Università di Genova
Facoltà di Ingegneria

Livello di Applicazione in Internet

3. HTTP (HyperText Transfer Protocol)

Prof. Raffaele Bolla
Ing. Matteo Repetto

dist

Introduzione

- Il World Wide Web (WWW) è l'applicazione che, a partire dagli inizi degli anni '90, ha maggiormente contribuito al successo e all'espansione di Internet
- L'HTTP (HyperText Transfer Protocol) è il protocollo di livello di applicazione che sta alla base del World Wide Web (WWW)
- L'HTTP definisce la struttura dei messaggi scambiati in una transazione Client-Server per la richiesta e il trasferimento di "pagine Web"

3.2

La pagina Web

- Una pagina Web può consistere di uno o più oggetti:
  - Un oggetto è un file indirizzabile attraverso un singolo URL.
  - I file possono contenere informazioni di diverso tipo: codice HTML (...), immagini JPEG o GIF, Java Applet, clip audio, etc.
- Generalmente una pagina Web è costituita da un file base di tipo HTML, che ne descrive la struttura generale e il contenuto testuale, e da altri oggetti a cui il file base HTML rinvia attraverso gli URL degli stessi .

3.3

La pagina Web

- Gli oggetti delle pagine Web possono essere:
  - Statici: sono costituiti da file di archivio che non hanno bisogno di uno stadio di pre-elaborazione o di interpretazione da parte del Server.
    - » Esempio: immagini GIF o JPEG, testi HTML
  - Dinamici: sono costruiti, al momento della richiesta, a partire da file tipo *script* che necessitano di pre-elaborazione da parte di uno specifico interprete presente sul Server. Per questo motivo uno stesso oggetto dinamico può apparire diverso ad ogni nuova richiesta HTTP.
    - » Esempio: testi, immagini, oggetti multimediali costruiti in PHP, ASP.....

3.4

Il Browser

- Il Browser è l'applicazione che realizza le funzionalità di Client Web.
- I Browser, generalmente, contengono sia le funzionalità di User-Agent per l'applicazione Web, sia la realizzazione del lato client per l'HTTP.
- I Browser più diffusi sono Mozilla, MS Internet Explorer, Netscape Navigator e Opera.

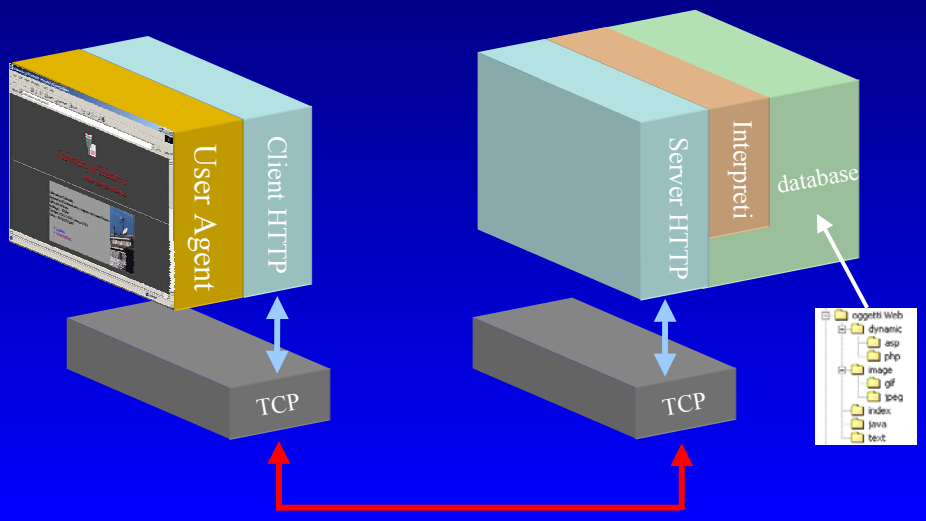
3.5

Il Server Web

- Il Server Web implementa sia il lato Server dell'HTTP sia un particolare database.
- All'interno di questo database è possibile reperire:
  - Gli oggetti statici delle pagine Web che risiedono sul Server stesso.
  - Tutti i file da elaborare, ognuno con un interprete specifico, con cui costruire gli oggetti dinamici.
- Apache, MS Internet Information Server e Netscape Enterprise Server sono solo alcuni esempi di Server Web molto diffusi.

3.6

Server Web e Browser



3.7

HTTP: introduzione

- L'HTTP necessita di trasmissioni affidabili (senza perdita di pacchetti), e, quindi, utilizza come protocollo di trasporto il TCP ed il numero di porta 80.
- Poiché i Server HTTP non mantengono alcuna informazione sui Client e ogni transazione Client-Server è gestita indipendentemente dalle altre, l'HTTP è definibile come un protocollo "senza stato".

3.8

HTTP 1.0 e 1.1

- Esistono due versioni di questo protocollo:
  - HTTP 1.0 definito nel RFC 1945
  - HTTP 1.1 definito nel RFC 2068

  Le due versioni (1.1 e 1.0) risultano compatibili tra loro (un Client HTTP 1.0 può "dialogare" con un Server HTTP 1.1 e viceversa)
- Queste due versioni dell'HTTP si distinguono solo per quanto concerne la gestione della connessione:
  - La versione 1.0 può utilizzare solo una connessione di tipo "non permanente"
  - La connessione di tipo "permanente" è, invece, di *default* per la versione 1.1

3.9

Connessione non permanente

- Il trasferimento di una pagina Web con HTTP in modalità "connessione non permanente" implica l'utilizzo di una diversa connessione TCP per ogni singolo oggetto da trasferire.
- Analizziamo il trasferimento di una pagina Web (per es. all'URL http://www.reti.dist.unige.it/index.html) con connessione in modalità non permanente. Supponiamo inoltre che la pagina Web da trasferire consista in un file base HTML e di altri 10 oggetti (immagini JPEG), tutti residenti, per semplicità, sullo stesso Server.

3.10

Connessione non permanente

- Ecco ciò che succede:
  1. Il Client HTTP apre una connessione TCP verso il Server.
  2. Il Client HTTP, utilizzando il Socket associato alla connessione aperta al passo precedente, invia il messaggio di richiesta HTTP per l'oggetto (il file base) che risiede all'URL ***http://www.reti.dist.unige.it/index.html***.
  3. Il Server HTTP riceve, sempre tramite un Socket associato alla connessione iniziata al passo 1, il messaggio di richiesta, cerca l'oggetto (il file base) che risiede all'URL richiesto, quindi lo incapsula in un messaggio HTTP di risposta e lo invia al Client utilizzando sempre la stessa connessione TCP

3.11

Connessione non permanente

4. Il Server HTTP chiude la connessione TCP
5. Il Client HTTP riceve il messaggio di risposta e la connessione TCP viene conclusa.
6. Il Client HTTP estrae dal messaggio di risposta l'oggetto trasmesso e, se esso fa riferimento ad altri oggetti (come nel caso del file base HTML), ripete il procedimento fin qui illustrato per richiedere al Server HTTP tutti gli oggetti correlati a quello appena estratto.

3.12

Connessione permanente

- Con la connessione permanente il Server HTTP mantiene aperta la connessione TCP dopo aver inviato la risposta al Client al fine di utilizzarla per la trasmissione di successive richieste e risposte: in particolare è possibile utilizzare la stessa connessione per il trasferimento di più pagine Web complete (file base+oggetti) residenti sullo stesso Server.
- Data la particolare struttura del controllo di congestione del TCP (slow-start) il trasferimento di pagine Web con connessione permanente e quindi il comportamento di default dell'HTTP 1.1 risulta più efficiente e veloce di quello a connessione non permanente.

3.13

Connessione permanente

- D'altra parte l'utilizzo della connessione in modalità permanente presenta alcuni problemi:
  - –Per ogni oggetto richiesto deve essere stabilita e mantenuta una nuova connessione. (Aumenta il numero di connessioni contemporaneamente attive)
  - –Per ognuna delle connessioni stabilite devono essere allocati e mantenuti i buffer e le variabili del TCP sia sul Client che sul Server
- Questo potrebbe portare a caricare oltremodo un Server che deve comunicare contemporaneamente con centinaia di Client

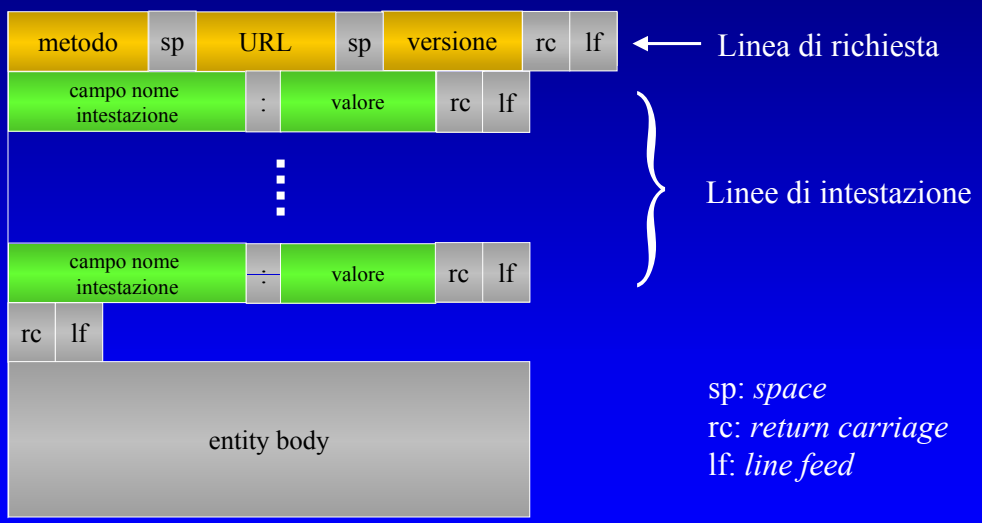
3.14

Il messaggio HTTP

- I messaggi HTTP sono scritti con il testo in codice ASCII.
- I messaggi HTTP possono essere suddivisi in due diverse categorie:
  - Messaggi di richiesta.
  - Messaggi di risposta.
- Ogni messaggio è suddiviso in tre parti principali:
  - Linea di richiesta o di stato.
  - Linee di intestazione.
  - Contenuto (*entity body*).

3.15

Il messaggio HTTP di richiesta



3.16

Il messaggio HTTP di richiesta

- La linea di richiesta presenta 3 campi:
  - Metodo: questo campo può assumere i seguenti valori:

options	head	put	copy	delete	unlink	wrapped
get	post	patch	move	link	trace	Ext.method

    Il metodo maggiormente utilizzato è il get, poiché è quello che consente ai Client di richiedere un oggetto
  - URL: l'indirizzo URL dell'oggetto richiesto
  - Versione (dell'HTTP): versione HTTP utilizzata dal Client

3.17

Il messaggio HTTP di richiesta

- Le linee immediatamente successive a quella di richiesta sono dette linee di intestazione.
- Le linee di intestazione possono essere distinte in tre diverse categorie sulla base della tipologia di informazione contenuta:
  - *General header line*: contiene informazioni generali sulla modalità di transazione Client-Server.
    - » Cache-Control, Connection, Date, Forwarded, Mime-Version.
  - *Request header line*: contiene informazioni relative alle richieste e al Client.
    - » Accept-Language,Accept-Encoding, Authorization, UserAgent.
  - *Entity header line*: contiene informazioni sulla risorsa richiesta e ciò che è contenuto nel corpo dell'entità.

3.18

Il messaggio HTTP di richiesta

- Il messaggio HTTP di richiesta prevede anche la presenza (opzionale) di un Contenuto (*Entity Body*)
- L'utilizzo di questo campo è previsto solo da alcuni metodi di richiesta (come per esempio il metodo *post*)

3.19

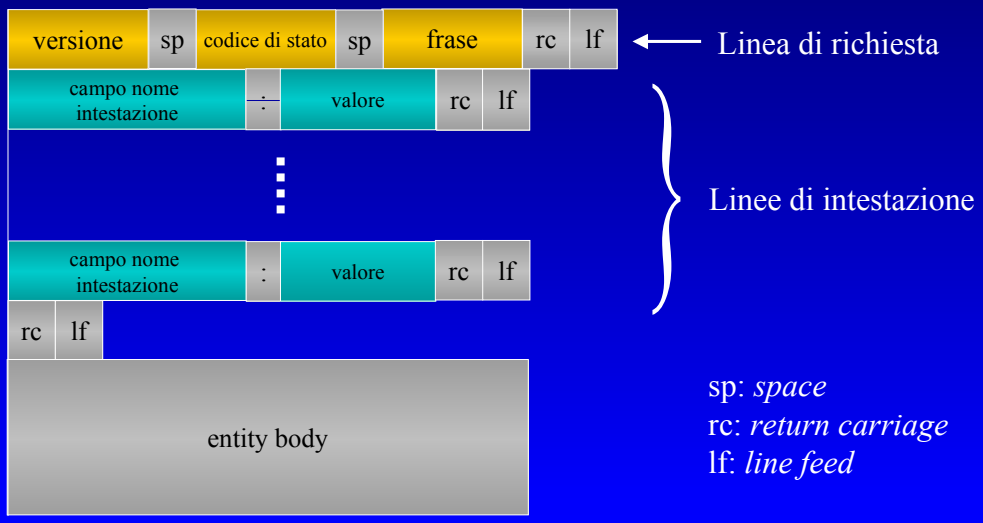
Il messaggio HTTP di richiesta

- Esempio pratico di un tipico messaggio HTTP di richiesta:

GET /reti/index.html HTTP/1.0 — Linea di richiesta
Host: www.reti.dist.unige.it
User-agent: Mozilla/4.0
Accept: text/html, image/gif,image/jpeg
Accept-language: it
} Linee di intestazione

3.20

Il messaggio HTTP di risposta



3.21

Il messaggio HTTP di risposta

- Anche il messaggio di risposta dell'HTTP può essere suddiviso in tre parti differenti:
  - Linea di stato
  - Linee di intestazione
  - Contenuto
- Il contenuto, poiché contiene l'oggetto richiesto, costituisce il nucleo del messaggio di risposta

3.22

Il messaggio HTTP di risposta

- La linea di stato è composta da tre diversi campi:
  - Versione: la versione di HTTP utilizzata dal Server.
  - Codice di stato: descrive lo stato della risposta alla richiesta inoltrata
    - »200, 400, 404, 301, 505
  - Frase: contiene un messaggio relativo allo stato descritto nel campo precedente
    - » OK, Bad Request, Forbidden, Not Found, Internal Server Error

3.23

Il messaggio HTTP di risposta

- Le linee di intestazione, analogamente al messaggio di richiesta, possono essere classificate in tre diverse categorie:
  - General header line: contiene informazioni generali sulla modalità di transazione Server-Client
    - » Gli stessi campi del messaggio di richiesta
  - Response header line: contiene informazioni relative alle risposte e al Server
    - » Location, Public, Server, Proxy Authenticate, Retry After
  - Entity header line: contiene informazioni concernenti all'oggetto contenuto nel corpo dell'entità
    - » Content Type, Content Length, Last Modified

3.24

Il messaggio HTTP di risposta

- Esempio di un tipico messaggio HTTP di risposta:

HTTP/1.1 200 OK — Linea di richiesta
Connection: Close
Date: Thu, 03 Feb 2003 09:15:04 GMT
Server:Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jul 2003 15:43:23 GMT
Content-Length:14287
Content-Type: text/html
— Linee di intestazione

[dati.....] — *Entity Body*

3.25

Autenticazione e Cookie

- E' spesso molto utile che un sito Web possa identificare gli utenti, sia al fine di limitare gli accessi al Server, sia al fine di dispensare contenuti in funzione dell'identità dell'utente.
- L'HTTP, essendo un protocollo senza stato, non permette una risoluzione diretta di questo problema. Per risolvere questo problema vengono, dunque, utilizzati due meccanismi:
  - Autenticazione.
  - Cookie.

3.26

Autenticazione

- L'autenticazione è un metodo di identificazione dell'utente tramite *username* e *password.*
- L'HTTP, per realizzare questo meccanismo, mette a disposizioni particolari codici di stato e intestazioni.
- Esempio Client-Server con autenticazione:
  - Il Client invia un messaggio di richiesta ordinario .
  - Il server risponde con un corpo dell'entità vuoto e un codice di stato 401 Authorization Required e una linea di intestazione con WWW-Authenticate che specifica le modalità di identificazione.
  - Il Client, a sua volta, risponde con un messaggio di richiesta contenente la linea di intestazione Authorization: completa di *username* e *password.*

3.27

Cookie

- I Cookie, definiti nel RFC 2109, costituiscono un meccanismo utilizzabile dai siti Web per associare specifiche informazioni agli utenti. Possono essere utilizzati per diversi scopi quali:
  - l'autenticazione senza la richiesta di *username* e *password*
  - Per memorizzare le preferenze di un utente
    - » Esempio pubblicità mirata in visite successive
  - Per conservare una sorta di stato della transazione in corso
    - » Esempio elenco dei prodotti nel carrello della spesa in un sito di acquisti on-line

3.28

Cookie

- Esempio di funzionamento del Cookie:
  1. Il Client contatta il sito Web per la prima volta
  2. Nel messaggio di risposta il Server include una linea di intestazione del tipo "Set-Cookie: <*cookie*>", dove il campo cookie comprende:
     Nome del cookie, valore, scadenza, cartella/e, dominio
  3. Il Client riceve la risposta e aggiunge una linea, contenente i dati del cookie ricevuto, ad un particolare file cookie interno al browser
  4. Nei successivi messaggi di richiesta il Client utilizzarà, dove richiesto dal cookie stesso, la linea di intestazione "Cookie:" e i dati di identificazione reperibile nel file di cookie
  5. A questo punto il Server conosce le informazioni che in precedenza aveva associato all'utente

3.29

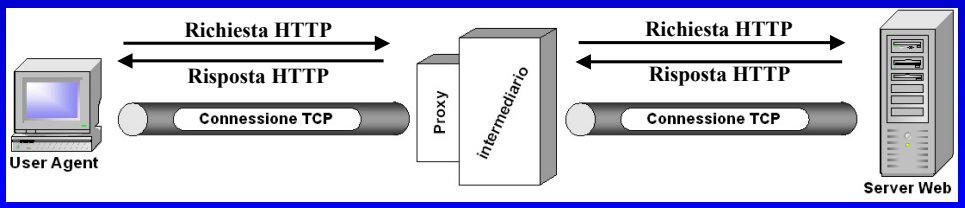
Sistemi intermediari

- Il funzionamento dell'HTTP può prevedere l'esistenza di alcuni sistemi intermediari.
- In presenza di questi sistemi intermediari non vi sarà una connessione *end-to-end* diretta tra il browser e il server Web, e le stesse richieste o risposte HTTP saranno in qualche modo "filtrate" o "permutate" da ognuno dei sistemi intermediari.
- Esistono tre tipologie di sistemi intermediari:
  - Proxy
  - Gateway
  - Tunnel

3.30

Proxy Server

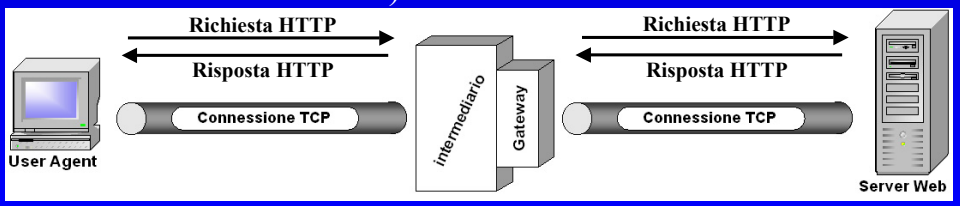
- Un Proxy agisce per conto di altri Client e presenta le richieste di tali Client ad un Server, operando come Server nell'iterazione con il Client e da Client nell'iterazione con il Server
- Vi sono due situazioni che richiedono l'uso di un Proxy:
  - Intermedio di sicurezza (es. firewall)
  - Interfaccia tra differenti versioni di HTTP



3.31

Gateway

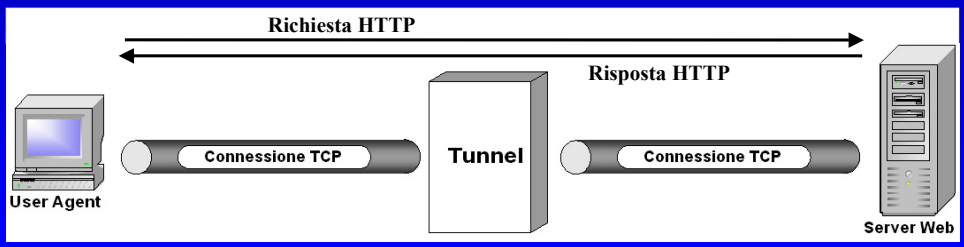
- Un Gateway è un server che appare al Client come se fosse il Server di destinazione, ed agisce per conto di Server che potrebbero non essere in grado di comunicare direttamente con il Client
- Vi sono due scenari in cui si può utilizzare un Gateway:
  - Intermediario di sicurezza (es. firewall)
  - Interfaccia con Server non-HTTP (realizzati con protocolli differenti dall'HTTP)



3.32

Tunnel

- A differenza del Proxy e del Gateway, il Tunnel non opera sulle richieste e risposte HTTP ma è semplicemente un punto di collegamento fra due connessioni TCP



3.33

Web Cache

- La **Web Cache** è un meccanismo presente nei Proxy Server che permette la memorizzazione degli oggetti delle pagine Web richieste recentemente.
- Le funzionalità di Web cache sono utili per:
  - Ridurre il tempo di risposta a una richiesta del Client.
  - Ridurre il traffico sul link di accesso ad Internet.
  - Ridurre il traffico globale all'interno di Internet.
  - Fornire un'infrastruttura per la rapida distribuzione dei contenuti, anche per siti che risiedono su server lenti e con poche risorse.

3.34

Web Cache

- Il ***Proxy Server*** consulta la Web cache ad ogni richiesta HTTP da parte di un Client:
  - Se l'oggetto cercato è contenuto nell'archivio locale, allora il Proxy inoltra direttamente al Client un messaggio di risposta contenente l'oggetto.
  - Se la Web cache non contiene l'oggetto, allora il Proxy Server invierà un messaggio di richiesta al Server di destinazione. Quando il Proxy riceve il messaggio di risposta, memorizza l'oggetto all'interno della Web cache e lo inoltra al Client tramite un messaggio di risposta HTTP.

3.35

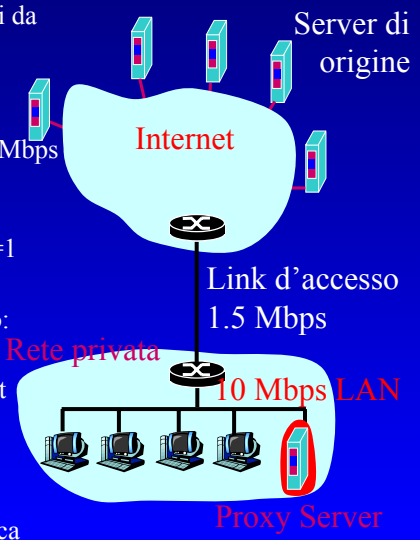
Caching Cooperativo

- Web cache multiple, situate in diversi siti, possono cooperare e migliorare le performance globali della rete.
- Le diverse cache possono essere organizzate in modo gerarchico.
- Un esempio di "caching" cooperativo è NLANR:
  - Molte cache a livello nazionale servono cache di diverse istituzioni sparse per il mondo.
- Una struttura alternativa è rappresentata dal "cluster di cache":
  - I Client consultano una cache diversa in base all'URL ricercato (es. tramite funzioni di Hash)

3.36

Web Cache

- Valutiamo, in modo grossolano, i benefici apportati da una Web Cache nel caso illustrato in figura:
- Dimensione media oggetti=100Kbit
- Tasso medio interarrivo richieste oggetti =15ric./s
- Intensità di traffico sulla Lan = 15ric./s.*100Kb/10Mbps = 0.15
- Intensità di traffico sul link di accesso:
  - senza Web cache = 15ric./s.*100Kb/1.5Mbps =1
  - con Web cache con *hit-rate* del 40% = 0.6
- Supponendo il ritardo medio di ricerca dell'oggetto:
  - Sul Proxy = 0.010 s.
  - Sul Server di destinazione dal Gateway Internet = 2.0 s.
- Ritardo medio:
  - Senza cache Web >> 2 sec. (ordine dei minuti)
  - Con cache Web = 0.6*2+0.4*0.010 = 1.2 s. circa



3.37

Configurazione di un server Web

- Sistema operativo: *Linux*
- Distribuzione: *Debian (Woody)*
- Server: *Apache 1.3.26*

3.38

I file di configurazione di Apache

- Apache si configura inserendo delle "direttive" all'interno di file di configurazione testuali.
  - *httpd.conf*: direttive generali per il server globale;
  - *srm.conf*: gestione delle risorse e dei nomi all'interno del filesystem (directory, indici, alias, ecc);
  - *access.conf*: accesso alle varie directory del sistema
  - altri file di configurazione tramite la direttiva **Include.**

3.39

Sintassi dei file di configurazione

- Il carattere "#" ad inizio riga indica un commento.
- È possibile spezzare un comando su più righe utilizzando "\"
- Le direttive nei file di configurazione sono "*case insensitive*", ma gli argomenti delle stesse sono spesso "*case sensitive*".
- Si può verificare la correttezza della sintassi dei file di configurazione tramite il comando:
  apachectl configtest

3.40

Configurazione di Apache

- Parametri di funzionamento del server (tra cui possono essere presenti specifiche HTTP*);*
- Indicazioni dei moduli dinamici;
- Direttive di accesso alle diverse sezioni del filesystem;
- Autenticazione;
- Virtual hosting;
- Connessioni cifrate.

3.41

httpd.conf

- Le direttive che vedremo all'interno del file di *httpd.conf* sono organizzate in tre sezioni:
  - direttive che controllano l'esecuzione del server Apache come programma;
  - direttive che definiscono i parametri del *main server*, a cui si riferiscono tutte le richieste che non fanno capo ad un virtual host;
  - configurazioni dei virtual host.

3.42

Sezione 1: *Global Environment*

- **ServerType**: standalone o inetd.
- **ServerRoot**: dove risiedono i file di configurazione con percorsi "relativi".
- **ResourceConfig**, ***AccessConfig***: locazione dei file *srm.config* e *access.config*. Il consiglio attuale è di non usare più questi file.
- **KeepAlive**: permette (on) o disabilita (off) la connessione permanente.
- **MaxKeepAliveRequests**: numero di richieste contemporanee che si possono accettare sulla stessa connessione (connessione incanalata o non incanalata).
- **KeepAliveTimeout**: tempo di attesa prima della chiusura della connessione.

3.43

Sezione 1: *Global Environment – 2*

- **Listen:** specifica ulteriori indirizziIP/porte di ascolto del server per le richieste di connessione.
- **LoadModule**: permette di caricare all'avvio moduli non compilati nel server stesso in grado di svolgere funzioni di supporto (es. configurazione, supporto imap, supporto script, supporto ssl).
- **Extended-status**: On/Off. Apache permette di monitorare il funzionamento del server; questa direttiva permette di estendere la lista delle informazioni disponibili. Da utilizzare in copia con la direttiva **<Directory server-status>**.

3.44

Sezione 2: *Main Server*

- Le direttive in questa sezione configurano il server di default che gestisce le richieste che non sono indirizzate a nessun altro virtual host.
- I valori definiti in questa sezione valgono come default in tutte le sezioni VirtualHost.
- **Port:** la porta TCP di ascolto del server in modalità *standalone*.
- **ServerAdmin**: l'indirizzo di posta del gestore del server (compare in alcune segnalazioni di errore agli utenti, spesso in corrispondenza del nome dle server nei messaggi di errore).
- **ServerName**: il nome dell'host, spesso utilizzato nei messaggi di errore. Utilizzato dala direttiva UseCanonicalName.
- **DocumentRoot**: la directory dove si trovano i file del sito.

3.45

La direttiva <Directory>

- Ogni cartella a cui Apache ha accesso può essere configurata sulla base di quali servizi e funzionalità sono abilitate o disabilitate in quella particolare cartella (e nelle sue sottocartelle):
- **Options**
- **AllowOverride**
- **<Limit *metodi*>** *controlli* **</Limit>**

3.46

Opzioni di <Directory>

- **ExecCGI**: abilita l'esecuzione di script CGI in questa directory.
- **FollowSymLinks**: permette si seguire i link simbolici.
- **Includes**: permette i SSI (Server-Side Include).
- **SymLinksIfOwnerMatch**: segue i link simbolici se i file destinazione appartengono allo stesso utente.
- **IncludesNOEXEC**: come il precedente, ma disabilita i comandi #exec e #include negli script CGI.
- **Indexes**: visualizza una lista dei contenuti della cartella, se non esiste nessun file specificato da **DirectoryIndex**.
- **Multiview**: permette la negoziazione dei contenuti con il metodo *multiviews*
  - *il server completa il nome richiesto con i file effettivamente disponibili e sceglie la versione più adatta per il browser.*

3.47

AllowOverride

- Permette ai file .htaccess di modificare gli attributi specificati:
  - None: nessun attributo può essere modificato;
  - Options: la direttiva options può essere specificata;
  - FileInfo: permette la definizione di direttive del tipo AddType e AddEncoding;
  - Indexes: permette la modifica dei parametri utilizzati nella generazione automatica dell'indice della directory (come AddDescription);
  - AuthConfig: definizione di AuthName, AuthType, AuthUserFile e AuthGroupFile;
  - Limit: direttiva Limit
  - All: permette la modifica di tutti i parametri.

3.48

<Limit>

- I controlli di accesso si applicano a **tutti** i metodi http.
- **In generale, le direttive per il controllo di accesso non devono essere poste entro una sezione <limit>.**
- Lo scopo della direttiva <Limit> è quello di limitare l'effetto del controllo di accesso ai soli metodi HTTP nominati. Tutti i metodi non citati non sono interessati.
- Il nome del metodo è *case-sensitive.*

3.49

Autenticazione, Autorizzazione e Controllo di accesso

- Autenticazione: verificare l'identità dell'utente:
  - login, password
- Autorizzazione: verificare se una persona ha accesso ad una determinata risorsa:
  - permessi relativi a singoli utenti o gruppi
- Controllo di accesso: verificare condizioni indipendenti dall'utente stesso:
  - indirizzo IP di provenienza, ora del giorno, ecc.

3.50

Autenticazione e autorizzazione

- Apache mette a disposizione 3 meccanismi per l'autenticazione e l'autorizzazione degli utenti:
  - **Autenticazione di base**
    - » semplice
    - » le password viaggiano in chiaro sulla rete
  - **Autenticazione mediante Digest**
    - » solo un digest della password viene trasmesso sulla rete
    - » non tutti i browser la supportano
    - » il digest è replicabile
  - **Autenticazione tramite database**
    - » accesso veloce nel caso si abbiano centinaia di

3.51

Autenticazione di base

- Può essere descritta:
  - all'interno di una direttiva <Directory>;
  - in un file all'interno della directory stessa
    - » il nome è specificato nella direttiva **AccessFileName** (di solito ".htaccess").
- Richiede di specificare:
  - il meccanismo di autenticazione;
  - gli utenti e le rispettive password;
  - eventuali gruppi di utenti.
- Il browser memorizza la password che l'utente fornisce finché non viene chiuso.

3.52

Autenticazione di base: configurazione

- **AuthType**: basic
- **AuthName**: messaggio di richiesta che il browser visualizza (realm)
- **AuthUserFile**: il file delle password
- **AuthGroupFile**: descrizione dei gruppi
- **Require:**
  - user: elenco degli utenti autorizzati
  - group: elenco dei gruppi di utenti autorizzati
  - valid-user: qualsiasi utente presente nel file specificato da AuthUserFile

3.53

Controllo di accesso

- **Order**
  - Deny,Allow;
  - Allow,Deny;
  - Mutual-failure: sconsigliato, corrisponde a Allow,Deny.
- **Allow from**
  - nome dominio (anche parziale)
    - » reti.dist.unige.it, unige.it
  - indirizzo IP (anche parziale)
    - » 10.1, 10.3.4.2
  - indirizzo di rete
    - » 10.3.0.0/25 o 10.3.0.0/255.255.0.0

3.54

Accesso alle cartelle – Esempio

```
<Directory /var/www/lai/>
   Order deny,allow
   Deny from all
   Allow from reti.dist.unige.it
</Directory>
```

Solo gli host interni al dominio reti.dist.unige.it possono accedere

```
<Directory /var/www/lai/>
   Order deny,allow
   Deny from all
   Allow from abete.reti.dist.unige.it
</Directory>
```

Solol'host abete.reti.dist.unige.it può accedere

3.55

Sezione 2: *Main Server – 2*

- **DirectoryIndex**: elenco (ordinato) dei file da visualizzare di default per gli URL contenenti una directory.
- **UserDir**: permette di creare aree gestite dai singoli utenti del sistema.
- **AccessFileName**: Nome del file per il controllo di accesso delle singole directory.
- **UseCanonicalName**: On/Off. Nei messaggi di errore visualizza l'indirizzo IP del server (Off) o il suo nome (On).
- **DefaultType**: estensione MIME di default da utilizzare per i documenti (di solito text/plain è una buona scelta).
- **HostnameLookups**: On/Off. Log degli indirizzi o dei nomi dei client.

3.56

Sezione 2: *Main Server* – 3

- **ServerSignature**: abilita la stampa di una riga contenente la versione del server e l'email del responsabile (opzionale) alle pagine generate automaticamente dal server (errori, elenco directory, ...).
- **Alias**: nomi alternativi.
- **ScriptAlias**: analogo ad Alias, ma i file sono considerati come script da elaborare.
- **Redirect**: redirezione della richiesta di un URL su un nuovo url.

3.57