Sistemi Distribuiti

Laurea in Informatica

Flavio De Paoli

flavio.depaoli@unimib.it

●●● INSIDE&S Lab ●●●

http://inside.disco.unimib.it/

Content

- Message-oriented communication
  - The Web and HTTP messages
  - Messages communication vs stream communication
- Communication types
  - Synchronous and asynchronous communication
  - Persistent and volatile communication
  - Queue-based communication

L'architettura del Web

- Il Web supporta l'interazione tra client e server via HTTP

- Il client è realizzato da un "Browser" o "User-Agent"
- Il server è realizzato da un "Web Server" o "HTTP Server"

Il browser

- Il browser è l'applicazione per il Web sul lato del client
- Un web browser (detto User-Agent) è un programma che consente la navigazione nel Web da parte di un utente
- La funzione primaria di un browser è quella di interpretare il codice con cui sono espresse le informazioni (pagine web) e visualizzarlo (operazione di rendering) in forma di ipertesto
  - I browser moderni hanno anche funzioni più avanzate per trattare altri tipi di dati: es. Multimedialità, RSS, XML, JSON …
  - Il rendering dipende dal dispositivi utilizzati, anche "non visuali", ad esempio per supportare utenti non-vedenti (es., sintesi vocale, alfabeto Braille)



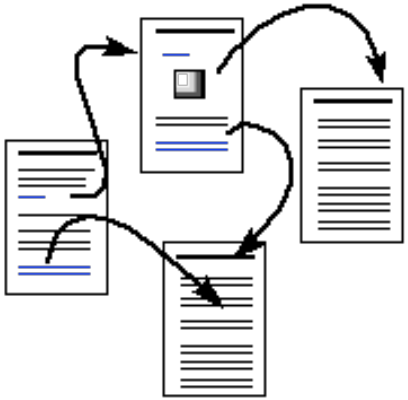


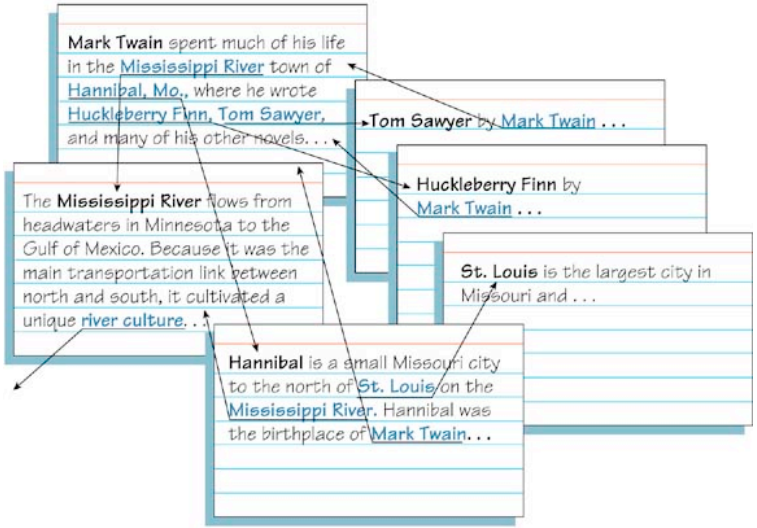

Web Page

- Una pagina web (web page, o anche documento) è costituita da diversi oggetti (risorse nella terminologia del web)
- Una risorsa è un file, cioè una sequenza di dati (in formato digitale) residente in un computer, che è identificato da una URL (cioè un indirizzo univoco per la risorsa)
  - Testo, immagini, musica, …
- La maggior parte delle pagine web sono costituite da un file HTML che definisce la struttura e i contenuti testuali della pagina, più altri oggetti aggiuntivi
  - HTML: HyperText Markup Language, è il linguaggio con cui si scrivono gli ipertesti (si definisce l'aspetto di una pagina, i collegamenti, si inseriscono immagini, …)
- Un Web Server è una applicazione che si occupa di gestire le risorse (file) su un computer e di renderle disponibili ai client

Gli ipertesti

- Un ipertesto (hypertext) è un insieme di testi o pagine leggibili con l'ausilio di un'interfaccia elettronica, in maniera non sequenziale, tramite hyperlink (o più semplicemente link, cioè collegamenti), che costituiscono un rete raggiata o variamente incrociata di informazioni organizzate secondo criteri paritetici o gerarchici (es. menu)





URL (Uniform Resource Locator)

- Identifica un oggetto nella rete e specifica come interpretare i dati ricevuti attraverso il protocollo
- Ha cinque componenti principali:
  1. nome del protocollo
  2. indirizzo dell'host
  3. porta del processo (la controparte)
  4. percorso nell'host
  5. Identificatore della risorsa

protocollo`://`*indirizzo_IP*`[:`*porta*`]/`*cammino*`/`*risorsa* RFC 3986

1. 2. 3. 4. 5.

```
ftp://www.adobe.com/dawnload/acroread.exe
http://www.biblio.unimib.it/go/Home/Home-English/Services
http://www.biblio.unimib.it/link/page.jsp?id=47502837
http://www.someSchool.edu/someDept/pic.gif
http://www.someSchool.edu:80/someDept/pic.gif
```

I linguaggi del Web

- I dati testuali sono espressi in linguaggi standard:
  - HTML (per definire la struttura dei contenuti e la loro impaginazione)
    - CSS (per gestire la presentazione, cioè il rendering), HTML dinamico, …
  - XML (focalizzato sui dati e la loro struttura)
    - XSL, RDF, …
  - JSON (focalizzato sui dati e la loro struttura)

- I dati possono essere non testuali (immagini, audio, video)
  - Encoding MIME (definisce il formato dei contenuti)

- La pagine Web possono contenere del codice espresso in linguaggi di scripting per arricchire l'interazione e la dinamicità del rendering
  - JavaScript, VBScript, Java/Applet, Adobe Flash …

Message Oriented communication

Protocollo HTTP

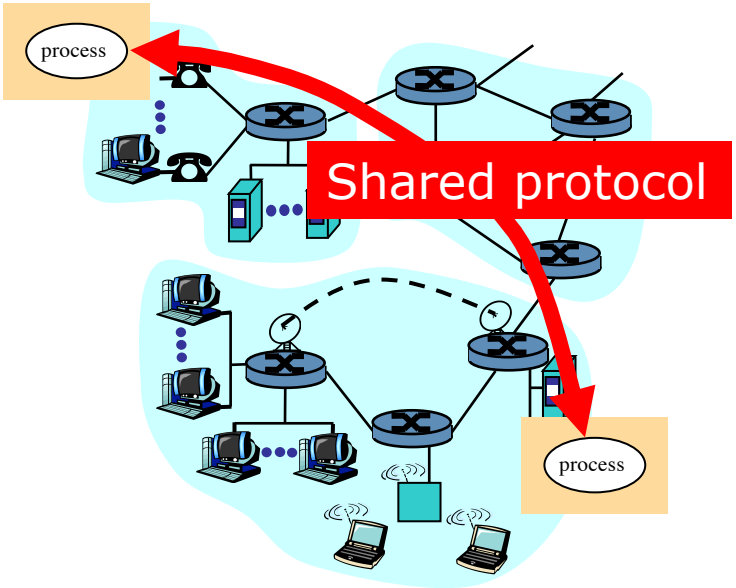
●●● INSIDE&S Lab ●●●
http://inside.disco.unimib.it/

Il concetto di protocollo

- Per poter capire le richieste e formulare le risposte i due processi devono *concordare un protocollo*

I protocolli definiscono il formato, l'ordine di invio e di ricezione dei messaggi tra i dispositivi, il tipo dei dati e le azioni da eseguire quando si riceve un messaggio

- Esempi di protocollo
  - HTTP - HyperText Transfer Protocol
  - FTP - File Transfer Protocol
  - SMTP - Simple Mail Transfer Protocol

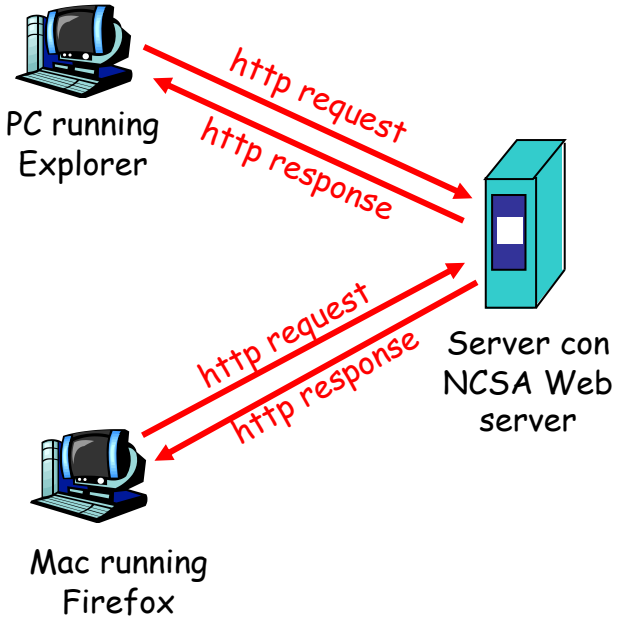


Il Web: protocollo http

http: hypertext transfer protocol

- Protocollo di livello applicativo per il Web
- Usa il modello client/server
  - *client:* browser che richiede, riceve e "mostra" oggetti Web
  - *server:* Web server che invia oggetti in risposta alle richieste

- http1.0: RFC 1945
- http1.1: RFC 2068



Il protocollo http (cont.)

http usa TCP

- Il client inizia una connessione TCP (crea una socket) verso il server sulla porta 80
- Il server accetta la connesione TCP dal client
- Vengono scambiati messaggi http (messaggi del protocollo di livello applicativo) tra il browser (client http) e il Web server (server http)

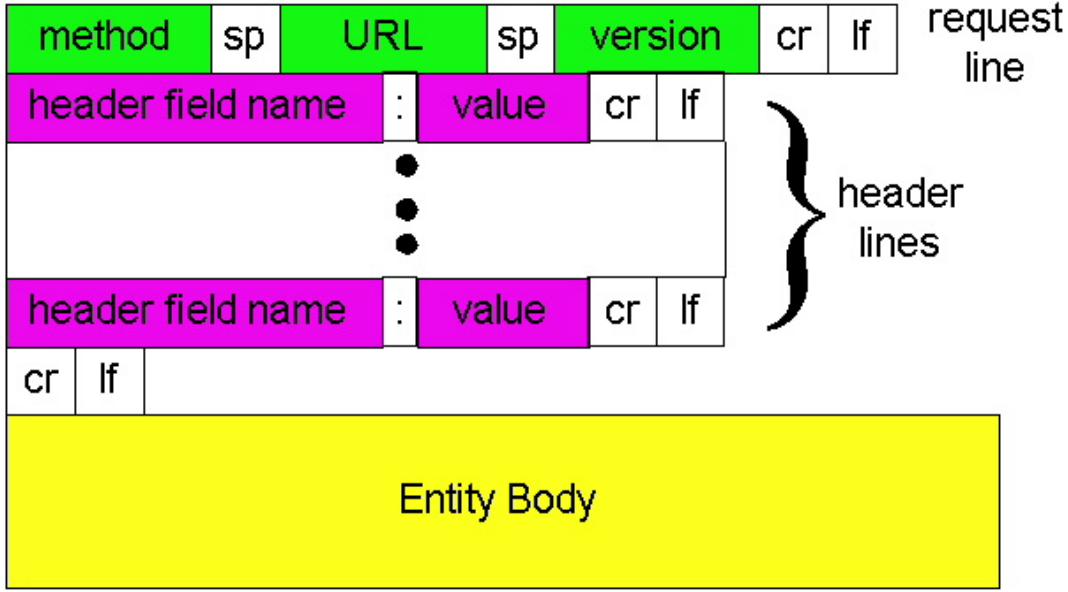
http è "stateless"

- Il server non mantiene informazione sulle richieste precedenti del client
- Quindi:
  ogni richiesta deve contenere tutte le informazioni necessarie per la sua esecuzione.

I protocolli che mantengono informazione di stato sono complessi (es. TCP)!

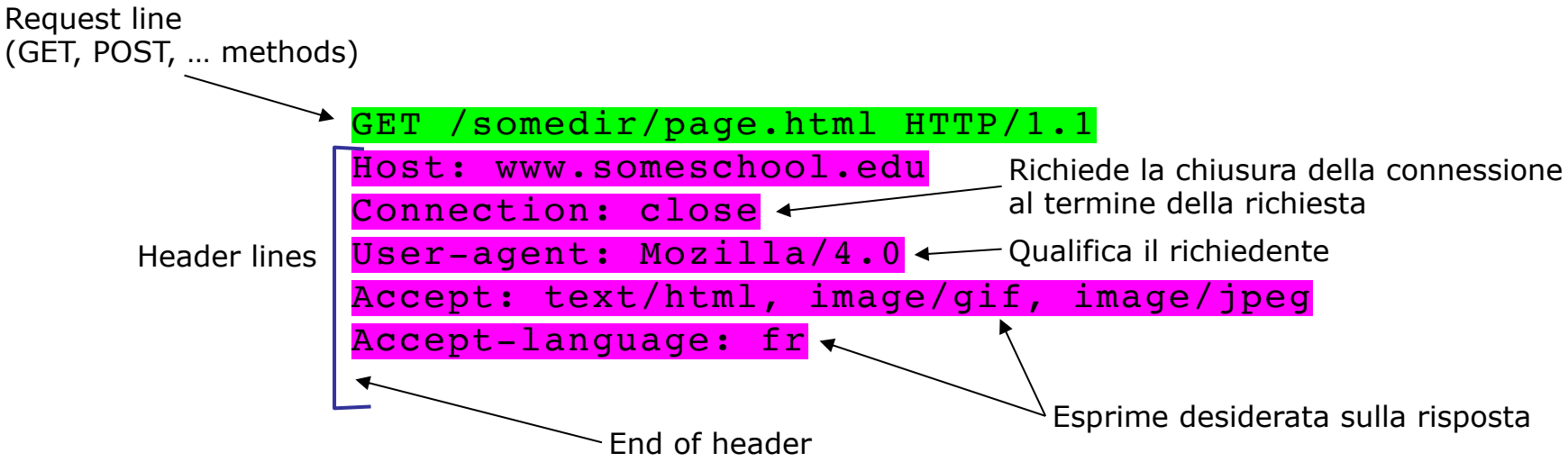
Formato dei messagi http

- Due tipi di messaggi http: *request*, *response*
  - ASCII (formato testo leggibile)
  - *Hanno la stessa struttura!*



Formato dei messagi http

- Messaggio http request



Metodi e applicazioni web

- I principali metodi utilizzati sono:

  Metodo GET
  - Restituisce una *rappresentazione di una risorsa*
  - Include un eventuale input in coda alla URL della risorsa
  - E' safe: l'esecuzione non ha effetti sul server
    => la risposta può essere gestita con una cache dal client
  - Uso tipico: ottenere dati in formato di pagine html e immagini, dati XML o JSON

  Metodo POST
  - Comunica dei dati da elaborare lato server o crea una nuova risorsa subordinata all'URL indicata (vedi più avanti)
  - L'input segue come documento autonomo (body)
  - Non è idempotente: ogni esecuzione ha un diverso effetto
    => La risposta NON può essere gestita con una cache dal client
  - Uso tipico: processare FORM e modificare dati in un DB
- Metodo HEAD
  - Simile al metodo GET ma viene restituito solo l'Head della pagina Web
  - Spesso usato in fase di debugging

HTTP Methods

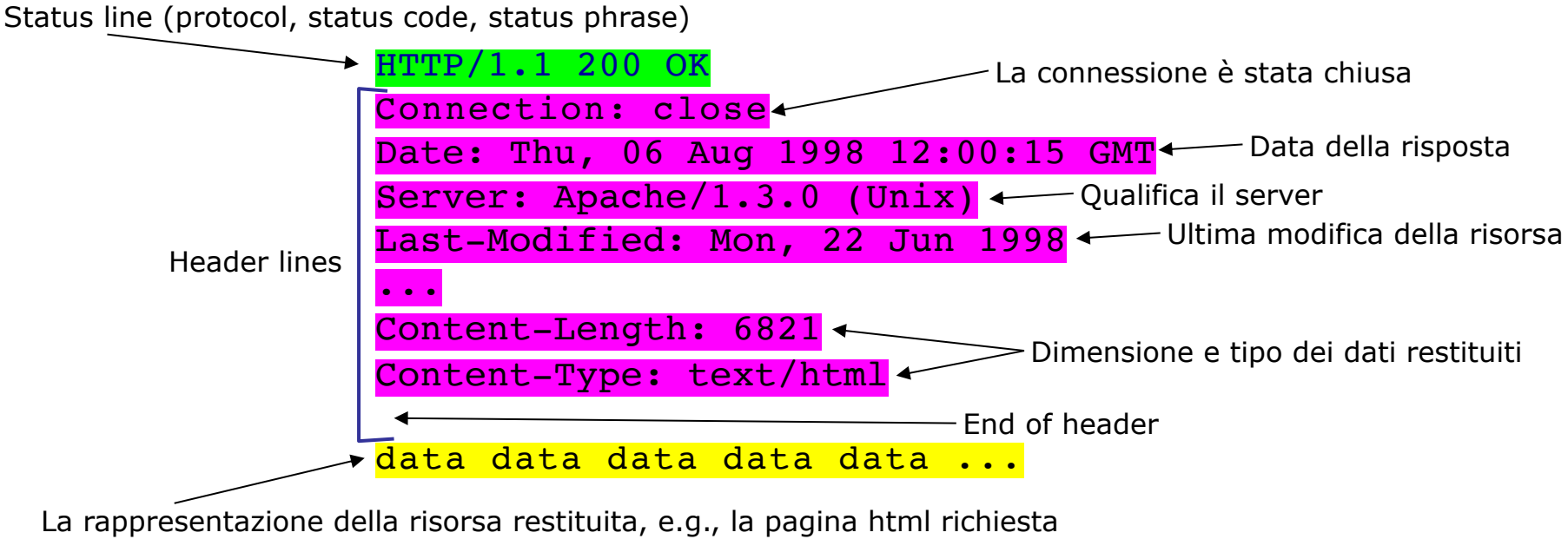
		cache	safe	idempotent
OPTIONS	represents a request for information about the communication options available on the request/response chain identified by the Request-URI			✓
GET	means retrieve whatever information (in the form of an entity) is identified by the Request-URI	✓	✓	
HEAD	identical to GET except that the server MUST NOT return a message-body in the response	✓	✓	
POST	is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line			
PUT	requests that the enclosed entity be stored under the supplied Request-URI			✓
DELETE	requests that the origin server delete the resource identified by the Request-URI			✓
TRACE	is used to invoke a remote, application-layer loop- back of the request message			✓

Safe = methods SHOULD NOT have the significance of taking an action other than retrieval.
Idempotent = the side-effects of N > 0 identical requests is the same as for a single request (aside from error or expiration issues).

http://tools.ietf.org/html/rfc2616

Formato dei messagi http

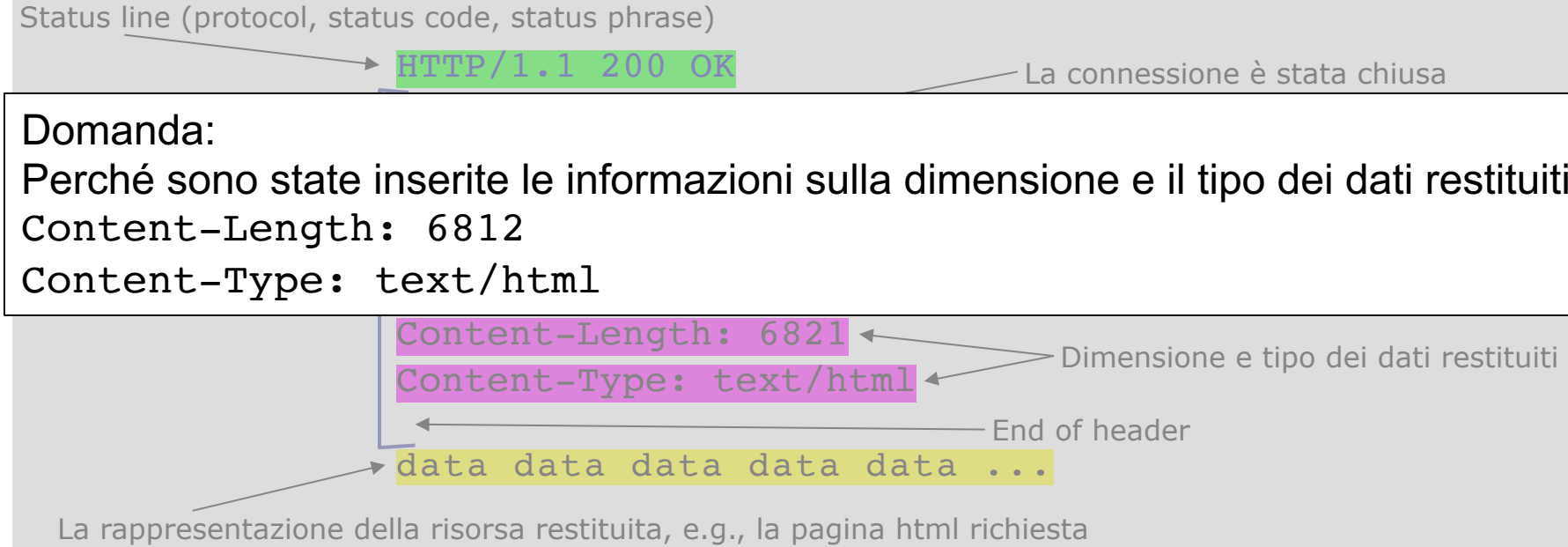
- Messaggio http response



HTTP 1.0: Server chiude connessione al termine della richiesta
HTTP 1.1: mantiene aperta la connessione oppure chiude se la richiesta contiene Connection: close

Formato del messaggio http response

- Messaggio http response



Domanda:
Perché sono state inserite le informazioni sulla dimensione e il tipo dei dati restituiti?

```
Content-Length: 6812
Content-Type: text/html
```

HTTP 1.0: Server chiude connessione al termine della richiesta
HTTP 1.1: mantiene aperta la connessione oppure chiude se la richiesta contiene Connection: close

Response: codici di stato

`1xx` (Informational):	Request received; server is continuing the process.
`2xx` (Success):	The request was successfully received, understood, accepted and serviced.
`3xx` (Redirection):	Further action must be taken in order to complete the request.
`4xx` (Client Error):	The request contains bad syntax or cannot be understood.
`5xx` (Server Error):	The server failed to fulfill an apparently valid request.

- Alcuni esempi più comuni

`200 OK`	Successo, se richiesto l'oggetto è contenuto nel messaggio
`301 Moved Permanently`	L'oggetto richiesto è stato spostato. Il nuovo indirizzo è specificato nel'header (`Location: …`)
`400 Bad Request`	Richiesta incomprensibile al server
`404 Not Found`	Il documento non è stato trovato sul server
`505 HTTP Version Not Supported`	

Tipi MIME

- MIME: Multipupose Internet Mail Extension,
  - Mutuato dal protocollo di email SMTP
  - RFC 2045, 2056.
- Qualifica i dati inviati via Internet (in http, qualifica il tipo dei dati del body)

Text

- Esempi di sottotipi: **`plain`**, **`html`**

Image

- Esempi di sottotipi : **`jpeg`**, **`gif`**

Audio

- Esempi di sottotipi : **`basic`** (8-bit mu-law encoded), **`32kadpcm`** (32 kbps coding)

Video

- Esempi di sottotipi : **`mpeg`**, **`quicktime`**

Application

- Dati che devono essere processati da un' applicazione prima di essere "visibili"
- Esempi di sottotipi : **`msword`**, **`octet-stream`**

https://www.iana.org/assignments/media-types/media-types.xhtml#text

GET condizionale

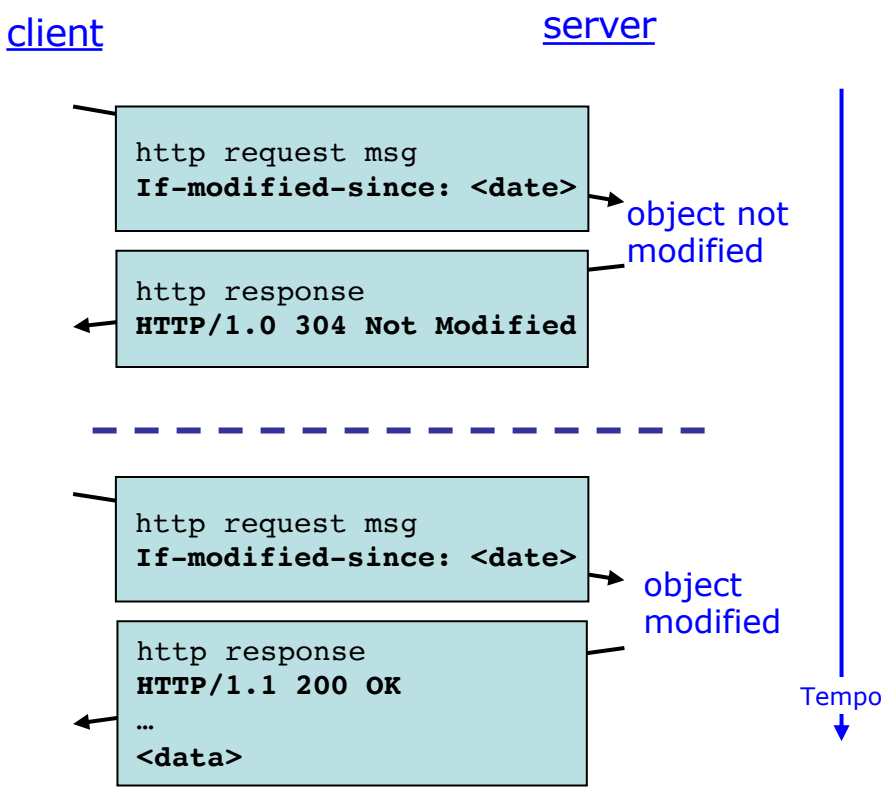
Obiettivo:
non inviare oggetti che il client ha già in cache

- client: data dell'oggetto memorizzato in cache

```
If-modified-since: <date>
```

- server: la risposta è vuota se l'oggetto in cache è aggiornato:

```
HTTP/1.0 304 Not Modified
```



Cookie

Obiettivo:
legare più richieste per associare un identificatore della conversazione

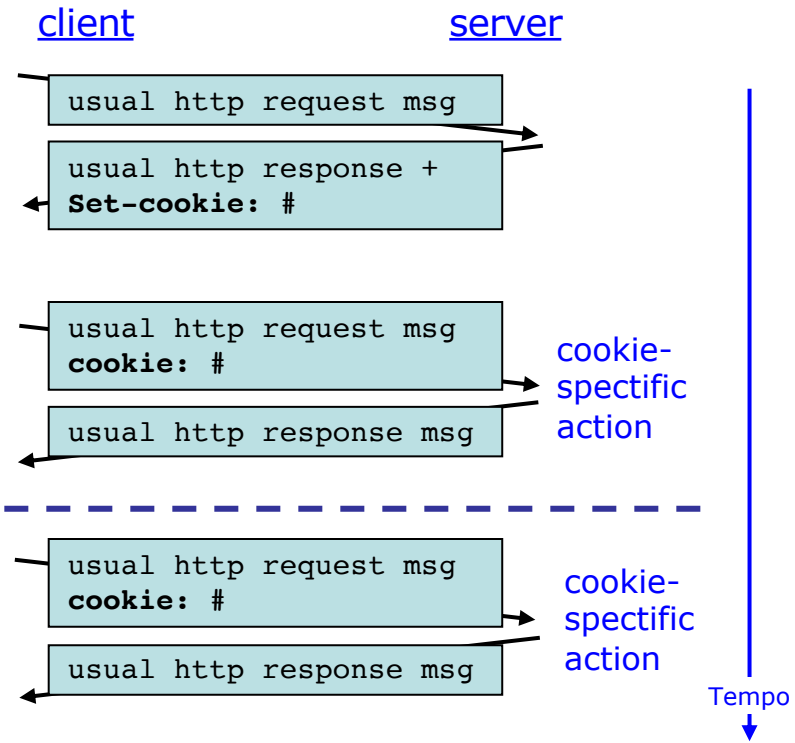
- server: invia un “cookie” al client con la risposta

```
Set-cookie: 1678453
```

- client: presenta il cookie in accessi successivi

```
cookie: 1678453
```

- Il server controlla il cookie presentato
  - Autenticazione
  - Traccia delle preferenze dell’ utente
  - Sessione di lavoro



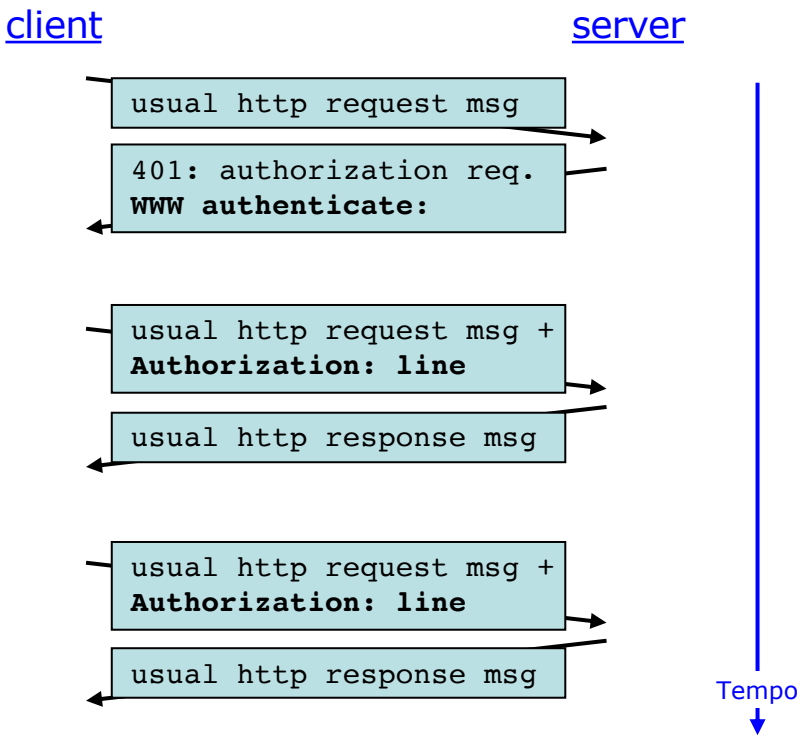
Autenticazione

Obiettivo:
controllare l'accesso ai documenti sul server

- Http è un protocollo stateless:
  il client deve autenticare ogni richiesta
- server: rifiuta la connessione e invia
  `401 authorization required`

  `WWW authenticate:`
- client: invia login e password nell'header del messaggio di richiesta

  authorization: line
- Il client memorizza i dati in modo che l'utente non debba digitarli ogni volta

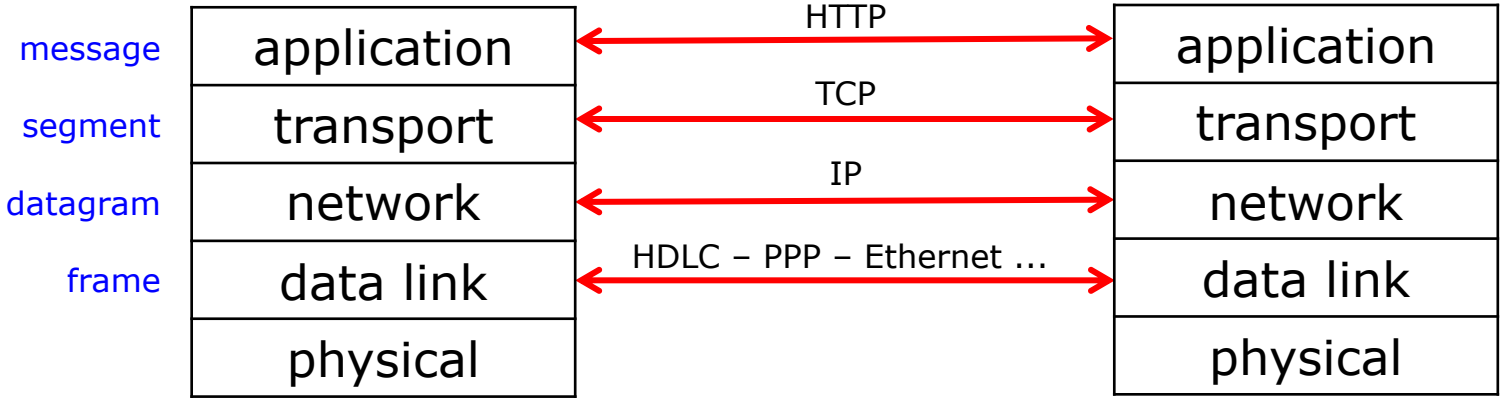


Message Oriented communication

Comunicazione a flusso e a messaggio

●●● INSIDE&S Lab ●●●
http://inside.disco.unimib.it/

Message vs stream communication



Message vs stream communication

Applicazione:

- Invia i messaggi come stream di byte al servizio di trasporto
- Legge lo stream di byte dal servizio di trasporto e ricostruisce i messaggi

```
GET /index.html HTTP/1.1<CR><LF>Host: www.unimib.it<CR><LF>User-agent: Mozilla/4.0<CR><LF><CR><LF>
```

Servizio UDP:

- Scompone lo stream di byte ricevuto in segmenti
- Invia i segmenti, con una deteraminata politica, ai servizi network

```
GET /index.html HTTP | /1.1<CR><LF>Host: ww | w.unimib.it<CR><LF>U | ser-agent: Mozilla/4 | .0<CR><LF><CR><LF>
```

Servizio TCP:

- Scompone e invia come UDP
- Ogni segmento viene numerato per garantire
  - Riordinamento dei segmenti arrivati
  - Controllo delle duplicazioni (scarto dei segmenti con ugual numero d'ordine)
  - Controllo delle perdite (rinvio dei segmenti mancanti)

```
0 GET /index.html HTTP | 2 w.unimib.it<CR><LF>U | 1 /1.1<CR><LF>Host: ww | 3 ser-agent: Mozilla/4 | 4 .0<CR><LF><CR><LF>
```

HTTP/TCP/IP/TCP/HTTP

HTTP

GET /index.html HTTP/1.1<CR><LF>Host: www.unimib.it<CR><LF>User-agent: Mozilla/4.0<CR><LF><CR><LF>

message

TCP

071 069 084 032 047 105 110 100 101 120 046 104 116 109 108 032 072 084 084 080 047 049 046 049 013
010 072 111 115 116 058 032 119 119 119 046 117 110 105 109 105 098 046 105 116 013 010 085 115 101
114 045 097 103 101 110 116 058 032 077 111 122 105 108 108 097 047 052 046 048 013 010 013 010

stream

0 071 069 084 032 047 | 1 105 110 100 101 120 | 2 046 104 116 109 108 | . . . | N 048 013 010 013 010

segment

IP – DATA LINK – PHYSICAL

0 071 069 084 032 047 | 2 046 104 116 109 108 | . . . | 1 105 110 100 101 120 | N 048 013 010 013 010

segment

071 069 084 032 047 105 110 100 101 120 046 104 116 109 108 032 072 084 084 080 047 049 046 049 013
010 072 111 115 116 058 032 119 119 119 046 117 110 105 109 105 098 046 105 116 013 010 085 115 101
114 045 097 103 101 110 116 058 032 077 111 122 105 108 108 097 047 052 046 048 013 010 013 010

stream

TCP

HTTP

GET /index.html HTTP/1.1<CR><LF>Host: www.unimib.it<CR><LF>User-agent: Mozilla/4.0<CR><LF><CR><LF>

message

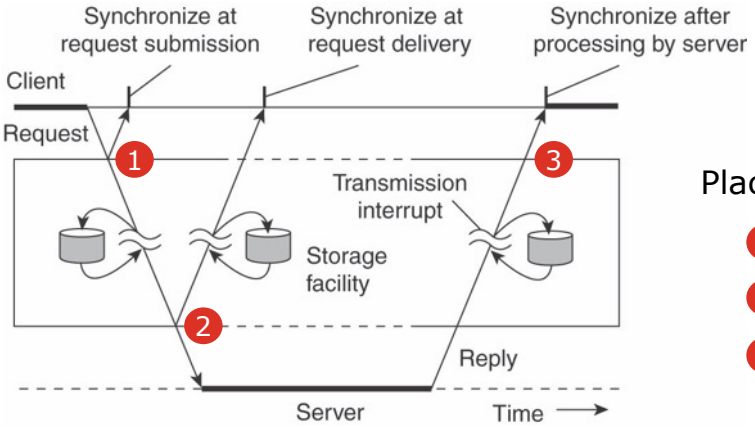
Message Oriented communication

Tipi di comunicazione

●●● INSIDE&S Lab ●●●
http://inside.disco.unimib.it/

Tipi di comunicazione

- Comunicazione asincrona o asincrona
- Comunicazione transiente
  - Se il destinatario non è connesso, i dati vengono scartati
- Comunicazione persistente
  - Il middleware memorizza i dati fino alla consegna del messaggio al destinatario
  - Non è necessario che i processi siano in esecuzione prima e dopo l'invio/ricezione dei messaggi

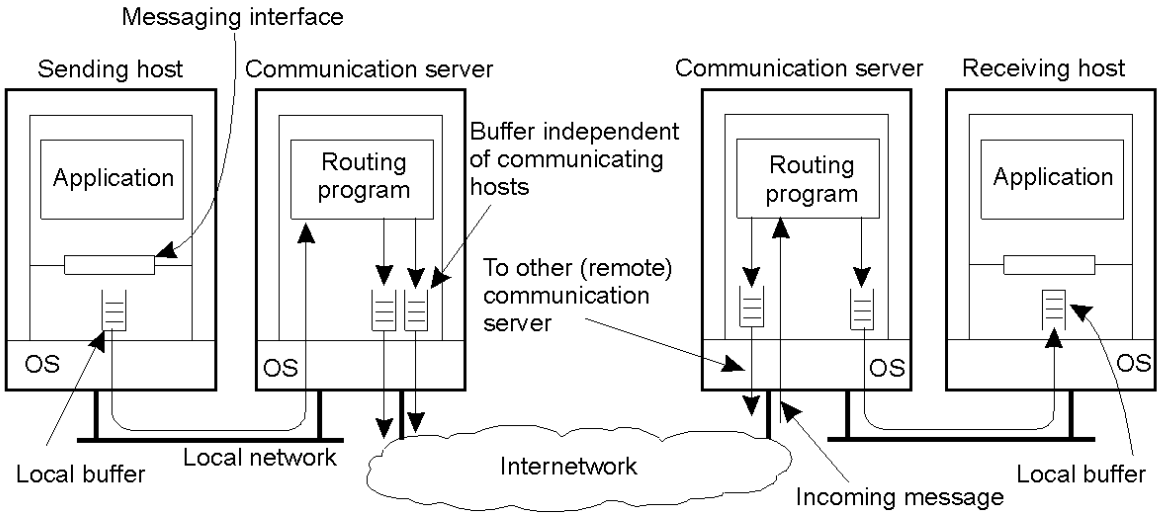


Places for synchronization

1. At request submission
2. At request delivery
3. After request processing

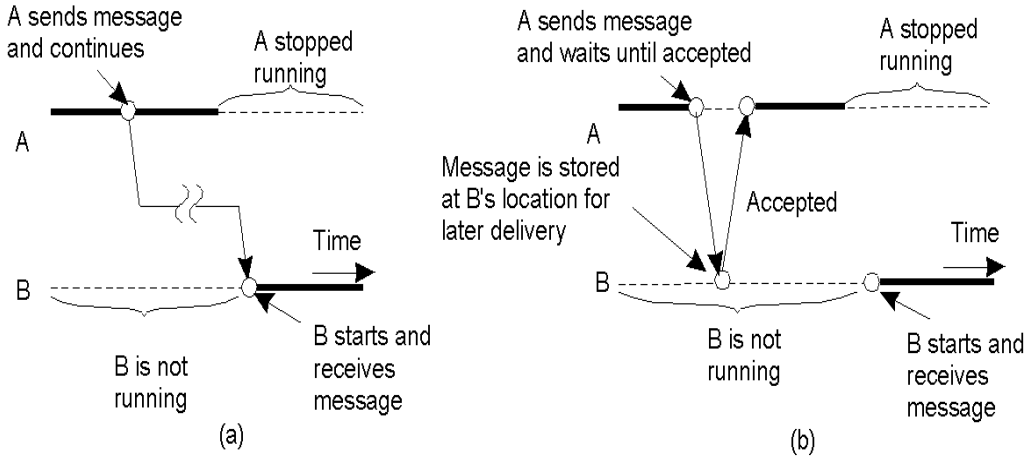
Persistence and Synchronicity in Communication (1)

- General organization of a communication system in which hosts are connected through a network



Persistence and Synchronicity in Communication (2)

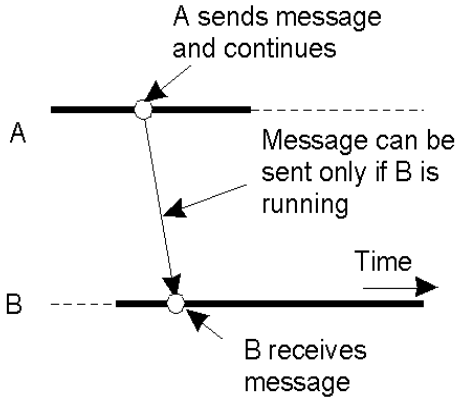
a) Persistent asynchronous communication
b) Persistent synchronous communication



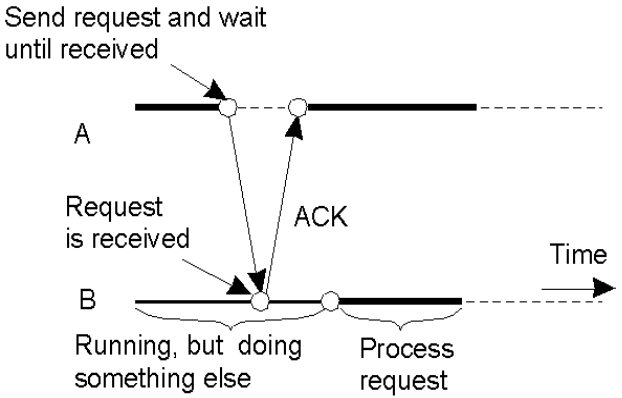
Persistence and Synchronicity in Communication (3)

c) Transient asynchronous communication

d) Receipt-based transient synchronous communication



(c)

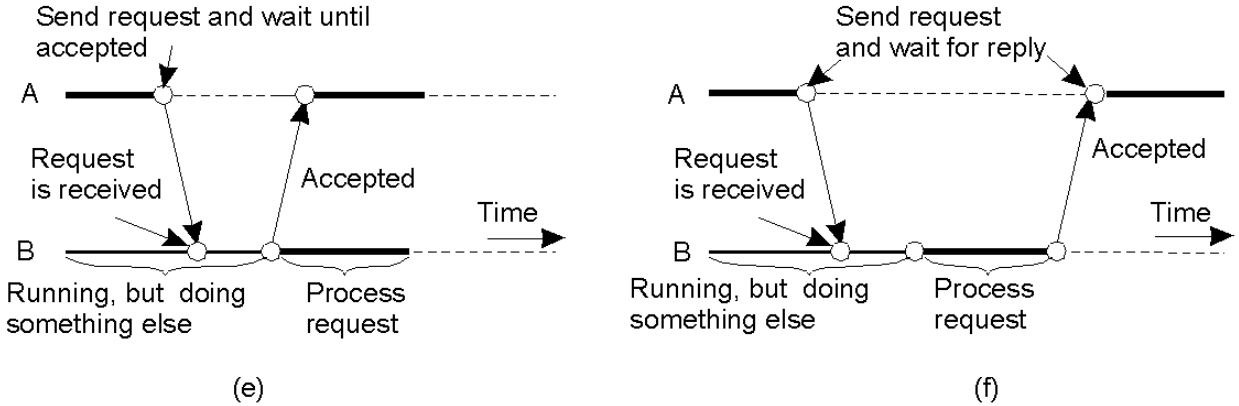


(d)

Persistence and Synchronicity in Communication (4)

e) Delivery-based transient synchronous communication at message delivery

f) Response-based transient synchronous communication

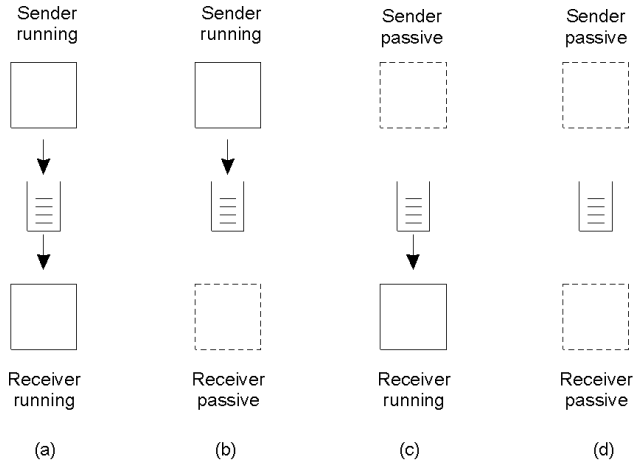


Message-Oriented Persistent Communication

●●● INSIDE&S Lab ●●●
http://inside.disco.unimib.it/

Message-Queuing Model (1)

- They offer intermediate-term storage capacity for messages, without requiring either the sender or receiver to be active during message transmission.
- Four combinations for loosely-coupled communications using queues.



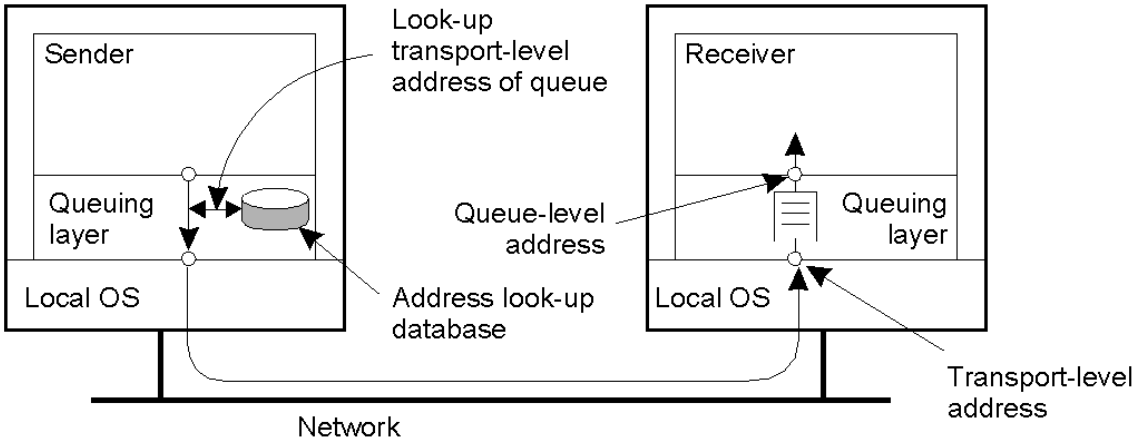
Message-Queuing Model (2)

- Basic interface to a queue in a message-queuing system.

Primitive	Meaning
Put	Append a message to a specified queue
Get	Block until the specified queue is nonempty, and remove the first message
Poll	Check a specified queue for messages and remove the first. Never block.
Notify	Install a handler to be called when a message is put into the specified queue.

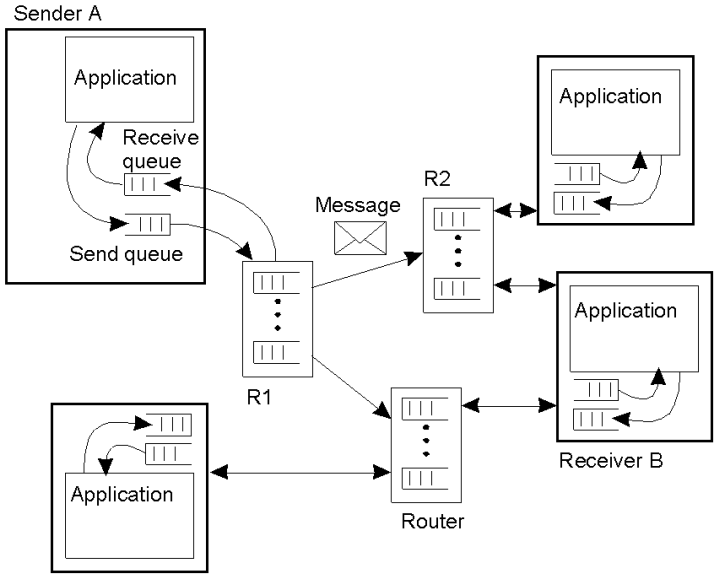
General Architecture of a Message-Queuing System (1)

- The relationship between queue-level addressing and network-level addressing



General Architecture of a Message-Queuing System (2)

- The general organization of a message-queuing system with routers.



Message Brokers

- The general organization of a message broker in a message-queuing system.
- *Publish and subscribe* protocols