Università di Genova
Facoltà di Ingegneria

Telematica

14. Livello di Applicazione

Prof. Raffaele Bolla
Ing. Matteo Repetto

dist

Introduzione al Livello di Applicazione

- La struttura del software/hardware che realizza le comunicazioni nella rete segue una architettura a strati di cui il strato/livello di Applicazione è quello più alto e quindi più vicino all'utente
- Livello di applicazione: protocollo e applicativi
  - Una applicazione "di rete" è composta processi applicativi distribuiti su più nodi terminali (*host*)
  - I processi su *host* diversi comunicano scambiandosi messaggi attraverso la rete, rispettando opportune regole e formati definiti tramite protocolli.

1.2

Esempi di applicazioni Internet

- Trasferimento di File (FTP)
- Teminale remoto (Telnet)
- **WWW (http)** ⇐
- **Posta elettronica (SMTP)** ⇐
- File System distribuiti (NFS)
- **Traduzione nomi-indirizzi (DNS)** ⇐
- Applicazioni di condivisione dati (Gnutella)
- Sistemi di viceoconferenza (H323, RTP)
- Sistemi per la distribuzione di video/audio broadcasting

1.3

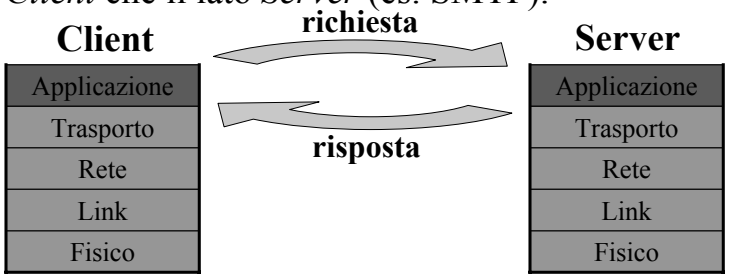
I Protocolli del Livello di Applicazione

- Un protocollo del Livello di Applicazione è solo una parte di una Applicazione di rete (http è solo una parte del Web)
- Un protocollo del Livello di Applicazione, in generale, stabilisce come i processi comunicano in rete.
- In particolare esso definisce:
  - La tipologia di messaggi da scambiare.
  - La sintassi dei messaggi.
  - La semantica dei campi.
  - Quando e come un processo invia o risponde ai messaggi.

1.4

Client e Server

- La comunicazione in rete tra due entità avviene individuando un *Client* ed un *Server.*
- Il *Client* è quell'*host* che inizia una sessione di dialogo formulando una richiesta ad un altro *host* (il *Server*), il quale risponderà.
- Per molte applicazioni, uno stesso *host* utilizzerà sia il lato *Client* che il lato *Server* (es. SMTP).



1.5

Interfacce Socket

- I processi *Client* e *Server* comunicano tra loro inviando e ricevendo messaggi tramite le loro interfacce: i *Socket.*
- Un *Socket* è definibile come un API (*Application Programmers' Interface*) tra l'applicazione e la rete, si tratta infatti di un'interfaccia fornita dal Livello di Trasporto per poter utilizzare i servizi di "trasporto" della rete in maniera trasparente.
- I soli aspetti che il gestore dell'applicazione può controllare in relazione al livello di trasporto sono:
  - Scelta del protocollo di trasporto (TCP o UDP);
  - Eventuale possibilità di stabilire alcuni parametri.

1.6

Applicazioni

- Non tutte le applicazioni di rete fornisco servizi direttamente all'utente (umano).
- Alcune applicazioni fornisco loro volta un servizio ad altre applicazioni (ad esempio il DNS, *Domain Name System*) o integrano applicazioni locali estendendone le funzionalità alla rete (ad es. i NFS, *Network File System*)

1.7

User agent

- Lo *User Agent* (Agente d'utente) è un'interfaccia tra l'utente e le applicazioni di rete che forniscono un servizio diretto all'utente.
- I *browser* (Mozilla, Explorer, Netscape, …) costituiscono un chiaro esempio di *User Agent* per l'applicazione di rete Web.
- Eudora, Outlook e Messenger sono invece tre diversi *User Agent* per l'applicazione di posta elettronica.

1.8

Processo di Indirizzamento

- Affinché un processo di un *host* possa inviare messaggi al processo di un altro *host*, esso dovrà conoscere:
  - L'indirizzo IP dell'*host* di destinazione.
  - L'identificativo del processo (numero di porta), sull'*host* di destinazione, che deve ricevere il messaggio.
- Ai protocolli più comuni del Livello di Applicazione sono stati assegnati [RFC 1700] numeri di porta specifici:
  - HTTP num. di porta 80, POP3 num. di porta 110, SMTP num. di porta 25, IMAP num. di porta 143, DNS num. di porta 53

1.9

URL: *Uniform Resource Locator*

- E' un indirizzo particolare che identifica la posizione delle risorse nella rete specificando anche le modalità per accedervi.
- La sua forma è del tipo

  <*scheme*>:<*scheme-specific-part*>

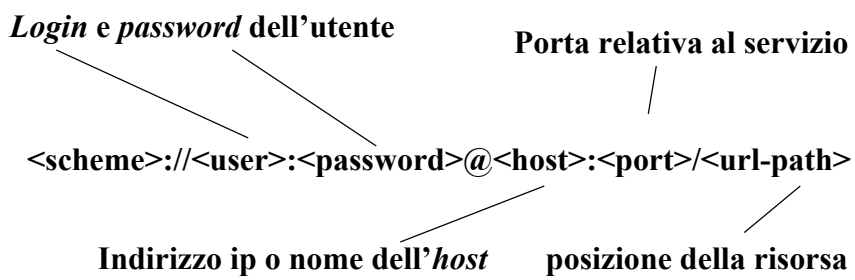
  dove:
  - <*scheme*> è, in sostanza, la modalità generale di accesso, ossia il protocollo (ftp, http, mailto ...)
  - <*scheme-specific-part*> è nella forma:

    //<user>:<password>@<host>:<port>/<url-path>

- <*user*> e <*password*> sono la *login* e la *password* dell'utente
- <*host*> è il nome o l'IP dell'*host* e
- <*port*> è la porta relativa al servizio
- <*url-path*> è la posizione della risorsa

1.10

URL: esempio



Esempi:

ftp://rossi@zeus.com.dist.unige.it:21/lavori
ftp://rossi:mr82@zeus.com.dist.unige.it:21/lavori
ftp://bianchi@130.251.8.191:21/lavori
http://zeus.com.dist.unige.it:8080/~bianchi
http://130.251.8.191:8080/~rossi
file:///C:/documenti/orario.doc

1.11

Livello di Applicazione

Domain Name System (DNS)

1.12

DNS: Identificazione degli host

- Ogni *host* è identificabile in due differenti modalità:
  - Indirizzo IP (ad es. 121.7.106.83)
  - *Hostname* (ad es. www.google.com)

  L'identificazione tramite *hostname* (URL) è normalmente preferita dall'utenza in quanto maggiormente mnemonica rispetto ad un indirizzo IP.

  Gli indirizzi IP, caratterizzati da una lunghezza e da una strutturazione gerarchica fissa, sono invece più efficacemente utilizzabili nell'instradamento rispetto agli *hostname*
- Il meccanismo che realizza la traduzione tra *hostname* e Indirizzo IP e viceversa è detto *Domain Name System* (DNS)

1.13

DNS: introduzione

- Il DNS è un'entità del Livello di Applicazione, infatti fra l'altro:
  - funziona tra terminali comunicanti che usano il paradigma *Client-Server*,
  - si appoggia a un protocollo di trasporto *end-to-end* per trasferire i messaggi DNS tra i due terminali.
- Normalmente, però, non viene utilizzato direttamente dall'utente ma viene richiamato da altre applicazioni (Web, E-mail...).

1.14

DNS: introduzione

- Il DNS è in generale:
  - Un database distribuito realizzato tramite una struttura gerarchica di ***Name Server***.
  - Un protocollo dello strato di applicazione che permette agli *host* di comunicare con i *Name Server* ed ai *Name Server* di interagire fra loro, in modo da fornire il servizio di traduzione.
- I *Name Server* operano prevalentemente su macchine Unix usando l'implementazione software "***B**erkeley **I**nternet **N**ame **D**omain*" (BIND)
- Il protocollo DNS utilizza UDP e la porta 53 (in casi particolari può operare su TCP)

1.15

DNS: introduzione

- Il DNS, inoltre, fornisce altri servizi molto importanti:
  - **Alias degli *hostname***: un *hostname* può avere più di un alias del nome. L'*hostname* originale è detto canonico. Gli alias sono solitamente più mnemonici (www ,ftp , smtp) dell'*hostname* canonico.
  - **Identificazione server di posta:** usato per legare un dominio all'indirizzo di un server di posta. E' possibile indicare eventuali *Server* di posta aggiuntivi da utilizzare in caso di guasto del principale.
  - **Distribuzione del carico**: il DNS è utilizzato anche per ripartire il carico di traffico tra diverse repliche di uno stesso *Server* (ciascuna replica di *Server* su un *host* diverso).

1.16

Name Server

- Il database distribuito è realizzato tramite una gerarchia di molti *Name Server.*
- Nessun *Name Server* possiede nel proprio *database* tutte le possibili traduzioni *hostname*/indirizzo IP.
- Perché il DNS non è centralizzato?
  - Volume di traffico.
  - Il database centralizzato potrebbe essere troppo distante dagli *host.*
  - Struttura distribuita più ridondante, quindi più robusta ai guasti e di manutenzione più agevole.

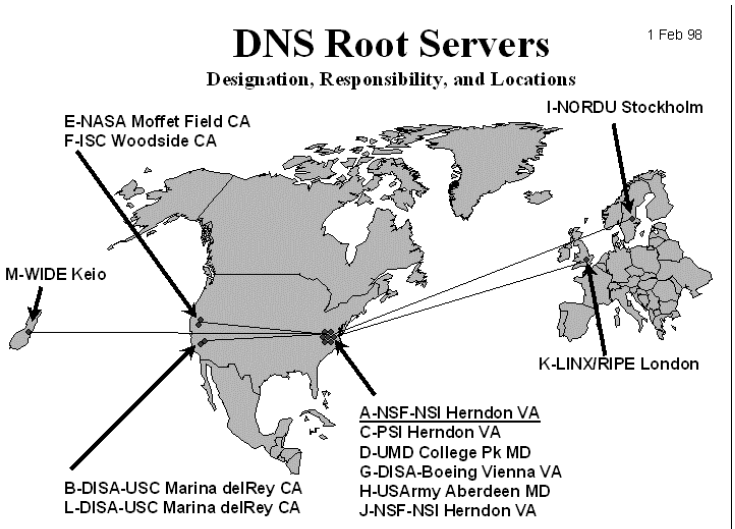
1.17

Name Server

- Local Name Servers:
  - ogni ISP ha un *local (default) name server*
  - le richieste di DNS di ogni host vengono inizialmente sempre indirizzate al *local name server*
- Root Name Servers:
  - Quando un *local name server* non riesce a soddisfare la richiesta, esso si comporta come un *Client* DNS e inoltra tale richiesta al *Root Name Server* (ne esistono alcune dozzine)
  - Se il R.N.S. non possiede la traduzione dell'*hostname* risponde inviando l'indirizzo di un *Authorative Name Server* che ha la corrispondenza di quel particolare *hostname*

1.18

Name Server



1.19

Name Server

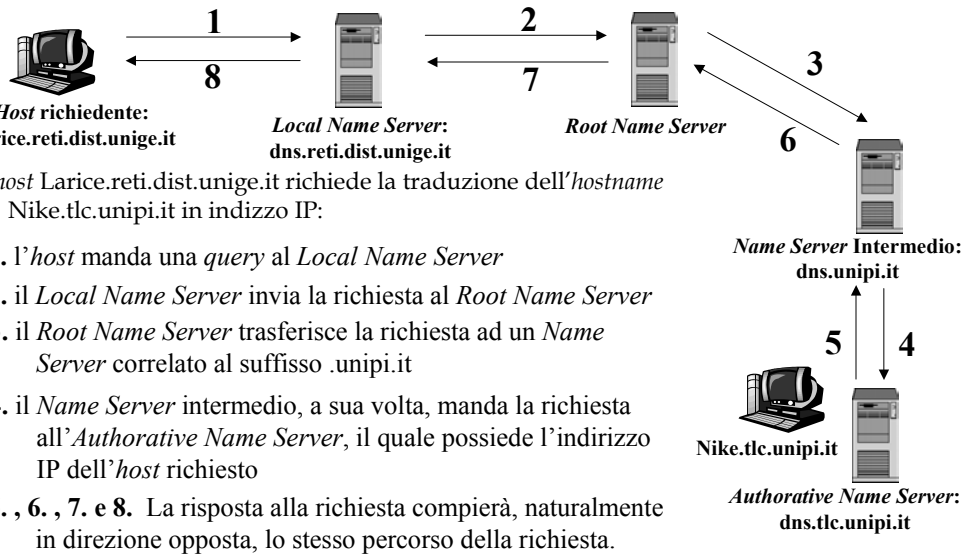
- Authoritative Name Server:

  Ciascun *host* è registrato presso un *Authoritative Name Server* che, tipicamente, per ogni *host* è il *Name Server* situato presso il suo ISP locale.

  Quando un *Authoritative Name Server* riceve una richiesta da un *Root Name Server*, esso risponde con la "traduzione" richiesta. Il *Root Name Server*, a sua volta, invierà la "traduzione" al *Local Name Server*, il quale, infine la girerà all'host richiedente.

1.20

DNS: Esempio



L'*host* Larice.reti.dist.unige.it richiede la traduzione dell'*hostname* Nike.tlc.unipi.it in indizzo IP:

1. l'*host* manda una *query* al *Local Name Server*

2. il *Local Name Server* invia la richiesta al *Root Name Server*

3. il *Root Name Server* trasferisce la richiesta ad un *Name Server* correlato al suffisso .unipi.it

4. il *Name Server* intermedio, a sua volta, manda la richiesta all'*Authorative Name Server*, il quale possiede l'indirizzo IP dell'*host* richiesto

5. , 6. , 7. e 8. La risposta alla richiesta compierà, naturalmente in direzione opposta, lo stesso percorso della richiesta.

1.21

Meccanismi per aumentare l'efficienza

- ***Iteratives query***
  - Ogni Server lungo la catena, se non la corrispondenza memorizzata direttamente, può fornire l'indirizzo IP del successivo Server della catena (riducendo la lunghezza del cammino di ritorno).
- ***Caching***
  - Ogni Server mantiene le ultime traduzioni in una memoria temporanea per un periodo di tempo configurabile
- ***Forwarding***
  - Invece che rivolgersi al *Root Server*, il *Local Server* può delegare un Server intermedio che conosce essere particolarmente "capace"

1.22

Livello di Applicazione

Posta Elettronica

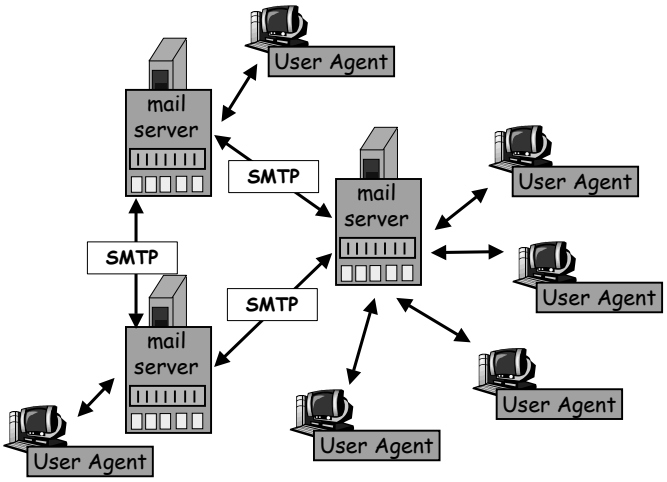
1.23

Posta elettronica

- L'*e-mail*, o posta elettronica, è una delle applicazioni di rete attualmente più diffuse.
- L'applicazione di posta elettronica consente la visione/trasferimento asincrono di messaggi contenenti testo, *hyperlink*, immagini, suoni e video.
- L'asincronicità del servizio consiste nel fatto che
  - il messaggio è consegnato e posto in una casella di posta (*mailbox*) associata al destinatario, senza che questi debba essere presente.
  - Il destinatario può accedere e leggere la propria "posta" in qualunque momento possa e lo desideri.
- Pur essendo simile al servizio di "posta ordinaria", la posta elettronica è molto più veloce (su Internet quasi in tempo reale), più flessibile e molto meno costosa.

1.24

Il servizio di posta elettronica: Schema di principio



1.25

I componenti del sistema

- Server di posta (*Mail Server*):
  - ricevono i messaggi da consegnare;
  - smistano la posta nelle *mailbox* degli utenti locali;
  - inviano i messaggi dei propri utenti al server di destinazione corretto;
  - consentono agli utenti di accedere alle proprie *mailbox;*
  - Es. di applicativi: *sendmail*, *exim*, *qmail*, *postfix*, *exchange*.
- *User Agent*:
  - forniscono un'interfaccia agli utenti per le operazioni legate ai messaggi (lettura, scrittura, ...);
  - interagiscono con i server di posta per inviare/ricevere messaggi e gestire le *mailbox*;
  - es. di applicativi: *mail*, *pine*, *elm*, *Eudora*, *Outlook*, *Netscape Messenger*.

1.26

Architettura del sistema di posta elettronica

- L'utente utilizza il proprio *user agent* per collegarsi ad un *Mail Server* allo scopo di:
  - accedere alla propria *mailbox*;
  - inviare messaggi.
- In linea di principio, un *Mail Server* potrebbe anche risiedere sul PC locale dell'utente
  - In questo caso, però, il PC dovrebbe rimanere sempre acceso e collegato alla rete, in modo da poter ricevere in qualunque momento nuovi messaggi dagli altri server.
- L'utente è individuato da un URL:

  <utente>@<server> o **<utente>@<dominio>**
  - il server che fornisce il servizio è identificato interrogando il DNS responsabile del dominio in questione;
  - lo stesso server può fornire il servizio di posta per domini diversi.

1.27

Servizi e protocolli di posta elettronica

- La posta elettronica fornisce due tipologie di servizio agli utenti:
  - Il trasferimento vero e proprio dei messaggi;
  - L'accesso remoto alle *mailbox* degli utenti.
- I protocolli utilizzati per fornire queste tipologie di servizio sono rappresentati da:
  - **SMTP** (Simple Mail Transfer Protocol): definisce le modalità di trasferimento del messaggio fra Server di posta.
  - **POP** (Post Office Protocol), **IMAP** (Internet Message Access Protocol): due distinti protocolli che definiscono le modalità di accesso da parte degli *user agent* alle proprie *mailbox*.

1.28

Invio di un messaggio di Posta Elettronica (1)

- Nel momento in cui un utente desidera inviare un messaggio di posta elettronica, le operazioni che consentono di consegnare il messaggio sono le seguenti:
  - Lo *user agent* del mittente richiede il servizio di inoltro al proprio *Mail Server* utilizzando il protocollo SMTP.
  - Tale *Mail Server* consegna il messaggio ricevuto direttamente al server di destinazione, senza utilizzare nessun intermediario.
  - Il *Mail Server* di destinazione riceve il messaggio e, in alternativa:
    - » inserisce il messaggio nella *mailbox* dell'utente ;
    - » inoltra il messaggio verso una diversa destinazione (*forwarding*).

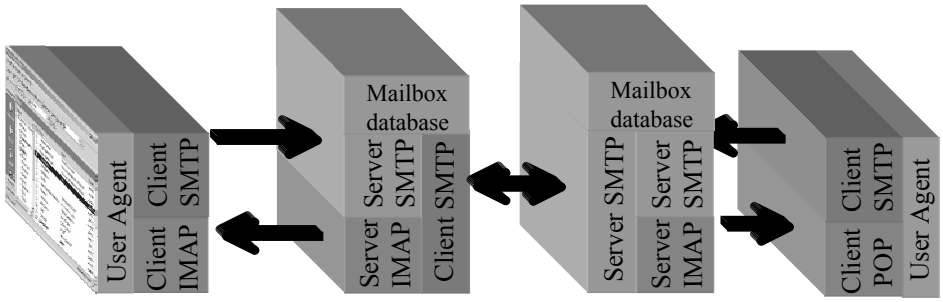
1.29

Invio di un messaggio di Posta Elettronica (2)

- In teoria lo *user agent* potrebbe anche inviare il messaggio direttamente al server dell'utente di destinazione.
- L'utilizzo del proprio *Mail Server* risulta più opportuno per gestire sia i permessi d'accesso, sia eventuali problemi nella consegna del messaggio (destinazione non raggiungibile o malfunzionante):
  - i server SMTP mantengono una coda (*message queue*) dei messaggi da inviare;
  - nel caso in cui la consegna non sia possibile, il messaggio rimane in coda e la consegna viene rimandata;
  - ad intervalli regolari (di solito 30 minuti), il mail server tenta l'invio dei messaggi ancora presenti nella coda;
  - dopo che il messaggio è rimasto in coda per un certo periodo (generalmente alcuni giorni) il server interrompe i tentativi e notifica il fallimento all'utente tramite *e-mail*.

1.30

Ciclo completo di trasferimento di un messaggio di posta



1. L'utente interroga il proprio server per l'invio del messaggio e/o per l'accesso alla propria *mailbox*:
   - La consegna del messaggio al proprio server avviene tramite il protocollo SMTP,
   - La ricezione può avvenire utilizzando il protocollo POP o IMAP.
2. Il messaggio in uscita dal Server viene trasferito, tramite una transazione SMTP, al Server di destinazione e da questi inserito nella *mailbox* dell'utente.
3. L'utente destinatario (quando lo ritiene opportuno) accede alla propria *mailbox* tramite i protocolli POP o IMAP e recupera/gestisce i messaggi in essa presenti.

1.31

SMTP - *Simple Mail Transfer Protocol*

- L'SMTP (*Simple Mail Transfer Protocol*), definito nella RFC 821, è il protocollo che definisce le modalità di dialogo tra le entità coinvolte nello scambio del messaggio.
- Il trasferimento del messaggio avviene tramite un'unica connessione TCP (sulla porta 25) tra il server che invia il messaggio e quello di destinazione.
- L'SMTP non utilizza server intermedi di transito.

1.32

SMTP *Simple Mail Transfer Protocol*

- I messaggi (intesi come PDU a livello di applicazione) scambiati sono di tipo testuale.
- Poiché l'SMTP è stato definito nei primi anni '80, presenta alcune caratteristiche ormai obsolete e limitative, come ad esempio, l'uso della codifica in codice ASCII a 7 bit come formato di testo.
- L'SMTP utilizza connessioni TCP permanenti: quindi ognuna di esse può essere utilizzata per il trasferimento di più messaggi tra due *mail server*.

1.33

SMTP

- L'SMTP è caratterizzato da una fase di *handshake,* dove il Client e il Server si scambiano, prima del trasferimento del vero e proprio messaggio, diverse informazioni.
- Generalmente il Client SMTP invia messaggi strutturati come segue:

  - *Command* (Comando) è il tipo di informazione o l'operazione che il Server SMTP deve compiere o ricevere, ad esempio:
    - » HELO, MAIL FROM, RCPT TO, DATA, QUIT.
  - *Argument* (Argomento) è l'eventuale informazione aggiuntiva richiesta dall'istruzione di *command*.
  - SP = "spazio", CRLF = Carriage Return Line Feed.

1.34

SMTP

- Il server SMTP generalmente risponde al Client con messaggi così strutturati:

- Dove:
  - *Code* (Codice): è un codice che identifica l'esito del messaggio inviato dal Client o lo stato del Server;
  - *Comment* (Commento): è un campo opzionale dove il Server può fornire un messaggio esplicativo del codice.

1.35

Formato del messaggio

- Il formato del messaggio di posta elettronica è definito nella specifica RFC 822.
- Il messaggio risulta composto da due parti:
  - **intestazione**, consiste in una parola chiave seguita dai relativi argomenti:
    - » i campi sono definiti dall'RFC 822;
    - » consente di includere informazioni accessorie al messaggio: nome del mittente (*From*), data di invio (*Date*), priorità (*Priority*), server SMTP attraversati, ecc.;
    - » alcune informazioni sono inserite durante la composizione del messaggio, altre dai server SMTP attraversati dallo stesso.
  - **contenuto** del messaggio o *entity-body*:
    - » contiene il messaggio vero e proprio utilizzando la codifica ASCII a 7 bit.

1.36

POP3 - *Post Office Protocol vers. 3*

- Il POP3 è definito nella RFC 1939.
- Il POP3 è un protocollo di accesso estremamente semplice ma con funzionalità limitate.
- Gli *user agent* implementano il lato *client* di questo protocollo, mentre i mail server implementano quello s*erver*.
- Il POP3 utilizza come protocollo di trasporto il TCP. Le connessioni sono sempre aperte dal client POP3 sulla porta numero 110 del server.

1.37

POP3

- A connessione TCP avvenuta, il POP3 agisce in tre fasi distinte:
  - **Autorizzazione**: l'utente invia la *login* (*username* + *password*) al Server.
  - **Transazione**: l'utente "scarica" i messaggi, li può contrassegnare per l'eliminazione (un *mail reader* può essere configurato per *download-and-delete* o per *download-and-keep*) e può ottenere delle statistiche sulla posta.
  - **Aggiornamento**: questa fase avviene dopo che il Client ha inviato il comando "quit"; concludendo la transazione il Server elimina tutti i messaggi che erano stati contrassegnati per l'eliminazione.
- La sintassi dei comandi, come ***list***, ***retr***, ***dele*** e ***quit***, e i relativi codici di risposta utilizzati nelle transazioni Client-Server POP3 è definita nella RFC 1939.

1.38

IMAP - *Internet Mail Access Protocol*

- L'**IMAP** (***Internet Mail Access Protocol***), definito dalla RFC 2060, è un protocollo di accesso ai Server di posta elettronica con molte più caratteristiche e, quindi, più complesso del POP3.
- I server IMAP usano connessioni TCP sulla porta 143.
- L'IMAP consente agli utenti di creare le proprie cartelle di posta sul server e di è gestirle come se fossero locali.
- In particolare consente di:
  - creare e mantenere cartelle di messaggi all'interno della *mailbox* sul server;
  - ricercare messaggi secondo criteri specifici nelle cartelle remote.

1.39

IMAP

- Una delle cause che rendono l'IMAP molto più complesso del POP3 è data dalla necessità, da parte del Server, di memorizzare e aggiornare una gerarchia di cartelle per ogni utente.
- Un'altra importante caratteristica di questo protocollo è data dalla possibilità di ottenere anche solo alcune parti (componenti) dei messaggi.
- Per esempio, uno *user agent* che accede al *Mail Server* utilizzando IMAP è in grado di estrapolare la sola intestazione dei messaggi o una singola componente di un messaggio MIME "multiparte".

1.40

IMAP

- Una sessione IMAP consiste di:
  - una connessione client-server,
  - una fase di *handshake*,
  - interazioni Client-Server,
  - chiusura della connessione.
- Le interazioni IMAP sono strutturalmente simili ma più complesse di quelle POP (comando del Client + risposta del Server).

1.41

HTTP

- Oggi molti utenti utilizzano il servizio e-mail tramite Web, ossia il Mail Server viene realizzato estendendo opportunamente le funzionalità di un *Web-Server* tramite script o programmi CGI (*Common Gateway Interface*).
- In questo modo è possibile usare come *User Agent* un normale *Web-Browser* e accedere alla propria *mailbox* attraverso transazioni HTTP.
- In questo caso, quindi, il *Web-Browser* opera come interfaccia ed usa l'HTTP (invece di MIME o POP) per comunicare con un *Mail -Web-Server*, che crea pagine dinamiche tramite le quali gli utenti gestiscono la posta.
- Il *Mail-Web-Server* usa comunque SMTP per comunicare con gli altri Mail-Server (Web o meno).

1.42

MIME - *Multipurpose Internet Mail Extension*

- Il MIME (*Multipurpose Internet Mail Extension*) è un'estensione della struttura RFC 822 che elimina alcune limitazioni presenti nello schema SMTP/RFC 822:
  - SMTP non può trasferire direttamente file eseguibili o comunque binari.
  - SMTP non può trasmettere i dati di un testo che comprenda caratteri di una lingua nazionale specifica (presenti in codici a 8 bit o superiori).
  - I Server SMTP possono rifiutare messaggi che superano una certa dimensione.
  - Alcune realizzazioni di SMTP non aderiscono completamente allo standard definito nella RFC 821.

1.43

MIME- *Multipurpose Internet Mail Extension*

- La specifica MIME comprende le seguenti caratteristiche:
  - Sono stati definiti 5 nuovi campi nell'intestazione del messaggio oltre a quelli già presenti ed ereditati dalla RFC 822.
  - Sono stati definiti nuovi formati di contenuto che supportano la multimedialità nella posta elettronica.
  - Sono state definite le codifiche di trasferimento che abilitano la conversione di ogni formato in una forma protetta da eventuali alterazioni introdotte dal sistema di posta.

1.44

MIME: I campi dell'intestazione

- I 5 nuovi campi di intestazione introdotti dal MIME sono:
  - **MIME-*Version***: deve assumere un valore 1, che indica che il messaggio è conforme agli RFC.
  - ***Content-Type***: Descrive il tipo di dati contenuti nell'*entity-body*.
  - ***Content-Transfer-Encoding***: Indica il tipo di codifica utilizzata per rappresentare l'*entity-body* in un modo compatibile al trasporto SMTP.
  - ***Content-ID***: Serve ad identificare univocamente le entità MIME in diversi contesti.
  - ***Content-Description***: Una descrizione in testo non codificato dell'oggetto contenuto nell'*entity-body*.

1.45

MIME: L'intestazione *Content-Type*

Type/Subtype

Type	*Subtype*	*Type*	*Subtype*
text	plain HTML	multipart	mixed
image	jpeg		parallel
	gif		alternative
video	mpeg		digest
audio	basic 32kadpcm	message	RFC822
application	Postscript Adobe		partial
	Dati binari a 8 bit		external body

1.46

MIME: L'intestazione *Context-Transfer-Encoding*

- Lo standard MIME definisce attualmente due metodi per la codifica dei dati:
  - ***quoted-printable***;
  - ***base64***.
- In realtà, il campo *Context-Transfer-Encoding* può assumere ulteriori 4 valori:
  - ***7bit***: i dati sono rappresentati da brevi linee di caratteri ASCII;
  - ***8bit***: possono essere presenti caratteri ASCII codificati a 8 bit (bit più significativo a 1);
  - ***binary***: oltre a essere presenti caratteri non-ASCII le linee potrebbero non essere brevi quanto necessario per il trasporto tramite SMTP;
  - ***x-token***: utilizzato per codifiche proprietarie.
- I valori *7bit*, *8bit* e *binary* indicano che non è stata applicata nessuna codifica specifica (da MIME), ma forniscono alcune indicazioni di massima sulla natura dei dati.

1.47

MIME: L'intestazione *Context-Transfer-Encoding*
Codifiche di trasferimento

- Il valore ***quoted-printable*** indica l'uso di una particolare codifica volta a mantenere la leggibilità di testi composti principalmente (in origine) da caratteri ASCII.
- La codifica di trasferimento ***base64*** è comunemente impiegata per i dati binari generici, in modo da renderli trasparenti alle elaborazioni da parte dei programmi di trasporto della posta elettronica.
- Questa tecnica codifica i dati binari in ingresso in caratteri "visualizzabili".

1.48

MIME: L'intestazione *Context-Transfer-Encoding*

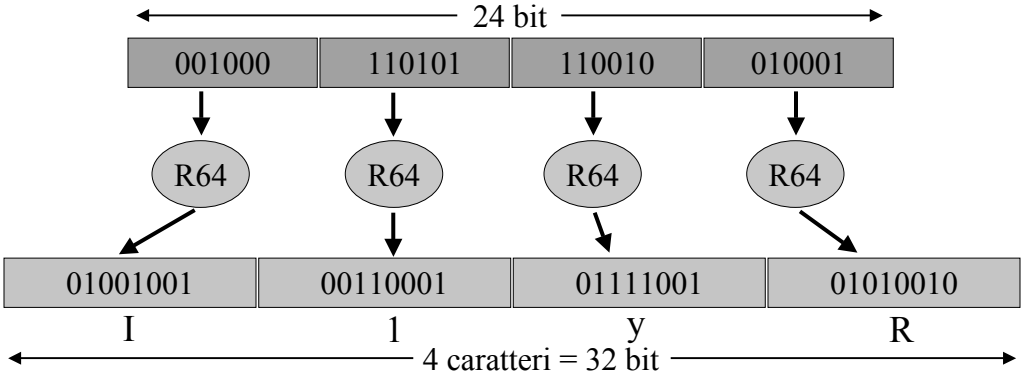
Codifiche di trasferimento: *Base64*

- La codifica ***Base64*** presenta le seguenti importanti caratteristiche:
  - L'intervallo dei valori forniti dalla funzione di codifica è un insieme di caratteri universalmente rappresentabile.
  - L'insieme dei caratteri è formato da 65 caratteri visualizzabili, uno dei quali serve da riempimento "**=**". Avendo 64 caratteri disponibili, ognuno di essi viene usato per rappresentare 6 bit del contenuto originario.
  - Nessun carattere di controllo è incluso nell'insieme.
  - Il carattere "-" non è usato perché nel formato RFC 822 ha un significato specifico.

1.49

MIME: L'intestazione *Context-Transfer-Encoding*

Codifiche di trasferimento: *Base64*



Valore 6 bit	codifica	Valore 6 bit	codifica
000000	A	000011	D
000001	B	000100	E
000010	C	000101	F

1.50

Livello di Applicazione

HyperText Transfer Protocol (http)

1.51

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"

1.52

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 .

1.53

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.....

1.54

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.

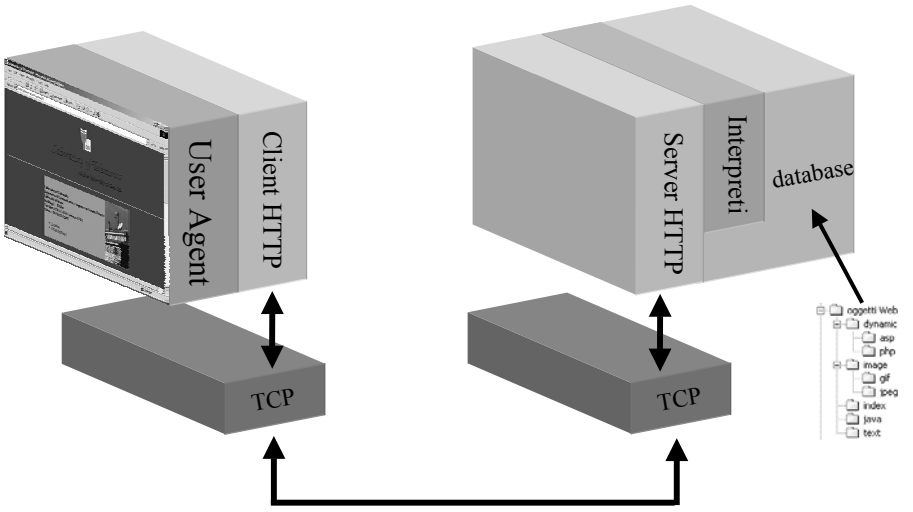
1.55

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.

1.56

Server Web e Browser



1.57

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".

1.58

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

1.59

Tipi di connessione

- 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.
- 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.

1.60

Tipi di connessione

- 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.
- D'altra parte l'utilizzo della connessione in modalità non 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

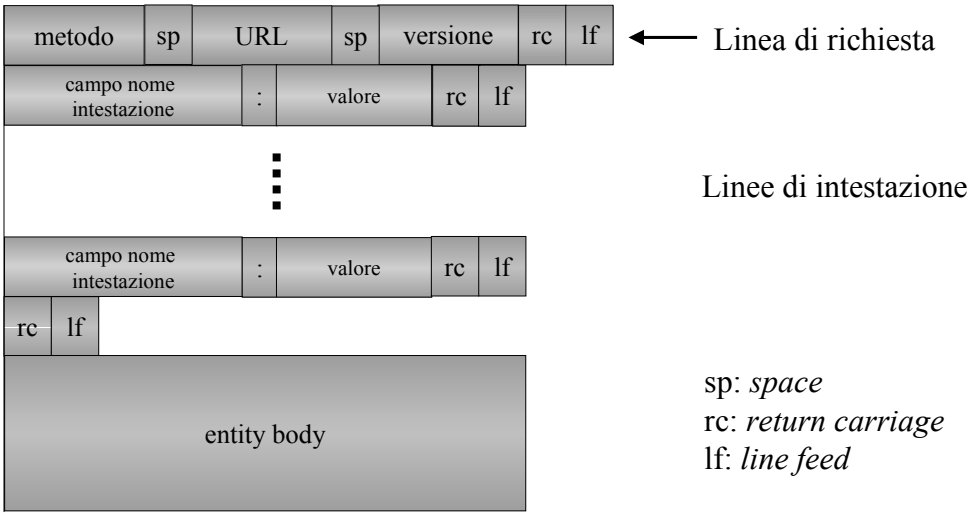
1.61

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*).

1.62

Il messaggio HTTP di richiesta



1.63

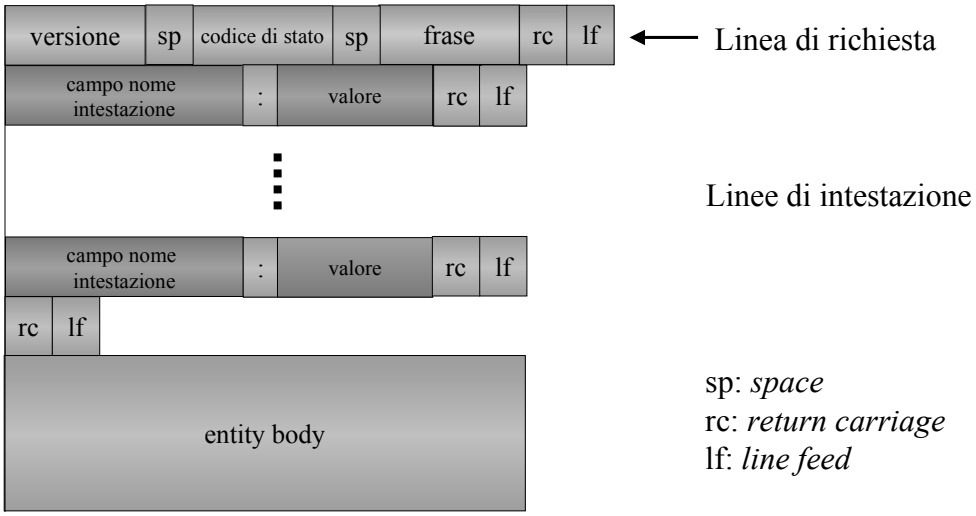
Il messaggio HTTP di richiesta

- 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

1.64

Il messaggio HTTP di risposta



1.65

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) Linee di
Last-Modified: Mon, 22 Jul 2003 15:43:23 GMT intestazione
Content-Length:14287
Content-Type: text/html

[dati.....] *Entity Body*

1.66

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.

1.67

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.*

1.68

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.

1.69

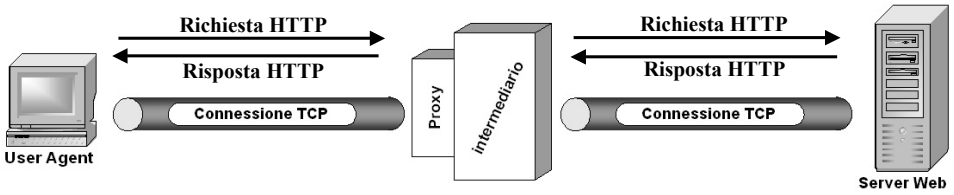
Cookie

- 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

1.70

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



1.71

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.

1.72

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.

1.73

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)

1.74