Hypertext Transfer Protocol

Hypertext Transfer Protocol [RFC7230-7235]

"The Hypertext Transfer Protocol (HTTP) is an application-level protocol for distributed, collaborative, hypermedia information systems. It is a generic, stateless, protocol which can be used for many tasks beyond its use for hypertext, such as name servers and distributed object management systems, through extension of its request methods, error codes and headers. A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred."

Caratteristiche salienti

- Protocollo applicativo orientato ai sistemi iper-mediali
  - Ma non limitato ad essi
- Protocollo senza stato e generale
- Supporta la tipizzazione e la negoziazione della rappresentazione dei dati
  - Consente eterogeneità delle implementazioni

Evoluzione del protocollo

- 1991 v0.9 è la prima versione documentata
- 1995 v1.0 Prima versione ratificata [RFC1945]
- 1997 v1.1 Estensioni del protocollo [RFC2068]
- 2012 Google introduce SPDY
- 2014 v1.1 Revisione e chiarificazione [RFC7230-7235]
- 2015 HTTP/2 [RFC7540]
- 2015 QUIC [draft-hamilton-early-deployment-quic-00]

Architettura

- HTTP è stato creato per il WWW
- La sua evoluzione ha assecondato le richieste di scalabilità di un sistema di dimensione mondiale
- Alla base prevede uno scambio di messaggi tra client e server
  - La comunicazione avviene per mezzo di un protocollo di trasporto affidabile -> **TCP**

Schema di comunicazione

- Il server è un programma che accetta connessioni e attende richieste HTTP. Serve le richieste HTTP produce risposte HTTP
  - Il termine **Origin Server** indica il programma che è in grado di generare la risposta autoritativa per una determinata risorsa
- Il client è un programma che stabilisce una connessione verso un server con l'obiettivo di inviare una o più richieste HTTP
  - Anche chiamato **User Agent**

Eterogeneità nelle implementazioni

- Il client non è sempre un browser Web
  – elettrodomestici, gadget, …
- Il server non è sempre un grande sito pubblico
  – domotica, dispositivi di rete configurabili, semafori, …
- L'utilizzo del termine User Agent non implica la presenza di un essere umano che interagisce col software a tempo della richiesta

Risorsa

- L'obiettivo di una richiesta HTTP è denominato "risorsa"
- HTTP non definisce la natura delle risorse, ma si limita ad offrire una interfaccia per l'interazione
- Ogni risorsa è identificata da un URI

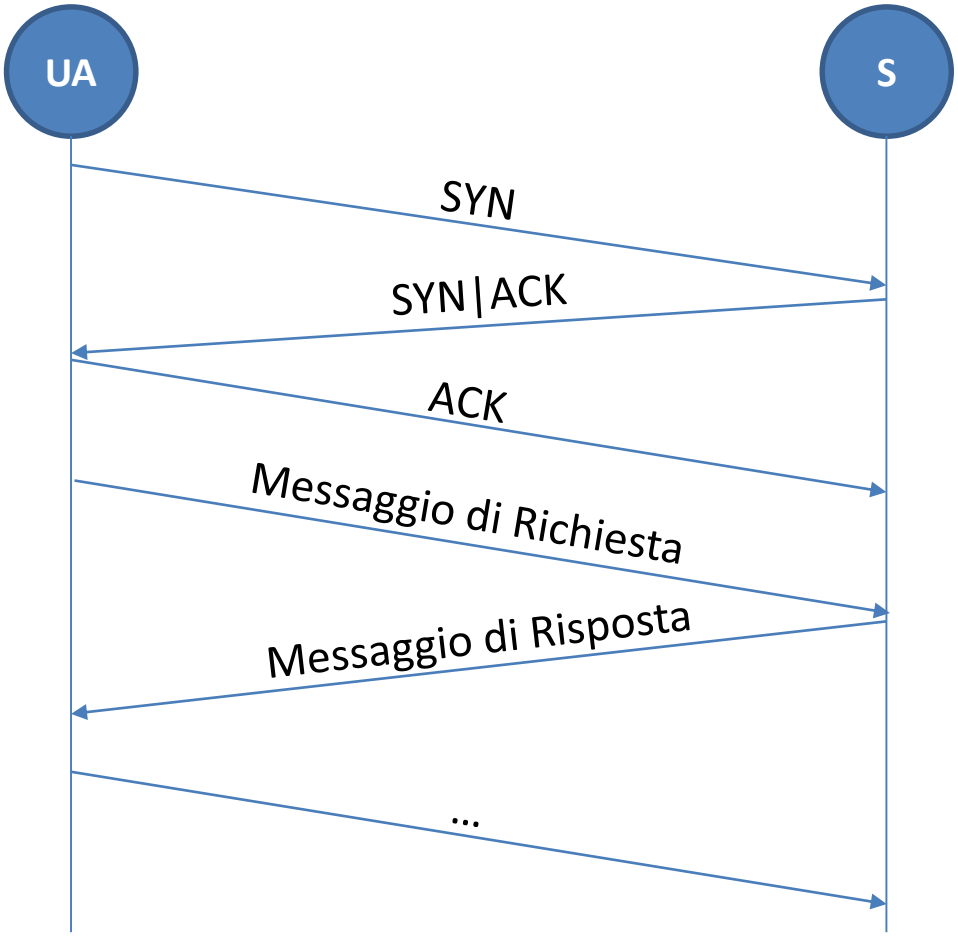
Rappresentazione

- Le risorse sono eterogenee, ma c'è bisogno di avere un'astrazione per la loro gestione
- Una rappresentazione è un insieme di informazioni che intendono riflettere lo stato passato, quello corrente o desiderato di una risorsa in un formato trasmissibile tramite il protocollo
- Consiste in metadati e uno stream di dati

Rappresentazione (cont.)

- Un origin server può ottenere o essere in grado di generare più rappresentazioni della stessa risorsa
- In base ad un algoritmo viene selezionata la rappresentazione applicabile alla singola richiesta
  - Solitamente in base ad una negoziazione

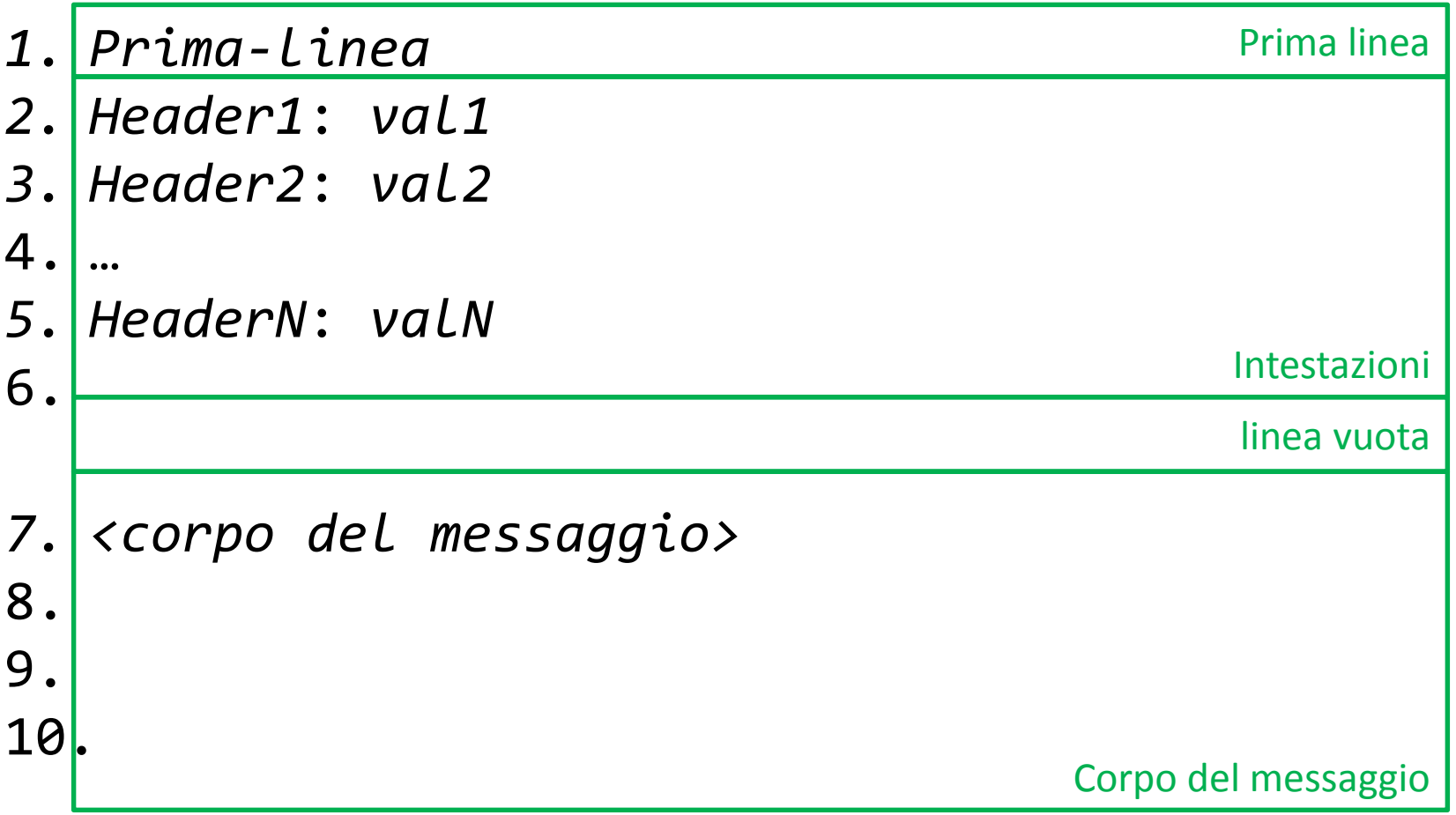
Comunicazione client-server



Sintassi dei Messaggi

- Tutti messaggi sono composti da una sezione di intestazione e da un payload
- La sezione di intestazione è organizzata per linee:
  - La prima indica l'operazione desiderata (richiesta) o il risultato ottenuto (risposta)
  - Successivamente sono riportati dei campi aggiuntivi di intestazione (*header*)
  - Una linea vuota separa la sezione di intestazione dal payload del messaggio
- Nell'intestazione, le linee sono terminate dalla sequenza CRLF (|0a 0d|)

Formato del messaggio HTTP



Messaggio di richiesta HTTP

- Il messaggio di richiesta inizia con una **request-line**
  - Contiene **metodo**, l'**URI** della risorsa, e la **versione** del protocollo
- Seguono un numero variabile di campi **header**, uno per linea.
  - Modificatori, informazioni del client, metadati relativi alla rappresentazione, …
- Segue una **linea vuota** indica la fine dell'intestazione della richiesta
- Se presente, sarà riportato il **contenuto** della richiesta

Esempio di richiesta HTTP

```
1. GET /hello.txt HTTP/1.1                          request line
2. User-Agent: curl/7.16.3 libcurl/7.16.3
3. Host: www.example.com
4. Accept-Language: en, mi                          header
5.                                                  linea vuota
```

Risposta HTTP

- Il server risponde alla richiesta del client inviando un messaggio di risposta HTTP
- Il messaggio di risposta è inviato sulla stessa connessione dal quale è stata ricevuta la richiesta

IDENTIFICAZIONE DELLE RISORSE

Uniform Resource Identifiers

- Gli URI sono usati per identificare le risorse
  - Specificare il target delle richieste
  - Reindirizzamenti
  - Definire relazioni tra entità
- Gli URI sono definiti nell'RFC3986
- Non è uno standard specifico per l'HTTP

Uniform Resource Indentifier [RFC3986]

- Definisce una sintassi uniforme per riferimenti a risorse di qualsiasi tipo
  - Estendibile: Permette l'introduzione di nuovi tipi di identificatori
  - Uniforme: Gli stessi identificatori possono essere usati in contesti diversi (pagine HTML, header HTTP, stampa su carta, …)
- Una risorsa è qualsiasi cosa possa essere identificata con un URI
  - Non necessariamente accessibile tramite Internet
- Distingue una risorsa da altre nello stesso contesto

Interazione con gli URI

- L'URI non è usato solamente per accedere alla risorsa
- Dato un URI, un sistema può eseguire una varietà di operazioni sulla risorsa
  - Accesso
  - Aggiornamento
  - Rimpiazzamento
  - Trovarne gli attributi
  - …
- Le operazioni disponibili sono definite dai protocolli che fanno uso degli URI

Identificatori gerarchici

- La sintassi è organizzata gerarchicamente, con componenti elencati in ordine dal più al meno significativo
- Il significato dei vari componenti è definito dalla specifica in esame

Sintassi

scheme ":" hier-part ["?" query] ["#" fragment]

hier-part = authority path

- I caratteri che fanno parte del set ASCII possono essere scritti direttamente nella loro forma stampabile
- Altrimenti è possibile rappresentare i caratteri utilizzando il "percent encoding"
  - Il carattere deve essere rappresentato nella sua forma UTF-8
  - Ogni ottetto viene rappresentato nella forma "%HH" dove HH è la rappresentazione esadecimale (case insensitive) dell'ottetto

Scheme

- Lo **scheme** si riferisce ad una specifica per assegnare identificatori alle risorse
- È case insensitive anche se la forma canonica è minuscola e dovrebbe essere quella utilizzata
- Gli scheme disponibili devono essere registrati (IANA) e fare riferimento ad una specifica nel quale si definiscono la semantica di altre parti dell'URI
- Alcuni esempi
  – http, file, ftp, mailto, ldap, …

Authority

[userinfo "@"] host [":" port]

- È preceduto da "//" ed è seguito da "/", "?" o "#" oppure dalla fine dell'URI
- Se lo scheme lo prevede, il campo authority definisce l'entità responsabile delegata alla definizione del namespace
- Se lo scheme lo prevede è possibile specificare delle informazioni su come ottenere l'autorizzazione ad accedere la risorsa

Authority (cont.)

[userinfo "@"] host [":" port]

- L'host può essere un
  - Indirizzo IPv6 racchiuso tra "[" e "]", es [::1]
  - Indirizzo IPv4 in forma decimale puntata, es 127.0.0.1
  - Un host identificato in un registry, es. DNS
- Il sottocomponente port è opzionale
  - Il tipo di porta (TCP, UDP, SCTP) è definito dallo scheme
  - Lo scheme può definire una porta di default
  - Va omesso se vuoto o uguale al default per lo scheme

Path

- Contiene delle informazioni gerarchiche che identificano la risorsa nel sistema di naming adottato
  - sequenza di segmenti separati da "/"

- Ogni segmento può specificare parametri
  - name;p1=v1
  - name,1.1

- Esempio
  - /path/name1;param1;p2=v2;p3/name2;par=val
- Il path termina al primo '?', '#' o al termine dell'URI

Query

hier-part ["?" query] ["#" fragment]

- Inizia al primo '?' ed è terminato da un eventuale '#' o dalla fine dell'URI
- Contiene delle informazioni non gerarchiche che identificano la risorsa nel sistema di naming adottato

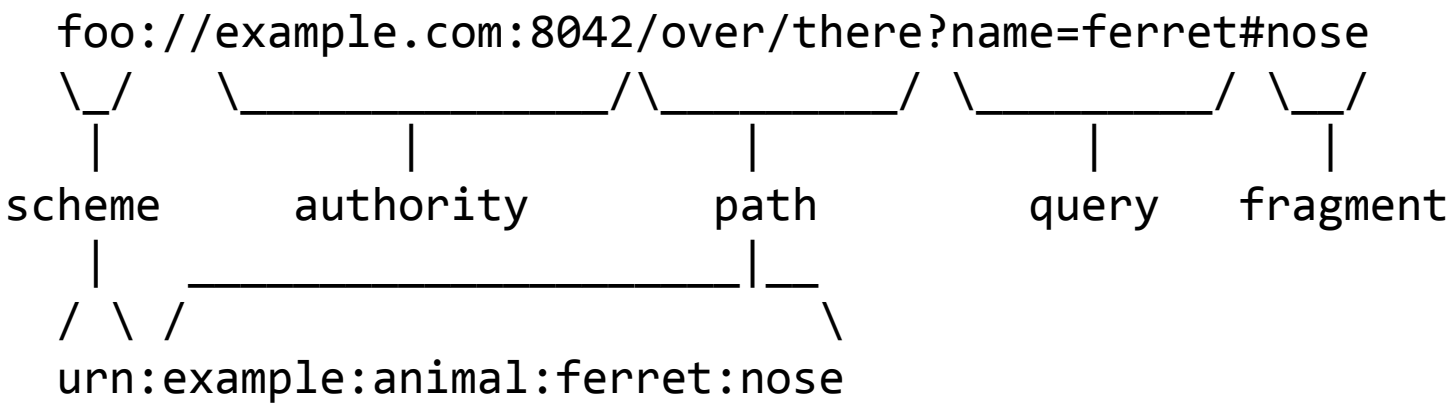
Fragment

hier-part ["?" query] ["#" fragment]

- Permette l'identificazione indiretta di una risorsa secondaria usando come riferimento la risorsa primaria identificata dal resto dell'URI
  – Ad esempio un sottoinsieme della risorsa principale

- Inizia al primo '#' e termina col termine dell'URI

Esempi


foo://example.com:8042/over/there?name=ferret#nose
scheme
authority
path
query
fragment
urn:example:animal:ferret:nose

Alcuni URI validi

- ftp://ftp.is.co.za/rfc/rfc1808.txt
- http://www.ietf.org/rfc/rfc2396.txt
- ldap://[2001:db8::7]/c=GB?objectClass?one
- mailto:John.Doe@example.com
- news:comp.infosystems.www.servers.unix
- tel:+1-816-555-1212
- telnet://192.0.2.16:80/
- urn:oasis:names:specification:docbook:dtd:xml:4.1.2

URI HTTP

"http:" "//" authority path-abempty ["?" query] ["#" fragment]

- Lo scheme è **http** ed implica l'utilizzo di TCP
- Il campo **authority** specifica l'origin server per un URI HTTP
  - Include l'identificativo host dell'origin server (case insensitive)
  - Opzionalmente può contenere un numero di porta
- La parte **path-abempty** è un percorso gerarchico
  - Può essere vuoto
  - Altrimenti incomincia con "/"

Normalizzazione degli URI HTTP

- I seguenti URI sono equivalenti
  - http://example.com:80/~smith/home.html
  - http://EXAMPLE.com/%7Esmith/home.html
  - http://EXAMPLE.com:/%7esmith/home.html

MESSAGGI

Messaggi HTTP

- Due tipi: Richiesta e Risposta
  - Si distinguono in base alla prima linea
  - Hanno la stessa sintassi
- All'interno dei messaggi vengono utilizzati riferimenti alle risorse
  - Per selezionare una risorsa oggetto della richiesta
  - Per reindirizzamenti
- I riferimenti sono implementati secondo un altro standard -> URI

Request line HTTP

method SP request-target SP HTTP-version CRLF

- **method** è l'operazione da invocare sulla risorsa in oggetto
- **request-target** identifica la risorsa (URI) oggetto della richiesta
- **http-version** versione supportata dal client che invia il messaggio

Metodo della richiesta HTTP

- Il metodo rappresenta la fonte primaria di semantica
  - scopo della richiesta
  - quale risultato il client si attende
- La semantica della richiesta può essere ulteriormente specificato tramite alcuni campi dell'intestazioni presenti nella richiesta

Metodi GET e HEAD

- GET
  - Richiede il trasferimento una rappresentazione corrente della risorsa in oggetto
  - L'identificativo della risorsa non è obbligatoriamente un percorso nel file system

- HEAD
  - come GET ma solo la sezione di intestazione
  - Serve per ottenere gli stessi metadati che sarebbero stati ritornati per una GET ad eccezione di quelli relativi al payload

Metodo POST

- Eseguire una gestione specificata dalla risorsa sul payload della richiesta

- Esempi
  - Dati in un modulo da compilare
  - Messaggio in un forum di discussione
  - Creare una nuova risorsa che non ha ancora un identificativo dell'origin server
  - Aggiungere dati alla rappresentazione esistente di una risorsa

Metodi PUT, DELETE e PATCH

- PUT
  - Sostituire tutte le rappresentazioni attuali della risorsa specificata con il payload della richiesta
  - Non implica che una richiesta GET sulla risorsa sia possibile
- DELETE
  - Rimuovere tutte le rappresentazioni correnti della risorsa
- PATCH [RFC5789]
  - Si richiede l'applicazione di modifiche alla risorsa

Metodi OPTIONS e TRACE

- OPTIONS
  - Richiede al server quali operazioni sono permesse su una determinata risorsa
- TRACE
  - Chiede al server di inviare la richiesta come ricevuta

Response line HTTP

HTTP-version SP status-code SP reason-phrase CRLF

- **HTTP-version** è la versione del protocollo supportata dal server
- **status-code** è un intero di 3 cifre decimali che descrive il risultato del tentativo del server di capire e soddisfare la richiesta del client
- **reason-phrase** fornisce una descrizione testuale associata al codice di ritorno numerico e dovrebbe essere ignorata

Status code

- I codici sono estendibili e un client non deve necessariamente conoscere tutti i codici
  - Deve però riconoscere la categoria del codice, indicata dalla cifra più significativa

Status code - classi

- **1xx** Richiesta ricevuta e il processing continua
- **2xx** la richiesta è stata ricevuta, interpretata e accettata
- **3xx** sono necessarie ulteriori azioni al fine di completare la richiesta
- **4xx** la richiesta ha una sintassi errata o non può essere soddisfatta
- **5xx** Il server non è riuscito a soddisfare una richiesta apparentemente valida

Status code - 1xx, 2xx

- 1xx
  - 100 (Continue)
  - 101 (Switching Protocols)
- 2xx
  - 200 (OK)
  - 201 (Created)
  - 202 (Accepted)
  - 203 (Non-Authoritative Information)
  - 204 (No Content)
  - 205 (Reset Content)
  - 206 (Partial Content)

Status code - 3xx

- 300 (Multiple Choices)
- 301 (Moved permanently)
- 302 (Found)
- 303 (See Other)
- 304 (Not Modified)
- 305 (Use Proxy)
- 307 (Temporary Redirect)

Status code - 4xx

- 400 (Bad Request)
- 401 (Unauthorized)
- 402 (Payment Required)
- 403 (Forbidden)
- 404 (Not Found)
- 405 (Method Not Allowed)
- 406 (Not Acceptable)
- 407 (Proxy Authentication Required)
- 408 (Request Timeout)
- 409 (Conflict)
- 410 (Gone)
- 411 (Length Required)
- 412 (Precondition Failed)
- 413 (Payload Too Large)
- 414 (URI Too Long)
- 415 (Unsupported media type)
- 416 (Range Not Satisfiable)
- 417 (Expectation Failed)
- 426 (Upgrade Required)

Status code - 5xx

- 500 (Internal Server Error)
- 501 (Not Implemented)
- 502 (Bad Gateway)
- 503 (Service Unavailable)
- 504 (Gateway Timeout)
- 505 (HTTP Version Not Supported)

Informational 1xx

- Risposte temporanee che comunicano lo stato della connessione o il progresso di una richiesta

- Esempio
  - 100 (Continue)
  - 101 (Switching Protocols)

200 (OK)

- La richiesta ha avuto successo

- La risposta sarà dipendente dal metodo
  - **GET** una rappresentazione della risorsa
  - **HEAD** stessa rappresentazione di GET ma senza il contenuto
  - **POST** una rappresentazione dello stato o il risultato dell'azione
  - **PUT, DELETE** una rappresentazione dello stato dell'azione
  - **OPTIONS** una rappresentazione delle opzioni di comunicazione
  - **TRACE** una rappresentazione del messaggio come ricevuto dal server
  - **CONNECT** nessun payload

300 (Multiple Choices)

- Utilizzato per la negoziazione reattiva
- Il payload del messaggio di risposta include informazioni su rappresentazioni alternative della risorsa

301 (Moved Permanently)

- La risorsa richiesta è stata spostata sotto un nuovo URI
  – Header "Location" della risposta specifica il nuovo URI
- I prossimi riferimenti dovrebbero utilizzare direttamente al nuovo URI
- Per motivi storici alcuni user agent potrebbero cambiare il metodo da POST a GET nelle richieste successive

302 (Found)

- Indica che la risorsa risiede temporaneamente ad un URI differente
  – Header "Location" della risposta specifica il nuovo URI

- Lo user agent dovrebbe in futuro continuare ad utilizzare l'URI originale perché il reindirizzamento può essere temporaneo

- Uno user agent potrebbe cambiare da POST a GET la risorsa

303 (See Other)

- Il server redirige lo user agent su un altro URI (specificato nell'header Location) che fornisce una risposta (indirettamente) alla richiesta originaria
  – Non c'è una rappresentazione utile della risorsa

- Ad esempio il risultato di una richiesta POST potrebbe indicare l'URI al quale ottenere la risorsa

307 (Temporary Redirect)

- La risorsa richiesta è raggiungibile temporaneamente tramite un URI diverso (specificato nell'header Location) e lo user agent non deve modificare il metodo della richiesta

- Analogo a 302

4xx - Client Error

- 400 (Bad Request)
  - Il server non può o non vuole processare la richiesta a causa di un errore da parte del client
- 401 (Unauthorized)
  - Invita il client ad inviare informazioni di autenticazione
- 403 (Forbidden)
  - Il server ha compreso la richiesta ma si rifiuta di autorizzarla. Può specificare la ragione nel payload

4xx - Client Error (cont.)

- 404 (Not Found)
  - Non è stata trovata una rappresentazione della risorsa o il server non vuole
- 405 (Method Not Allowed)
  - Il metodo specificato è conosciuto ma non è supportato dalla risorsa
  - Includerà un header "Allow" contenente l'elenco dei metodi ammessi
- 406 (Not Acceptable)
  - La risorsa richiesta non ha una rappresentazione accettabile per lo user agent

5xx - Server Error

- 500 (Internal Server Error)
  - Si è verificato un errore lato server
- 501 (Not Implemented)
  - Il server non dispone delle funzionalità necessarie per soddisfare la richiesta
- 502 (Bad Gateway)
  - Il proxy o gateway ha ricevuto una risposta non valida

5xx - Server error (cont.)

- 503 (Service Unavailable)
  - Impossibile soddisfare la richiesta per causa temporanea (manutenzione, sovraccarico)
  - Può includere un header "Retry-After"

- 504 (Gateway Timeout)
  - Il proxy (o gateway) non ha ricevuto una risposta entro un determinato tempo

GESTIONE DELLA RICHIESTA

Negoziazione del contenuto

- Proattiva
  - Il server seleziona la rappresentazione basandosi sulle preferenze espresse dallo user agent
- Reattiva
  - Il server fornisce al client una lista di rappresentazioni tra le quali scegliere
- Contenuto condizionale
  - La rappresentazione consiste di molteplici parti che sono utilizzate in base a parametri dello user agent
- Contenuto attivo
  - La rappresentazione contiene uno script che farà richieste più specifiche in base alle caratteristiche dello user agent

Negoziazione Proattiva

- Non aggiunge round-trip
- La rappresentazione selezionata è una *best-guess*
  - Impossibile per il server scegliere la miglior rappresentazione per l'utente
- Ogni richiesta include le preferenze
- Complica l'implementazione di un origin server
- Limita la riutilizzabilità delle risposte nelle cache condivise

Negoziazione Reattiva

- Il server fornisce una rappresentazione preliminare nel quale elenca possibili rappresentazioni alternative alternative
- Lo user agent farà ulteriori richieste se preferisce

- Permette allo user agent di scegliere la rappresentazione migliore
- Introduce round-trip aggiuntivi

CAMPI INTESTAZIONE RICHIESTA

Header

- Gli header permettono alle parti di specificare meta-informazioni
- Gli header sono molto importanti ed influiscono nella gestione della richiesta, nella rappresentazione scelta della risorsa, nel modo di trasferire la risorsa stessa
- Esistono header standard, ma è possibile definirne nuovi (come per i metodi)

Header "Host"

- Fornisce l'host (e la porta) relativi alla risorsa richiesta
  - permette all'origin server di distinguere tra più host name serviti su un singolo indirizzo IP (**virtual hosting**)
  - Deve essere uguale alla parte authority di una richiesta in absolute-form

- **Obbligatorio** in HTTP/1.1

Controlli

- Sono quegli header che specificano come gestire la richiesta stessa
  - Cache-Control
  - Expect
  - Host
  - Max-Forwards
  - Pragma
  - Range
  - TE

Metadati sul payload della richiesta

- Content-Length
  - Definisce la lunghezza in byte del corpo della richiesta
- Content-Type
  - Definisce il tipo (internet media type) del corpo della richiesta
- Ad esempio
  - application/x-www-form-urlencoded
  - mulitpart/form-data

Header "Expect"

- Indica un insieme di comportamenti attesi da parte del client
  - Expect: 100-continue
  - Informa il ricevente che il mittente è in procinto di inviare un corpo del messaggio (grande) e vuole sincerarsi che il ricevente approvi tale operazione inviando una risposta 100 (Continue)
- Il client potrebbe ottenere un errore, esempio 417 (Expectation Failed)

Internet media types (tipi MIME)

- Specificano il tipo di una risorsa
  - type/subtype;parameter1=val1
- Ad esempio
  - text/html;charset=utf-8
- Registrati in un registro IANA
- Anche i charset sono registrati nel registro IANA

Header "Accept"

- Lo user agent specifica quali media type sono accettabili
  - Accept: text/plain; q=0.5, text/html, text/x-dvi; q=0.8, text/x-c
  - "text/html e text/x-c sono preferiti in egual modo. se non esistono utilizza text/x-dvi. Se a sua volta non esista usa la rappresentazione text/plain"

- Se nessuno dei tipi specificati è disponibile per la risorsa viene ritornato 406 (Not Acceptable)

Header "Accept" (cont.)

- I media range possono essere ridefiniti da media range più specifici
  – Quelli più specifici hanno precedenza

- Accept: text/*, text/plain, text/plain;format=flowed, */*
  1. text/plain;format=flowed
  2. text/plain
  3. text/*
  4. */*

Header "Accept-Charset"

- Indica la preferenza rispetto al set di caratteri da usare per codificare testo nel contenuto della risposta
  – Accept-Charset: iso-8859-5, unicode-1-1;q=0.8

- Il valore speciale "*" si applica tutti i charset non specificati

Header "Accept-Encoding"

- Indica quali content-coding sono accettabili nelle risposte
  - identity significa nessun coding
  - Accept-Encoding: gzip;q=1.0, identity; q=0.5, *;q=0

Header "Accept-Language"

- Indica le lingue naturali preferite per la risposta
  - A parità di qualità si guarda l'ordine
- Accept-Language: da, en-gb;q=0.8, en;q=0.7
  - Preferisco il danese, accetto l'inglese britannico e altri tipi di inglese
- Se assente implica l'accettazione di qualsiasi lingua nella risposta
- Se il server non è in grado di soddisfare la richiesta ignora l'header oppure (raro) ritorna 406 (Not Acceptable)

Header "Referer"

- Permette allo user agent di specificare dove è stato ottenuto l'URI per la risorsa richiesta
  – non deve includere il fragment e informazioni utente

- Referer: http://www.example.org/hypertext/Overview.html

- Se l'URI non è stato ottenuto da una risorsa con un proprio URI, l'header deve essere omesso (oppure "about:blank")

Header "User-Agent"

- Informazioni riguardanti il programma che ha generato la richiesta
- Formato da una lista di prodotti con commenti opzionali
- In ordine di importanza
- prodotto può contenere una versione
- Non deve contenere informazioni troppo dettagliate

HEADER DELLA RISPOSTA

Metadati sul payload della risposta

- Alcuni messaggi HTTP contengono una rappresentazione completa o parziale nel payload
- Lo scopo del payload è definito dal metodo

- Ci sono header che descrivono il payload (non la rappresentazione)
  - Content-Length
  - Content-Range
  - Trailer
  - Transfer-Encoding

Header "Content-Length", "Content-Type"

- Content-Length
  - Dimensione del payload in byte
- Content-Type
  - Tipo della rappresentazione della risorsa ritornata

Header "Content-Range"

- Il server che interpreta correttamente una Range Request risponderà con 206 (Partial Content)
  - la risposta comprende un header Content-Range
  - Content-Range: bytes 500-599/8000

- Se il numero di range specificato è maggiore di uno il messaggio avrà un Content-Type multipart/byteranges (vedi MIME)
  - Ogni parte avrà il suo Content-Range

Header "Transfer-Encoding"

- Un campo dell'intestazione che elenca quali modifiche sono state (o verranno) applicate al corpo del messaggio
  – Analogo al Content-Transfer-Encoding di MIME
- È una proprietà del messaggio e non della rappresentazione
- Alcuni valori: gzip, chunked

Header "Content-Type"

- Il campo Content-Type indica il media type della rappresentazione inclusa nel payload del messaggio

- Se non presente il default dovrebbe essere "application/octet-stream"

Header "Content-Encoding"

- Trasformazioni applicate alla rappresentazione
- Spesso usate per la compressione
  – es: compress, deflat, gzip
- Nel caso siano combinate devono essere elencate nell'ordine giusto
- Se un origin server non è in grado di generare una rappresentazione secondo il content-coding richiesta dovrebbe ritornare una 451

Header "Content-Language"

- Descrive i linguaggi (naturali) dei possibili utilizzatori della rappresentazione
  - Non è l'elenco delle lingue utilizzate

Header "Content-Location"

- Permette al mittente di specificare un identificativo della risorsa insieme alla rappresentazione della stessa contenuta nel payload
- Se l'URI specificato in Content-Location differisce significa che quello sarebbe un URI più appropriato (per la specifica rappresentazione)
- Se una risposta 201 (Created) ha lo stesso URI di Content-Location significa che il payload è una rappresentazione della risorsa appena creata

Control Data

- Supplementano lo satus code

- Header
  - Age
  - Cache-Control
  - Expires
  - Date
  - Location
  - Retry-After
  - Vary
  - Warning

Header "Date", "Location"

- Date
  - Rappresenta la data e l'ora nel quale il messaggio di risposta è stato generato
  - Date: Tue, 15 Nov 1994 08:12:31 GMT

- Location
  - Riferimento ad una risorsa correlata alla risposta
  - Può essere sia assoluto che relativo
  - Usato nei reindirizzamenti (3xx)

HTTP senza stato

- HTTP è stateless
  - La connessione viene chiusa alla fine della risposta (1.0)
  - La connessione viene chiusa dopo un timeout (1.1)
  - La connessione invia richiesta da tanti utenti (proxy)

- Non è possibile collegare tra loro richieste HTTP
  - Sessione di navigazione dell'utente

HTTP + stato

- Si utilizzano degli header aggiuntivi per la gestione dello stato
- Il server invia informazioni di stato allo user agent
- Lo user agent riporterà le informazioni di stato nelle richieste

Header "Cookie" e "Set-Cookie"

- Il cookie si presenta come una variabile
  - Nome + valore
- Il server usa uno o più header "Set-Cookie" in una qualsiasi risposta
  - Set-Cookie: SID=1234556234; *Attr1=Val1*; *Attr2*
- Il client successivamente fa una richiesta ed include il cookie tramite l'header "Cookie"
  - Cookie : SID=1234556234[; Cookie2=val2; …]

Attributi del cookie

- Oltre al valore del cookie stesso sono possibili parametri aggiuntivi che aggiungono limitazioni all'utilizzo dei cookie
  – Path, Domain, Secure , HttpOnly, Expires

- Gli attributi vengono memorizzati dallo user agent insieme al cookie stesso
  – utilizzati per determinare quali cookie inviare in una determinata richiesta

Attributi "Expires", "Max-Age", "Domain"

- Expires
  - Indica la data limite per la validità del cookie
  - Se non specificato il cookie è "di sessione", ovvero perdura fino a che lo user agent non viene terminato
- Max-Age
  - Massimo tempo di vita in secondi
- Domain
  - indica il dominio per il quale impostare il cookie
  - Può essere una wildcard, che vale per tutti i sottodomini

Attributi "Path", "Secure", "HttpOnly"

- Path
  - Se assente viene determinato in base all'URI della risorsa richiesta
  - Lo user agent invierà un cookie solo per risorse che sono gerarchicamente contenute nel path
- Secure
  - Lo user agent può include il cookie in una richiesta solo se questa avviene su un canale sicuro (HTTP su TLS)
- HttpOnly
  - Il cookie è limitato alle richieste HTTP e non è accedibile ad esempio agli script

Cancellazione un cookie

- Il server manda un header Set-Cookie con Expires antecedente o Max-Age minore o uguale a zero

AUTENTICAZIONE

Autenticazione HTTP

- Il protocollo HTTP mette a disposizione un framework generico per l'autenticazione

- Il protocollo si basa su un meccanismo di challenge-response tra server e client
  - Il client richiede una risorsa protetta
  - Il server risponde richiedendo credenziali

Risposta 401 (Unauthorized)

- Se il client prova ad accedere una risorsa "protetta", l'origin server risponde chiedendo al client informazioni di autenticazione per l'accesso alla risorsa richiesta

- La risposta contiene l'header **WWW-Authenticate**
  - Elenco di metodi di autenticazione accettati
  - Il client sceglie quello che preferisce

```
WWW-Authenticate: Newauth realm="apps",
type=1, title="Login to \"apps\"",
Basic realm="simple"
```

Realm

- Indica una particolare area protetta di un sito Web
- Ogni sito può avere diverse aree protette, identificate da realm diversi

Invio delle credenziali

- Il client che invia le credenziali (in seguito ad una richiesta da parte del server o di propia iniziativa) le include nell'header **Authorization** delle richieste

Basic Authentication

- Utilizza il modello username + password per l'autenticazione
- Il client utilizza l'header Authorization per inviare username e password
  - *username*:*password* codificati in base64
- Esempio:

```
Authorization: Basic cHBhbGxpbm86czNjcjN0
```

Problemi della Basic Authentication

- Lo username e la password vengono inviati non cifrati
- Basta intercettare una volta il token inviato dal client e questo è valido per tutto il realm di autenticazione