p.", "Bari"  $\Rightarrow$  8 }

{ 46  $\Rightarrow$  36, "72"  $\Rightarrow$  "n.p.", "Napoli"  $\Rightarrow$  8 }

### **RISOLUZIONE**

{ "Napoli"  $\Rightarrow$  36, "Aosta"  $\Rightarrow$  "n.p.", "Napoli"  $\Rightarrow$  8 }

La sequenza è un dizionario! Difatti il dizionario prevede che ci siano COPPIE chiave-valore uguali! In questo caso si hanno due chiavi uguali (Napoli) ma valori diversi (36,8). La codifica risulta:

d6:Napolii36e5:Aosta4:n.p.6:Napolii8ee

[ "Palermo", "Roma", "L'Aquila", 12, "n.p.", 8 ]

La sequenza è una lista. La codifica è la seguente:

l7:Palermo4:Roma8:L'Aquilai12e4:n.p.i8ee

[ "Siracusa", "Caltanissetta", "Teramo", 5487, "n.p.", 8 ]

La sequenza è una lista. La codifica è la seguente:

l8:Siracusa13:Caltanissetta6:Teramoi5487e4:n.p.i8ee

{ "46"  $\Rightarrow$  36, 72  $\Rightarrow$  "n.p.", "Bari"  $\Rightarrow$  8 }

La sequenza è un dizionario con una chiave di tipo numerico. Bencoding prevede che tutte le chiavi siano codificate come stringhe. Si ha:

d2:46i36e2:724:n.p.4:Barii8ee

{ 46  $\Rightarrow$  36, "72"  $\Rightarrow$  "n.p.", "Napoli"  $\Rightarrow$  8 }

La sequenza è un dizionario con una chiave numerica. Anche in questo caso va codificata come stringa:

d2:46i36e2:724:n.p.6:Napolii8ee

## Quesito n. 2 appello del 28 febbraio 2013

Si descrivano i principali elementi del linguaggio HTML. Si provveda inoltre a evidenziare i costrutti base per la strutturazione di una risorsa.

### RISOLUZIONE

L'HTML (Hyper Text Markup Language) è un linguaggio a marcatori utilizzato per la creazione di pagine web. Permette di formattare le risorse (testo, immagini, applicazioni etc..) all'interno di una web page attraverso l'utilizzo di "keyword" denominate "TAG". Questi TAG vengono interpretati dall'engine interno dei browser web il quale mostra all'utente le risorse formattate.

Ogni pagina web comincia con un TAG "<html>" il quale dà un'indicazione sull'inizio del contenuto della pagina web. Ogni pagina web è suddivisa in due sezioni: la prima è dedicata alla definizione delle intestazioni, delineata dal tag "<head>", il cui obiettivo è definire il contenuto della pagina web stessa, la seconda rappresenta il contenuto vero e proprio della pagina ed è individuata dal tag "<body>". Un esempio di pagina web strutturata come appena descritto è il seguente:

```
<html>
  <head>
    -----INTESTAZIONI
  </head>
  <body>
    -----CORPO DELLA PAGINA WEB
  </body>
</html>
```

A queste intestazioni di base si affiancano altri costrutti che permettono di formattare le risorse e vanno inseriti all'interno del tag "<body>". I più utilizzati sono i seguenti:

<b>Text</b> → Mostra il testo "Text" in grassetto

<u>Text</u> → Mostra il testo "Text" in sottolineato

<i>Text</i> → Mostra il testo "Text" in corsivo

<center>Text</center> → Permette di centrare il testo "Text"

 → Mostra la risorsa immagine "image.jpg"

<a href="link">Text</a> → Mostra il testo "Text" come link ipertestuale che permette di accedere alla pagina "link"

<applet code="Applet"></applet> → Inserisce un applet Java

<ol></ol> → Permette di mostrare una lista ordinata. All'interno ogni elemento della lista va definito con il tag "<li>"

<ul></ul> → Permette di definire una lista non ordinata. All'interno ogni elemento va definito con il tag "<li>"

<li>Text</li> → Intestazione che definisce un elemento di una lista ordinata o non ordinata

COMMENTO: Tutto facile, tutto semplice!

#### **Quesito n. 4 appello del 28 Febbraio 2013**

Si descrivano le principali tipologie di attacchi a cui vanno soggette le reti P2P chiarendo per ciascuno di essi eventuali modalità di difesa.

#### **RISOLUZIONE**

- **Poisoning attacks:** Vengono forniti file con contenuti diversi da quelli descritti.  
DIFESA → Inserire la possibilità di effettuare un'anteprima del file nei software P2P
- **Pollution attacks:** Vengono inseriti chunk “cattivi” in un file valido.  
DIFESA → Utilizzo dei commenti
- **Defection Attacks:** Utenti o software utilizzano la rete P2P senza contribuire alla condivisione delle risorse  
DIFESA → Utilizzo di un sistema di crediti per penalizzare gli utenti che non sono partecipativi
- **Inserimento di virus:** I file scaricati contengono dei virus  
DIFESA → Utilizzo dei commenti
- **Malware inserito nel software di rete stesso:** Il software per la gestione della rete P2P contiene spyware (vedi caso Kazaa)  
DIFESA → Informarsi in rete sul software che si vuole utilizzare
- **Denial of Service attacks:** Attacchi che rendono la rete satura facendola apparire lenta o fuori servizio
- **Filtering:** Gli operatori di rete (ISP) possono impedire il transito di pacchetti della rete P2P imponendo restrizioni su alcune porte  
DIFESA → Utilizzare porte diverse da quelle standard
- **Identity Attacks:** Identificare gli utenti della rete e perseguirli legalmente approfittando del passaggio in chiaro degli IP  
DIFESA → Utilizzo di un server proxy anonimo
- **Spamming:** Invio di informazioni non richieste agli utenti attraverso la rete  
DIFESA → Limitare le possibilità di contatto tra utenti

#### **COMMENTO**

Le risposte sono immediate. Esercizio mnemonico: per ricordarti la differenza fra attacchi “pollution” e attacchi “poisoning” associa il loro significato italiano. “Pollution” significa “inquinato”, quindi l'inquinamento riguarda qualcosa che nella sua totalità è buono ma che ha alcune componenti non buone (chunk corrotti), mentre “poisoning” è avvelenato quindi è qualcosa che non è buono nella sua interezza.

### **Quesito n. 3 appello del 22 febbraio 2012**

Chiarire la differenza esistente tra le seguenti coppie di header di una richiesta HTTP:

- Range, If-Range:
- Accept-Charset, Accept-Encoding
- Date, If-Modified-Since
- WWW-Authenticate, Proxy-Authorization

### **RISOLUZIONE**

- Range, If-Range  
L'intestazione Range è una intestazione presente solo nelle request. Comunica al server che il client vuole ricevere solo una parte della risorsa oggetto della request, solitamente un intervallo di byte.  
L'intestazione If-Range è una intestazione condizionale che può essere utilizzata solo in presenza dell'header "Range". La condizione indica al server che SE la risorsa che sto chiedendo non è cambiata, allora può inviare il contenuto parziale, altrimenti deve inviare la risposta per intero. E' utile per evitare che il client riceva parti parziali provenienti da versioni diverse della stessa risorsa. Il valore da specificare è un "Etag" o una data nel formato HTTP-date.
- Accept-Charset, Accept-Encoding  
Sono due intestazioni condizionali. La prima comunica al server che il client accetta risorse il cui charset, ovvero il set di codifica dei caratteri, è quello specificato (ad esempio utf-8). Accept-Encoding, invece, specifica al server che il client accetta contenuti codificati con l'algoritmo di compressione specificato oppure, se indicato il valore "identity", che è accettato solo un contenuto non codificato.
- Date, If-Modified-Since  
L'header "Date" specifica la data e l'ora in cui un certo messaggio HTTP è stato inviato. Il-Modified-Since, invece, è una intestazione condizionale utilizzata a scopi di validazione di una risorsa presente in cache. La condizione comunica al server che se la risorsa richiesta ha una data di ultima modifica successiva a quella passata come valore di questa intestazione, allora la risorsa deve essere inviata. In caso contrario il server invia il codice 304 (Not Modified).  
Quindi Date specifica la data del messaggio inviato, mentre l'header condizionale specifica una la data a cui risale la risorsa in cache.
- WWW-Authenticate, Proxy-Authorization  
Queste due intestazioni sono entrambe utilizzate per implementare gli algoritmi di autenticazione. Il primo è utilizzato nelle response di un server per comunicare al client che, per accedere alla risorsa richiesta, deve autenticarsi.  
Il secondo si riferisce all'autenticazione presso un server Proxy. In particolare, quando un server proxy invia un messaggio ad un client con codice 407 (Proxy authentication required), il client genera un messaggio che include questa intestazione specificando le credenziali di accesso.

### **COMMENTO**

Tutte le informazioni possono essere reperite nell'RFC 2616.

### Quesito n. 3 appello del 9 luglio 2012

Si associ a ciascuna componente del campo FLAGS del seguente pacchetto DNS il rispettivo significato e se ne indichi la codifica compatta:

1... .. =

.000 0... .. =

.... .1... .. =

.... ..1. .... =

.... ...1 .... =

.... .... 1... .. =

.... .... .0.. .... =

.... .... ..0. .... =

.... .... .... 0001 =

### RISOLUZIONE

1... .. = QR(query/response) Indica che il messaggio è una risposta

.000 0... .. = OPCODE(operation code) Indica che il messaggio è una query standard

.... .1... .. = AA(Authoritative answer) indica che la risposta proviene da un server autoritativo

.... ..1. .... = TC(truncate) indica che la risposta è stata troncata in più messaggi

.... ...1 .... = RD(recursion desired) indica che è stato chiesto di risolvere la query in modo ricorsivo

.... .... 1... .. = RA(recursion available) indica che il server è in grado di risolvere query ricorsive

.... .... .0.. .... = Z bit di padding posto a zero

..... ..0. .... = AD(authenticated data) indica che la risposta contiene dati non autenticati

..... .. 0001 = RCODE(response code) Indica codice di risposta 1, ovvero indica che c'è stato un errore nel formato della richiesta della query (Format error)

La forma compatta risulta:

BINARIO	1000	0111	1000	0001
ESADECIMALE	8	7	8	1

FORMA COMPATTA 0x8781

COMMENTO

Per approfondimenti vedi il quesito n. 2 del 14 novembre 2013.

#### **Quesito n. 4 appello del 9 luglio 2012**

Si spieghino le principali differenze esistenti tra la modalità attiva e quella passiva in FTP e chiarire in quali casi è opportuno adoperare l'uno in luogo dell'altra.

#### **RISOLUZIONE**

Il protocollo FTP prevede due tipi di connessioni: una connessione di controllo attraverso la quale un client può inviare dei comandi al server ed una connessione dati attraverso la quale avviene il trasferimento dei file vero e proprio. Quest'ultimo tipo di connessione può realizzarsi in due modalità, dette “Attiva” e “Passiva”.

La modalità attiva, prevede che sia il server FTP a creare la connessione dati ed a collegarsi al client. Un client FTP ha la possibilità di inviare, tramite connessione di controllo, il comando “PORT” attraverso il quale specifica al server il suo indirizzo IP e una porta TCP, il cui numero è scelto casualmente e deve essere maggiore di 1024, su cui il server dovrà far pervenire la richiesta di connessione. Il server, ricevuto queste informazioni, utilizza la porta standard per la connessione dati, ovvero la porta 20, e instaura la connessione con il client.

La modalità passiva, invece, prevede che sia il client ad instaurare la connessione dati con il server. Attraverso il comando PASV, il client richiede al server di instaurare una comunicazione in questa modalità. Il server risponde con il proprio indirizzo IP ed una porta TCP random, maggiore di 1024, alla quale il client dovrà inoltrare la richiesta di connessione. Il client, ricevute queste informazioni, apre una porta TCP random maggiore di 1024 e instaura la connessione sulla porta comunicata in precedenza dal server.

Il problema legato alla scelta tra queste modalità ricade su questioni legate alla sicurezza. Difatti i sistemi informatici utilizzano dei firewall il cui compito è bloccare o accettare il traffico pervenuto su determinate porte. Tale firewall può essere presente sia lato client che lato server.

La modalità attiva risulta essere la più sicura lato server, in quanto il firewall del server in questa modalità deve essere configurato per accettare traffico in uscita/entrata solo sulla porta standard 20. Ciò riduce il rischio di attacchi malevoli sulle altre porte del server. Questa modalità, però, non può essere sempre utilizzata in quanto c'è la possibilità che anche il client sia protetto da un firewall, il quale inibisce il traffico in entrata ed impedisce al server di collegarsi sulla porta specificata con il comando PORT dal client.

In modalità passiva è il client ad essere maggiormente protetto, poiché è lui a doversi connettere su una porta scelta dal server e, quindi, il firewall del client può essere configurato in maniera più sicura. In questo caso, però, il server è costretto a configurare il proprio firewall lasciando “aperte” un range di porte (in questa modalità il server si mette in ascolto su una porta random) sulle quali il client potrà connettersi, esponendolo maggiormente agli attacchi esterni.

In sostanza, è bene utilizzare la modalità passiva in tutti quei casi in cui si è costretti poiché in modalità attiva risulterebbe impossibile, mentre in tutti gli altri casi, ad esempio in situazioni in cui client e server fanno parte della stessa rete interna oppure di una VPN crittografata in cui non sono necessarie impostazioni di sicurezza molto restrittive, è preferibile usare la modalità attiva la quale garantisce maggiore sicurezza al server.



## Quesito n. 1 appello del 25 luglio 2008

Si provveda a spiegare il significato dell'output del comando [utente@host:~]\$dig poliba.it MX:

```
; <<>> DiG 9.3.1 <<>> poliba.it MX
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 14562
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 4, ADDITIONAL: 5
;; QUESTION SECTION:
;poliba.it. IN MX
;; ANSWER SECTION:
poliba.it. 172800 IN MX 10 mail.poliba.it.
poliba.it. 172800 IN MX 20 anthares.poliba.it.
;; AUTHORITY SECTION:
poliba.it. 604800 IN NS cstar.poliba.it.
poliba.it. 604800 IN NS server2.garr.net.
poliba.it. 604800 IN NS anthares.poliba.it.
poliba.it. 604800 IN NS dns2.nic.it.
;; ADDITIONAL SECTION:
mail.poliba.it. 43200 IN A 193.204.49.50
anthares.poliba.it. 172800 IN A 193.204.49.37
cstar.poliba.it. 43200 IN A 193.204.49.36
server2.garr.net. 73632 IN A 193.206.141.38
dns2.nic.it. 73355 IN A 193.205.245.8
;; Query time: 7 msec
;; SERVER: 193.204.49.36#53(193.204.49.36)
;; WHEN: Mon May 29 16:03:24 2006
;; MSG SIZE rcvd: 240
```

## RISOLUZIONE

Il messaggio mostrato è la risposta ottenuta dal comando “dig” il quale richiede di effettuare una query di tipo MX. Questo tipo di query permette di individuare il server/i server di posta a cui compete la gestione degli indirizzi e-mail facenti riferimento al dominio specificato.

Il risultato mostrato indica in primo luogo le intestazioni del messaggio DNS di risposta. OPCODE indica che la query era di tipo standard, STATUS(equivalente di RCODE) indica che non ci sono stati errori ed inoltre sono esttati i flag QR, AA,RD,RA che indicano che il messaggio è una risposta, che la risposta è stata data da un server autoritativo e che la query era stata richiesta in modalità ricorsiva ed è stata effettivamente così risolta. Viene poi mostrato che sono presenti 2 answer, 4 RR nella sezina authority e 5 RR additional. Ognuno di questi RR specifica il proprio time-to-live, la propria classe (IN) ed il proprio tipo (MX, A, NS).

La sezione “answer” mostra il risultato della query, ovvero che sono due i server di posta elettronica adibiti, ovvero “mail.poliba.it” e “anthares.poliba.it”. Di questi, il primo ha un preference number più basso(10), pertanto sarà il primo ad essere contattato.

La sezione “authority” riporta i server DNS che risultano autoritativi per la zona “poliba.it” e che, nel loro database interno, possiedono il resource record MX contenente le risposte fornite nella sezione “answer”. In questo caso sono quattro.

Infine la sezione “additional” mostra dei record addizionali di tipo A che indicano gli indirizzi IP dei mailserver avuti in risposta e dei server DNS autoritativi che la risposta l'hanno prodotta.

Le ultime righe sono delle informazioni aggiuntive sul messaggio di risposta. In particolare, viene specificato il tempo totale della risoluzione (7 msec) il server DNS resolver (192.204.49.36) la data di invio del messaggio e la dimensione dello stesso.

## COMMENTI

Oltre a spiegare il contenuto del messaggio relativo alla parte attinente la risoluzione DNS, è bene spiegare anche il significato delle altre stringhe fornite in output da dig.

#### **Quesito n. 4 appello del 25 luglio 2008**

Spiegare brevemente l'origine, il significato e l'utilità della codifica MIME.

#### **RISOLUZIONE**

La codifica MIME (Multipurpose Internet Mail Exchange) è uno standard di comunicazione che prevede l'invio di dati binari codificati ai quali vengono affiancate delle intestazioni che descrivono e qualificano il contenuto informativo dei dati codificati inviati.

E' nato dall'esigenza pratica legata ad un limite del protocollo SMTP (Simple Mail Transfer Protocol) il quale è in grado di interpretare correttamente soltanto i primi 128 caratteri (quindi solo i primi 7 bit sono significativi) del codice ASCII. Da qui l'esigenza di trovare un modo per inviare sfruttando questo protocollo messaggi ASCII in modo corretto. Difatti le codifiche utilizzate sono codifiche a 7 bit (tra cui la base64 a 7 bit) che possono correttamente essere interpretate dall'SMTP. Per poter decodificare in modo corretto questi tipo di dati, sono state affiancate delle intestazioni che caratterizzano il contenuto dei dati e che definiscono i cosiddetti MIME type. Le due intestazioni principali MIME sono "Content-Type" e "Content-Transfer-Encoding".

Queste intestazioni MIME e questi MIME type, inizialmente pensati solo per lo scambio di e-mail, sono stati estesi ad altri ambiti ed in particolare sono ad oggi utilizzate per la comunicazione e lo scambio di risorse web in HTTP.

#### **COMMENTO**

Informazioni essenziali, né più né meno

## Quesito n. 2 appello del 22 febbraio 2012

Si descrivano i vantaggi dello schema di sviluppo AJAX rispetto al caso di applicazioni tradizionali. Si utilizzi una opportuna schematizzazione per chiarire ogni aspetto.

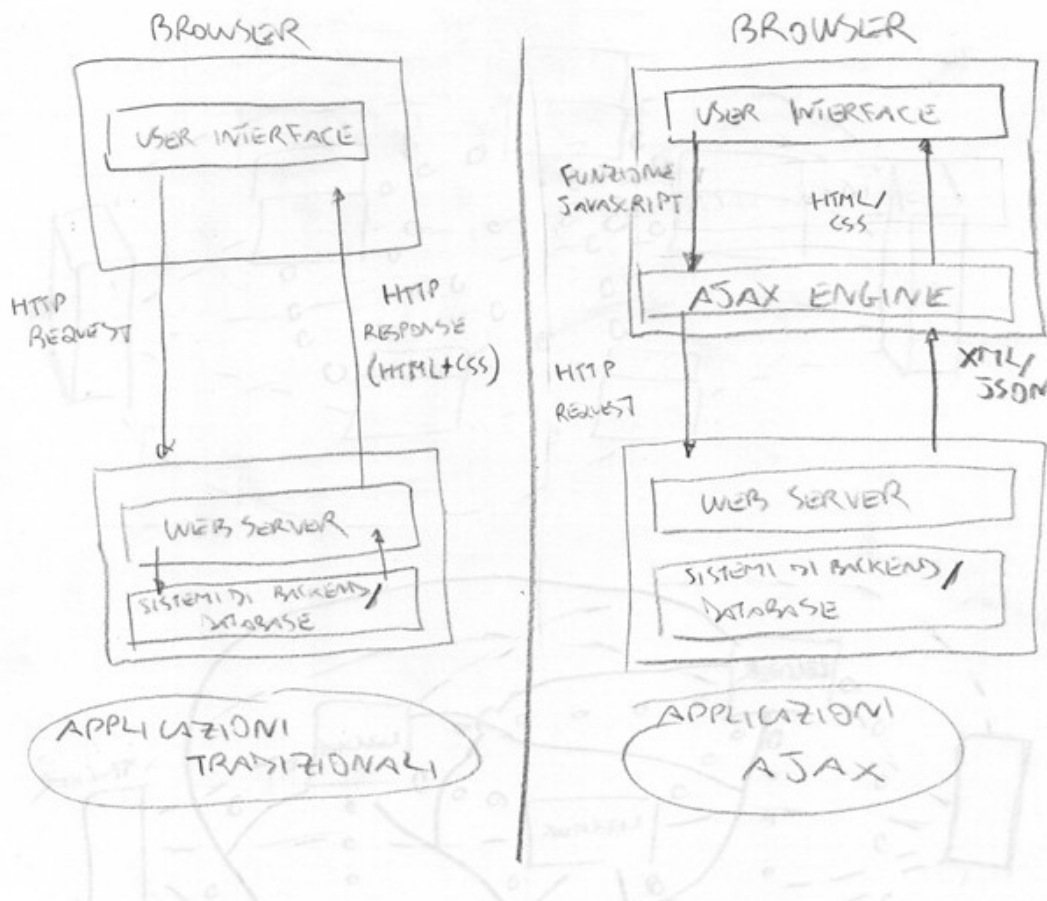
### RISOLUZIONE

AJAX (Asynchronous Javascript And XML) rappresenta un insieme di tecniche di programmazione per lo sviluppo di applicazioni web lato client.

Tale gruppo di tecniche basa il suo funzionamento sul concetto di “asincrono”, ovvero i dati tra client e server vengono scambiati in “background”, senza cioè interferire con i comportamenti e con ciò che è visualizzato su una pagina web esistente. In questo modo, i contenuti di una pagina possono essere dinamicamente aggiornati senza che l'utente sia esplicitamente costretto a ricaricare interamente la pagina. Il linguaggio lato client utilizzato per implementare le funzionalità legate ai meccanismi asincroni è il JavaScript, il quale permette l'accesso alla modifica degli oggetti HTML presenti in una pagina tramite DOM (Document Object Model). Ad esso si affiancano i formati l'XML o JSON (JavaScript Object Notation) per lo scambio dei dati tra client e server. L'oggetto che permette di implementare in JavaScript l'interscambio dati è “XMLHttpRequest”.

I vantaggi principali di AJAX, quindi, risiedono nella diversa user experience poichè l'utente può accedere alle informazioni di un sito web senza interruzioni e con interfacce user friendly, nel risparmio di banda dovuto al fatto che vengono inviate minori informazioni (le parti HTML non modificate non vengono reinviare), nella minore spesa elaborativa da parte di client e server, nella facilità con cui è incapsulabile nel codice HTML.

Gli svantaggi legati all'utilizzo di AJAX sono da un lato legate a come le applicazioni sono sviluppate, ad esempio se la mole di dati JSON o XML è importante, i tempi di attesa per lo scambio di dati possono essere lunghi o il browser può risultare rallentato. Inoltre ci possono essere problematiche legate all'utilizzo di alcune funzionalità dei browser come i tasti back/forward o al fatto che il JavaScript sia disabilitato. Il meccanismo implementato da AJAX può essere messo a confronto con quello tradizionale nella schematizzazione seguente:



### **Quesito n. 3 appello del 24 novembre 2011**

Decifrare il modo puntuale il seguente contenuto chiarendo in via preliminare di cosa si tratta:

```
;; AUTHORITY SECTION:
59.204.193.in-addr.arpa. 604800 IN NS cstar.poliba.it.
59.204.193.in-addr.arpa. 604800 IN NS server2.garr.net.
59.204.193.in-addr.arpa. 604800 IN NS anthares.poliba.it.
59.204.193.in-addr.arpa. 604800 IN NS dns2.nic.it.
;; ADDITIONAL SECTION:
cstar.poliba.it. 43200 IN A 193.204.49.36
server2.garr.net. 73768 IN A 193.206.141.38
anthares.poliba.it. 172800 IN A 193.204.49.37
dns2.nic.it. 73491 IN A 193.205.245.8
;; Query time: 7 msec
;; SERVER: 193.204.49.36#53(193.204.49.36)
;; WHEN: Mon May 29 16:01:08 2006
;; MSG SIZE rcvd: 236
```

### **RISOLUZIONE**

Il contenuto mostrato è una parte del messaggio di risposta fornito dal comando “dig”. In particolare, sono presenti la sola sezione “AUTHORITY” e “ADDITIONAL” più altre informazioni aggiuntive. Il comando digitato dall'utente è del tipo:

[utente@host:~]\$ dig x.59.204.193.in-addr.arpa PTR

il cui scopo è effettuare la risoluzione inversa dell'indirizzo specificato.

Nella sezione “AUTHORITY” del contenuto mostrato sono presente i server DNS autoritativi per la zona “ 59.204.193.in-addr.arpa”. Questi sono resource record in cui sono definiti il TTL (time-to-live), la classe (IN) il tipo (NS) e il valore vero e proprio.

Nella sezione “ADDITIONAL” sono presenti dei resource record aggiuntivi che associano a tutti i server DNS presenti nella sezione “AUTHORITY” il loro indirizzo IP. Per ogni RR è definito il TTL, la classe (IN) il tipo (A) e il valore vero e proprio.

Le informazioni presenti nelle ultime quattro righe sono rispettivamente il tempo impiegato per risolvere la query (7 msec), il DNS resolver (193.204.49.36), la data in cui la query è stata effettuata (lunedì 29 maggio 2006 alle ore 16:01 circa).

## Quesito n. 1 appello del 24 novembre 2011

Spiegare con dovizia di particolari la modalità di implementazione dei meccanismi di cache control basati su Server-specified expiration. Si citino tutte le direttive e gli header coinvolti.

### RISOLUZIONE

La “Server-specified expiration” è un metodo di gestione dei meccanismi di caching in HTTP introdotto a partire da HTTP 1.1. Tale metodo presuppone che ad ogni risorsa memorizzata in cache debba essere assegnato in modo deterministico ed ESPLICITO dai server una “età” oltre la quale il contenuto è da ritenersi scaduto. Si contrappone alla “Heuristic expiration” la quale prevede che se una risorsa non ha una età definita esplicitamente dal server, questa deve essere assegnata in modo euristico, talvolta con un calcolo statistico ottimistico (si assegna un valore scaduto il quale la risorsa è in realtà scaduta) o pessimistico (si assegna un valore scaduto il quale in realtà la risorsa è ancora valida).

Le intestazioni che permettono l'implementazione di tale meccanismo sono “Expires” e “Cache-Control”. Il primo permette di specificare una data in formato HTTP-date oltre la quale la risorsa è da considerarsi scaduta e deve essere rivalidata. Con “Cache-Control”, invece, abbiamo a disposizione diversi parametri i quali vengono applicati a tutti i meccanismi di caching presenti (lato client, lato proxy, lato server). I parametri principali sono i seguenti:

**public** → indica che la risorsa deve essere salvata SEMPRE nelle cache condivise, anche quando vengono interscambiati messaggi che dovrebbero essere salvati solo in cache private (esempio autenticazione a realm)

**private** → indica che la risposta messa in cache deve essere salvata solo in cache private (non condivise) e quindi accessibile solo ad alcuni utenti

**max-age** → è una direttiva che sovrascrive expires. Indica l'età in secondi della risorsa. Passato questo tempo, la risorsa deve essere considerata scaduta.

**max-stale** → indica che si accettano anche risposte scadute. Se è indicato un valore, indica quanti secondi dopo la scadenza si può restituire la risorsa scaduta. Se non specificato un valore, la risorsa scaduta può sempre essere restituita. Quando un server invia un response scaduto, deve aggiungere l'intestazione “Warning” con codice 110 (response is stale)

**must-revalidate** → è una direttiva che obbliga i meccanismi di caching a non fornire mai una risposta scaduta fino a quando questa non è stata rivalidata. Se non si riesce a contattare il server di origine per rivalidarla, è necessario rispondere con errore 504 (Gateway timeout)

**s-max-age** (o s-maxage) → unisce le funzioni di max-age e must-revalidate ma tali direttive devono essere recepite solo dalle cache condivise. Sovrascrive max-age ed expires.

**no-cache** → è una direttiva che indica che i contenuti non devono mai essere salvati in cache ma che devono essere sempre richiesti dall'origin server

**proxy-revalidate** → ha la stessa funzionalità di must-revalidate ma viene recepita solo dalle cache private (che richiedono il login e valgono, quindi, solo per determinati utenti)

## Quesito n. 2 appello del 24 novembre 2011

Si spieghi in modo conciso ma compendioso cosa è ISO/OSI, quali obiettivi si prefigge di raggiungere e qual è il rapporto della suite TCP/IP con esso.

### RISOLUZIONE

L'ISO/OSI (International Organization for Standardization / Open system Interconnection) è un modello standard il quale si pone l'obiettivo di fornire delle linee guida per la comunicazione tra host (computer, dispositivi di rete etc...) facenti parte di una rete informatica e, quindi, tra loro interconnessi. Fornisce, inoltre, dei criteri per confrontare due architetture di rete. Affinchè due host possano comunicare, quindi, le loro architetture devono essere conformi alle direttive fornite dall'ISO/OSI. Il modello non definisce, però, i protocolli e i servizi in modo specifico e dettagliato, ma delinea con il suo modello a layer (livelli) dei task che ognuno di questi livelli deve portare a termine. Ogni livello riceve dal livello inferiore delle informazioni, le elabora e le passa al livello successivo tramite dei servizi. Grazie a questa struttura, ogni layer deve occuparsi soltanto dei suoi task senza badare alle problematiche relative agli altri livelli.

Il TCP/IP (Transmission control protocol/ Internet protocol) è invece una suite di protocolli la quale è ad oggi utilizzata per la comunicazione su Internet. Prende il nome dai due protocolli che rappresentano maggiormente la suite, ovvero il TCP (protocollo livello trasporto) e IP (protocollo del network layer). Al contrario dell'ISO/OSI, quindi, definisce in modo specifico ogni protocollo e servizio dei layer che lo compongono.

Confrontando l'ISO/OSI con il TCP/IP è possibile identificare un numero diverso di layer in quanto alcuni dei layer dell'ISO/OSI vengono integrati in un unico layer del TCP. In particolare:

ISO/OSI		TCP/IP
Applicazione	→	Applicazione
Presentazione	→	Applicazione
Sessione	→	Applicazione
Trasporto	→	Trasporto
Network	→	Network (o Internet)
Data Link	→	Media Access
Livello Fisico	→	Media Access

#### Quesito n. 4 Appello del 24 novembre 2011

Si chiarisca il significato della interazione qui di seguito spiegando passo per passo quali sono le operazioni in essere, le componenti protocollari coinvolte e le funzionalità espletate.

```
1 0.000000 193.204.59.21 193.204.59.227 TCP 32786 > ftp(21) [SYN] Seq=0 Ack=0
2 0.000083 193.204.59.227 193.204.59.21 TCP ftp(21) > 32786 [SYN, ACK] Seq=0 Ack=1
3 0.000135 193.204.59.21 193.204.59.227 TCP 32786 > ftp [ACK] Seq=1 Ack=1
4 0.016440 193.204.59.227 193.204.59.21 FTP ftp > 32786 Response: 220 (vsFTPD 2.0.1)
```

#### RISOLUZIONE

Nelle iterazioni mostrate un client FTP sta instaurando una connessione con un server FTP attraverso il “three-way handshake”.

Al primo rigo, il client con indirizzo IP 193.204.59.21 utilizza la porta TCP 32786 per inviare al server FTP (porta 21) di indirizzo 193.204.59.227 un messaggio con intestazione TCP che vede il flag “SYN” posto a 1 e i campi “Seq” (sequence number) e “Ack” (**acknowledgment** number) posti a valore zero. Nel TWH, il valore di “Seq” viene generato casualmente dal client o comunque con logiche da lui decise (lo indichiamo in generale con X). In questo caso, il valore generato è zero.

Al secondo rigo, il server FTP risponde con un messaggio la cui intestazione TCP vede settati a 1 i flag “SYN” e “ACK”. Inoltre il “Seq” viene fissato a valore Y=zero, valore che viene generato casualmente dal server, e l’Ack” viene fissato al valore del sequence number arrivato dal client incrementato di uno (Ack=X+1 dove X è il sequence number inviato dal client). In questo caso, essendo zero il valore di seq inviato dal client zero, “Ack” risulta settato a uno.

Al terzo rigo, il client invia l'ultimo messaggio necessario al completamento del TWH. Invia un messaggio con intestazione TCP che vede settato a 1 il flag “ACK” e il campo “Ack” settato a Y+1=1, ovvero il valore del “Seq” inviato dal server (Y=zero) incrementato di uno. Inoltre risulta settato il “Seq” a valore uno. Questo ad indicare che il client da quel messaggio in poi considera conclusa la fase di handshake e, quindi, genera un nuovo valore di “Seq”.

Infine, la quarta riga mostra il primo messaggio del protocollo inviato dal server FTP. Si vede che viene restituito il codice 220 (codice di benvenuto) con la descrizione nome del server FTP che, in questo caso, è vsFTPD.

#### COMMENTO

E' un argomento in realtà studiato in Telematica I. E' bene ad ogni modo andarselo a rivedere!!



### Quesito n. 3 appello del 6 maggio 2008

Si spieghi ogni passo della seguente interazione precisando in via preliminare di cosa si tratta:

```
+OK dovecot ready.
USER user
+OK
PASS tlcruta2
+OK Logged in.
STAT
+OK 3 6304
LIST
+OK 3 messages:
1 4935
2 1248
3 121
.
RETR 1
+OK 4935 octets
Return-Path: <www-r@cs.cmu.com>
Received: from mail.cs.cmu.com (mail.cs.cmu.com [128.101.35.125]) by
deemail.poliba.it (Postfix) with ESMTP id 1DB2EEB3F4 for
<ruta@deemail.poliba.it>; Fri, 25 Apr 2008 13:15:03 +0100 (CEST)
.
QUIT
+OK Logging out.
```

### RISOLUZIONE

Il contenuto proposto rappresenta una sessione tra un client ed un server POP3. Il protocollo POP3 non prevede status code in risposta. Analizziamo nel dettaglio ogni passaggio:

La prima riga è il messaggio di benvenuto del server. In questo caso il server POP tre è “dovecot”.

La seconda riga è un comando dell'utente. Esso si sta identificando con il comando “USER” ed ha digitato il nome utente “user”.

La terza riga è una risposta affermativa del server.

La quarta riga mostra l'utente digitare il comando “PASS” ed inserire la password “tlcruta2”.

La quinta riga mostra il server rispondere in modo affermativo alle credenziali inserite ed indicare che adesso l'utente è loggato.

La sesta riga mostra l'utente inserire il comando STAT, il quale permette di mostrare una lista dei messaggi presenti nella mailbox dell'utente loggato.

La settima riga è il server che mostra all'utente i messaggi presenti indicando il loro numero identificativo e la loro lunghezza in byte. Sono presenti 3 messaggi. Il carattere “.” indica la fine del messaggio di risposta del client.

L'ottava riga mostra l'utente inserire il comando “RETR 1” con il quale vuole scaricare il messaggio di indice 1.

La nona riga indica la risposta affermativa del server e la lunghezza del messaggio.

Le righe dalla 10 alla 13 mostrano il contenuto dell'email. Vengono mostrate le intestazioni del messaggio che specificano il server SMTP che ha inviato il messaggio è l'indirizzo e-mail del mittente (ruta@deemail.poliba.it). Viene, inoltre, indicata la data dell'invio (Venerdì 25 aprile 2008 alle ore 15:13 circa). La riga 14 indica la fine del messaggio del server POP3 (carattere “.”).

La riga quindici mostra l'utente inserire il comando QUIT per chiudere la connessione con il server

L'ultima riga è il messaggio di chiusura connessione inviato dal server.

**Quesito n. 3 appello del 26 settembre 2007**

Si scriva la risposta ad una richiesta http avente avuto esito positivo, inviata da un Web server di tipo Microsoft IIS su una macchina Microsoft Windows 2003 Server, in data e ora correnti. Si precisa che la dimensione della risorsa in oggetto (una pagina HTML) è di 8.51 KB, il suo ultimo aggiornamento risale al 04 agosto 2007 alle 13:20 e il server ha impostato un timeout di 40s sulla connessione e un numero massimo di scambi pari a 80. Si tralasci il corpo della reply.

**RISOLUZIONE**

HTTP/1.1 200 OK

Server: Microsoft-IIS/6.0

Date: Wed, 26 Sep 2007 18:00:00 GMT

Last-Modified: Sat, 04 Aug 2007 13:20:00 GMT

Content-Type: text/html

Content-Length: 8715

Keep-Alive: timeout=40, max=80

Connection: Keep-Alive

**COMMENTO**

L'esercizio esplicita tutti i valori da inserire. La dimensione totale è stata ottenuta con  $8,51 \times 1024 = 8714,24 = 8715$ . I due elementi da tenere a mente sono il nome del sistema operativo (Microsoft Windows 2003) in quale non viene esplicitamente scritto nell'intestazione "Server" ma a seconda del sistema operativo la versione del "IIS cambia! In questo caso, Windows Server 2003 monta di default la versione 6.0. Le altre versioni si possono vedere alla pagina [http://en.wikipedia.org/wiki/Internet\\_Information\\_Services](http://en.wikipedia.org/wiki/Internet_Information_Services). Oltre a questo c'è un altro elemento di difficoltà: la data di ultima modifica. Viene esplicitato 4 agosto 2007 senza giorno della settimana...tenendo conto che la traccia è stata data il 26 settembre, come fare a ricordarselo???? .\_.

**Quesito n. 2 appello del 09 settembre 2011**

Si scriva la risposta ad una richiesta http avente avuto esito positivo, inviata da un Web Server di tipo Apache da una macchina Microsoft Windows ME, in data e ora correnti. Si precisa che la dimensione della risorsa in oggetto (una pagina HTML) è di 65.521 B, il suo ultimo aggiornamento risale al 29 settembre 2010 alle 08:30 e risultano impostati i vincoli di connessione seguenti: timeout:120s; numero massimo di iterazioni:80. Si tralasci il corpo del pacchetto.

**RISOLUZIONE**

HTTP/1.1 200 OK

Server: Apache/2.4.2 (Win32)

Date: Thu, 09 Sep 2011 08:00:00 GMT

Last-Modified: Wed, 29 Sep 2010 08:30:00 GMT

Content-Type: text/html; charset=iso-8859-1

Content-Length: 66

Keep-Alive: timeout=120, max=80

Connection: Keep-Alive

Ho assunto che il valore “65.521” sia con la virgola, quindi ho arrotondato al byte successivo ottenendo 66.

**RISOLUZIONE**

Anche qui la data del last modified è senza giorno della settimana...a parte questo, l'informazione del sistema operativo così dettagliata è irrilevante poiché apache definisce il sistema operativo sempre con un generico “Win32”. E' bene ricordarsi che il web server apache, inoltre, esplicita sempre nel content type anche il charset (valore standard iso-8859-1). Infine, la lunghezza dei dati è espressa in BYTE. Assumendo che quel punto sia una virgola (così come negli altri esercizi), arrotondo e scrivo 66 byte. Se si sceglie che quello sia un punto, il valore è 65521 e va scritto così come dato nella traccia. L'importante è inserire un commento per specificare la scelta fatta.

### **Quesito n.1 appello del 7 maggio 2010**

Si provveda a indicare se le affermazioni riportate qui di seguito risultano essere vere o false giustificandone la motivazione:

[V]                      [F]                      La chiusura di un trasferimento di dati binari via FTP è segnalata da <CRLF>.<CRLF>

[V]                      [F]                      La codifica bencoded è di tipo “senza perdite”

[V]                      [F]                      Una risoluzione DNS iterativa è più rapida di una ricorsiva

[V]                      [F]                      Il metodo Digest Access Authentication è immune ad un attacco di tipo “man in the middle”

#### **RISOLUZIONE**

[V]                      [F]                      La chiusura di un trasferimento di dati binari via FTP è segnalata da <CRLF>.<CRLF>

E' falso. Per i dati binari, la chiusura del trasferimento avviene nel momento in cui il client decide di interrompere la connessione. La chiusura della connessione è l'EOF (end of file).

[V]                      [F]                      La codifica bencoded è di tipo “senza perdite”

Vero. Difatti una volta decodificato un messaggio bencoded riusciamo a ricavare le stesse informazioni del messaggio iniziale prima che questo venisse codificato.

[V]                      [F]                      Una risoluzione DNS iterativa è più rapida di una ricorsiva

Solitamente vero, anche se possono esserci situazioni legate alla struttura di rete in cui accade il contrario. In generale, però, i tempi di risposta sono inferiori perchè il messaggio di risposta non deve ripercorrere la strada a ritroso attraverso tutti i server DNS che hanno contribuito alla risoluzione.

[V]                      [F]                      Il metodo Digest Access Authentication è immune ad un attacco di tipo “man in the middle”

Falso. Nonostante le password siano inviate crittografate, i messaggi scambiati sono comunque in chiaro. Ergo un terzo può leggerli e intromettersi comunque nella conversazione.

**Quesito n. 2 appello del 5 marzo 2009**

Si chiarisca il significato ed utilizzo dei seguenti codici di replica FTP:

- 2xx
- 3xx
- 4xx
- x0x
- x1x
- x2x
- x3x
- x4x
- x5x

**RISOLUZIONE**

- 2xx: Comando andato a buon fine. Esempio: 200 Command OK
- 3xx: Comando accettato, è necessario inserirne un altro per proseguire. Esempio: 331 Username OK, password required
- 4xx: Errore temporaneo. Riprovare in seguito. Esempio: 425: Can't open data connection
- x0x: Errore di sintassi. Esempio: 501 Syntax error
- x1x: Risposta ad una richiesta di informazioni. Esempio: 214 Help Message
- x2x: Risposta relativa alla connessione. Esempio: 125: Data connection already open. Transfer starting data
- x3x: Risposta relativa all'account o ai permessi. Esempio: 331
- x4x: Non specificato nell'RFC959
- x5x: Risposta relativa al file system. Esempio: 452 Error writing file error

**COMMENTO**

Oltre a queste è presente anche la classe "1xx" la quale fa riferimento ai comandi accettati ed in fase di elaborazione e "5xx" per gli errori permanenti.