

Reti di Elaboratori

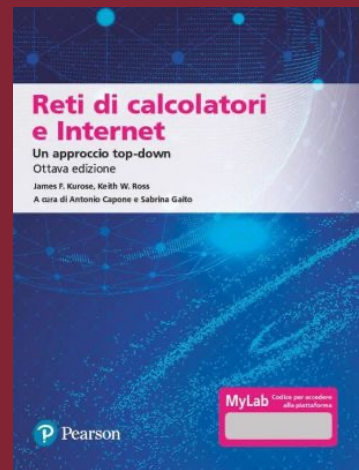
Livello di Applicazione, caching, e-mail



SAPIENZA
UNIVERSITÀ DI ROMA

Alessandro Checco

alessandro.checco@uniroma1.it



Capitolo 2

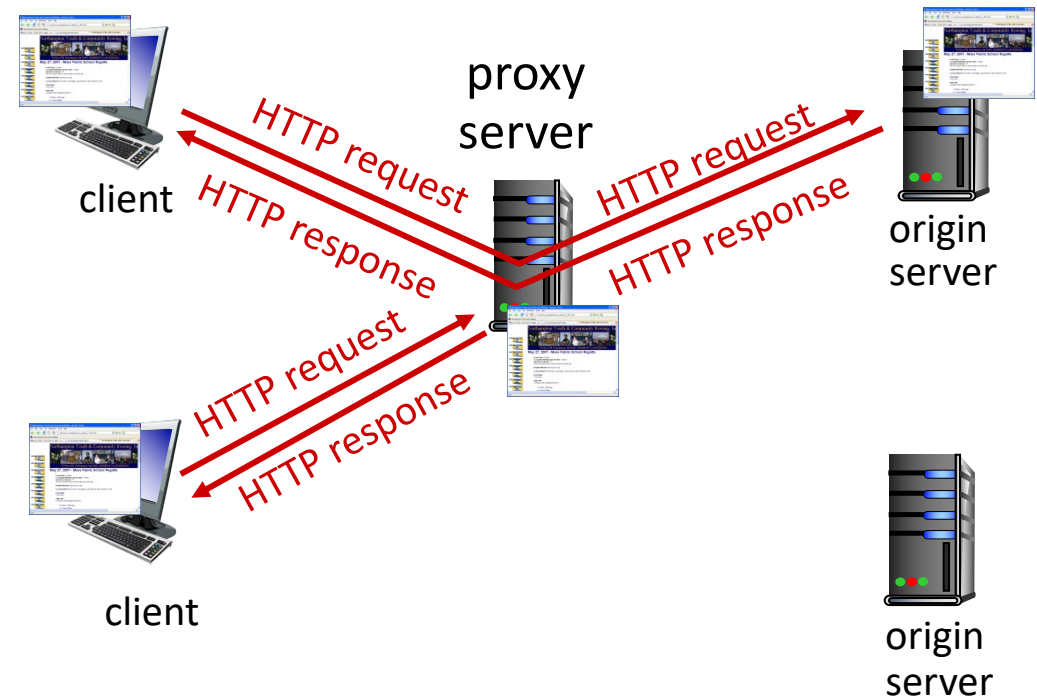
Livello di applicazione: sommario

- Principi delle applicazioni di rete
- Web e HTTP
 - caching
- Posta elettronica, SMTP, IMAP
- Domain Name System: DNS
- Applicazioni P2P
- streaming video e content distribution networks
- programmazione socket con UDP e TCP

Web cache (proxy server)

Obiettivo: rispondere al client senza coinvolgere il server di origine

- l'utente configura il browser in modo che punti a una **Web cache**
- il browser invia tutte le richieste HTTP alla cache
 - **se** oggetto nella cache: la cache restituisce l'oggetto al client
 - **altrimenti** la cache richiede l'oggetto dal server di origine, memorizza nella cache l'oggetto ricevuto, e restituisce l'oggetto al client



Web cache (proxy server)

- La cache Web ha sia il ruolo di client che di server
 - server per il client della richiesta originaria
 - client rispetto al server di origine
 - in genere la cache viene installata dall'ISP (università, azienda, ISP residenziale)
- Perché* usare la Web cache?
- ridurre i tempi di risposta per la richiesta del cliente
 - cache è più vicina al client
 - ridurre il traffico nel link di accesso a una rete locale
 - Ubiqua in Internet
 - consente ai fornitori di contenuti di essere più efficaci

Esempio di memorizzazione nella cache

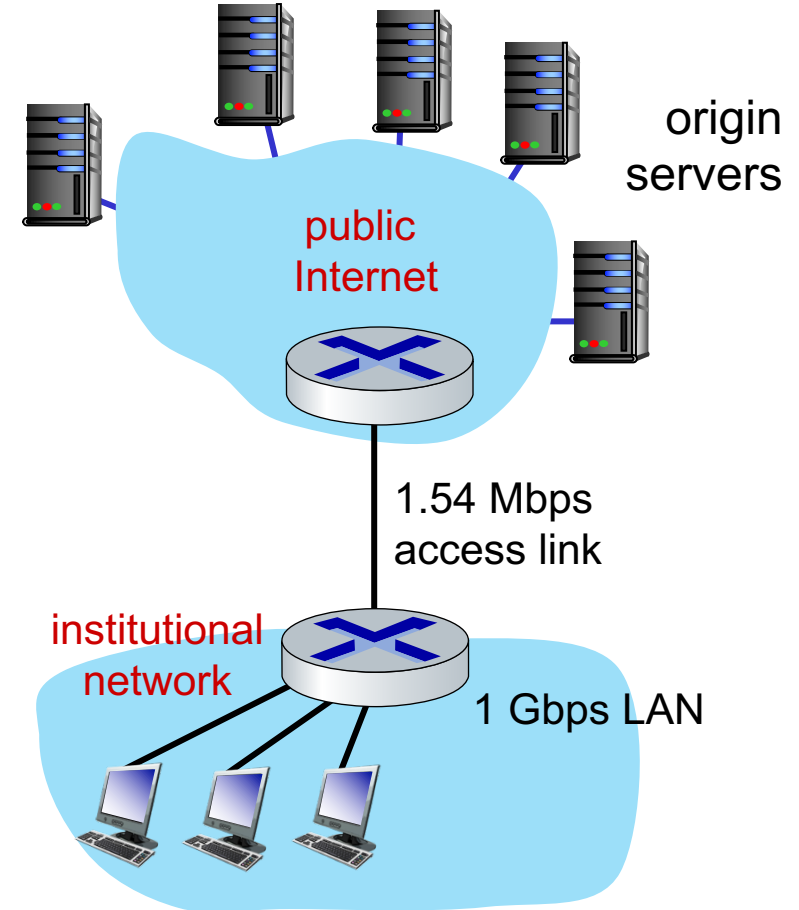
Scenario:

- velocità di collegamento di accesso: 1,54 Mbps
- RTT da router istituzionale a server: 2 s
- Dimensione oggetto Web: 100K bit
- Tasso medio di richieste dai browser ai server (downstream): 15 richieste/s
 - data rate medio in downstream: 1,50 Mbps

Performance:

- Utilizzo LAN: .0015
- Intensità di traffico = **.97**
- ritardo end-to-end = ritardo Internet + ritardo collegamento accesso(queueing) + ritardo LAN = 2 s + [min] + [us]

problema: grandi ritardi quando intensità vicina a 1



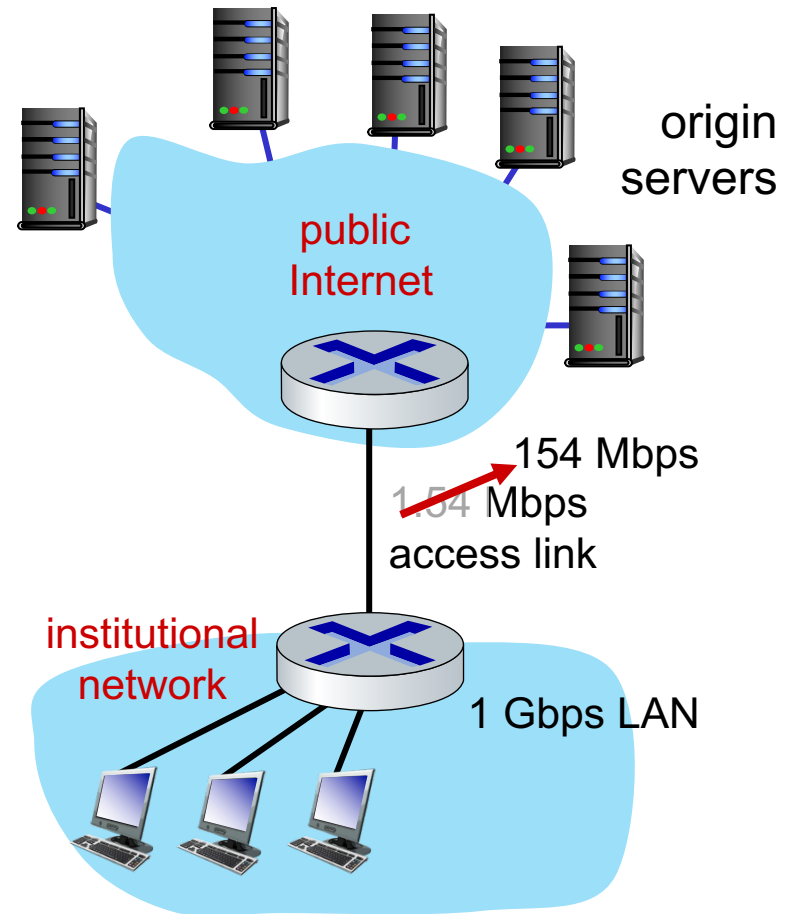
Soluzione 1: acquista un collegamento di accesso più veloce

Scenario:

- velocità di collegamento di accesso: ~~1,54 Mbps~~ ^{154 Mbps}
- RTT da router istituzionale a server: 2 s
- Dimensione oggetto Web: 100K bit
- Tasso medio di richieste dai browser ai server (downstream): 15 richieste/s
 - data rate medio in downstream: 1,50 Mbps

Performance:

- Utilizzo LAN: .0015
- Intensità di traffico = ~~.97~~ ^{.0097}
- ritardo end-to-end = ritardo Internet + ritardo collegamento accesso(queueing) + ritardo LAN = 2 s + ~~[min]~~ ^[ms] + [us]



Costo: link di accesso più veloce (costoso!)

Esempio di memorizzazione nella cache: installa una cache web

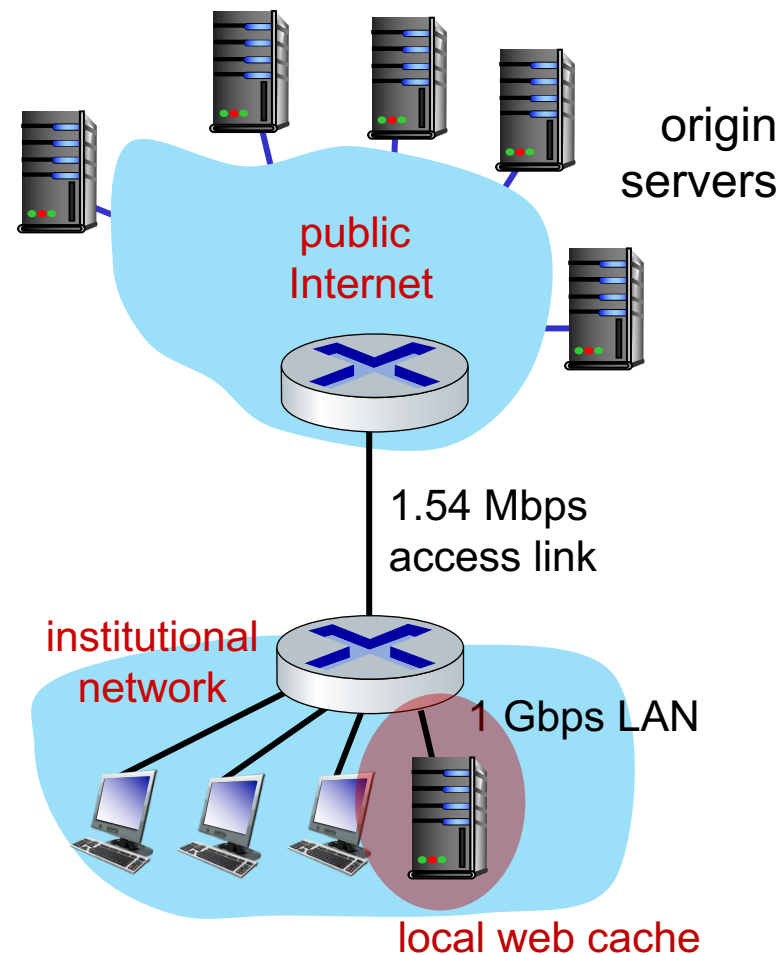
Scenario:

- velocità di collegamento di accesso: 1,54 Mbps
- RTT da router istituzionale a server: 2 s
- Dimensione oggetto Web: 100K bit
- Tasso medio di richieste dai browser ai server (downstream): 15 richieste/s
 - data rate medio in downstream: 1,50 Mbps

Performance:

- Utilizzo LAN: ?
- Intensità di traffico = ?
- ritardo end-to-end = ?

Come si calcola?

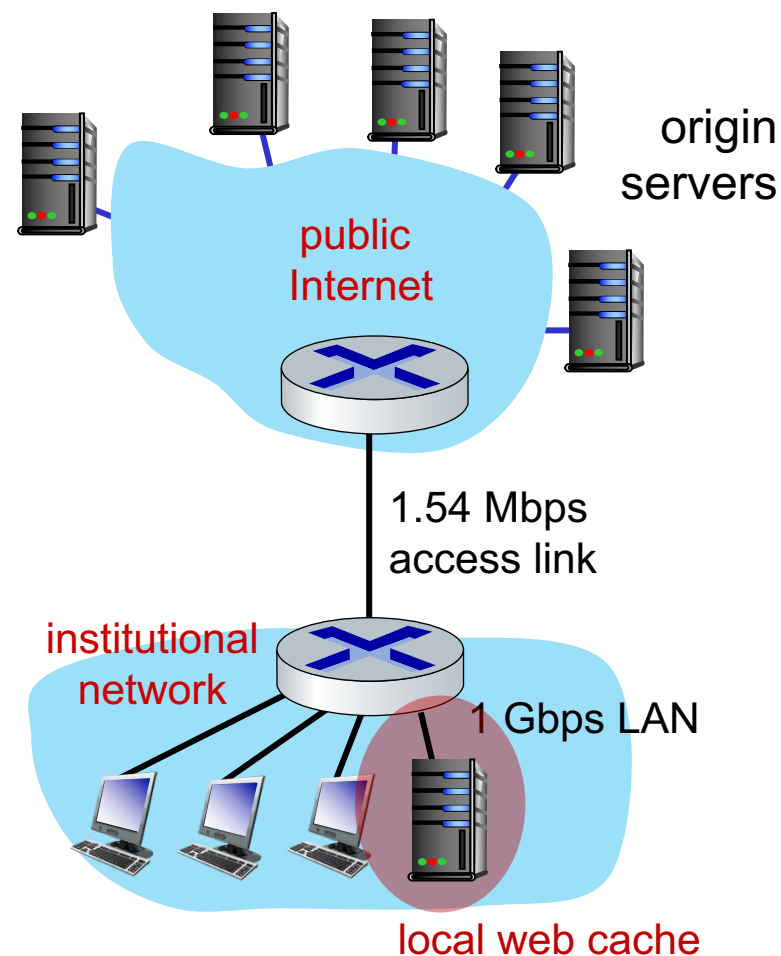


Costo: web cache (economico!)

Esempio di memorizzazione nella cache: installa una cache web

Calcolo dell'intensità di traffico, ritardo end-end con cache:

- supponiamo che il tasso di successo della cache sia 0,4: 40% richieste soddisfatte dalla cache, 60% dal server di origine
- link di accesso: il 60% delle richieste utilizza il link di accesso
- data rate in downstream che passa dall'access link = $0,6 * 1,50 \text{ Mbps} = 0,9 \text{ Mbps}$
- utilizzo = $0,9 / 1,54 = 0,58$
- ritardo medio end-to-end
= $0,6 * (\text{ritardo dai server di origine})$
+ $0,4 * (\text{ritardo quando soddisfatto alla cache})$
= $0,6 (2,01) + 0,4 (\sim \text{ms}) = \sim 1,2 \text{ sec}$

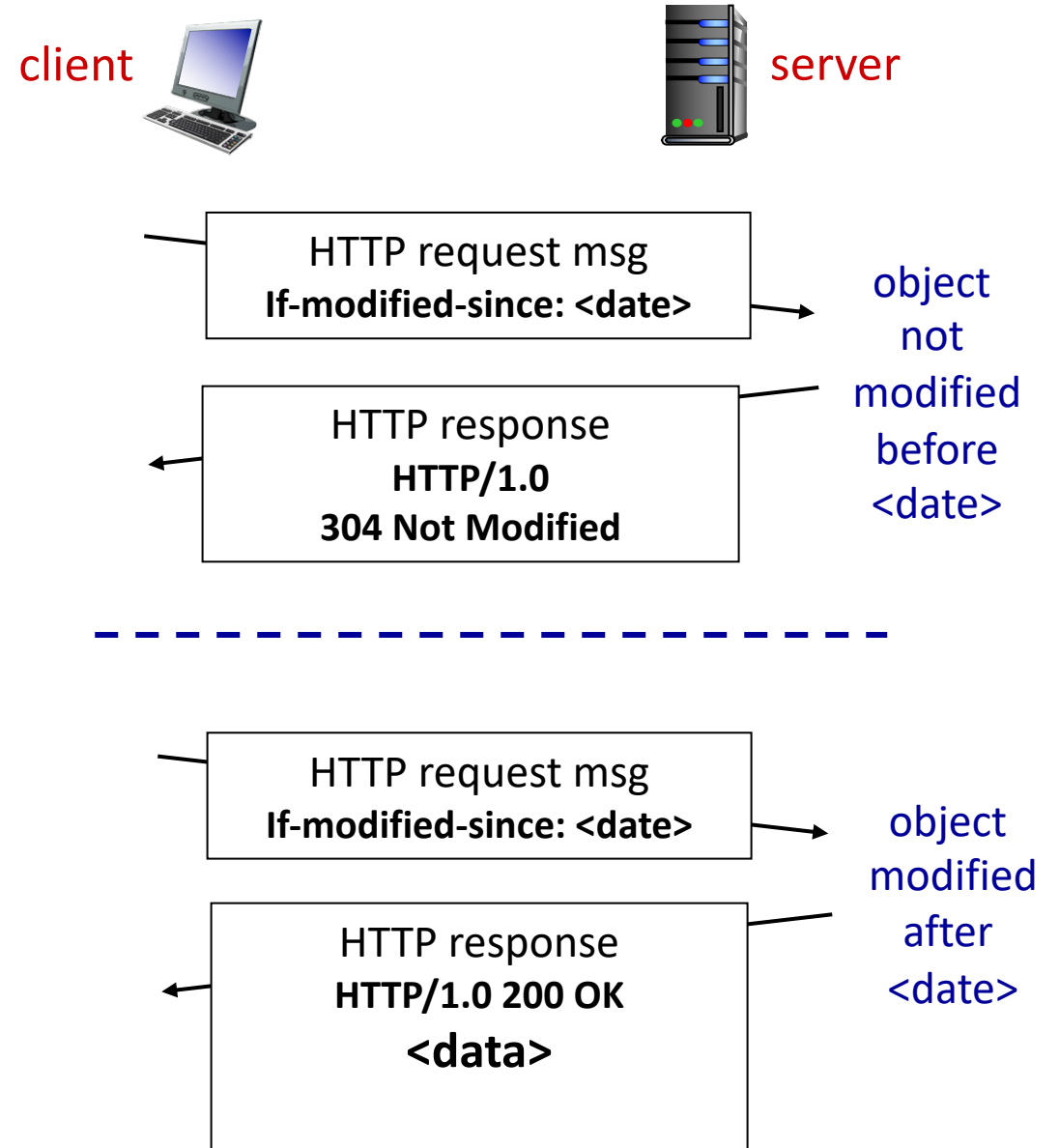


ritardo medio inferiore rispetto al collegamento a 154 Mbps (e anche più economico!)

GET condizionale

Obiettivo: non inviare oggetti se la cache ha la versione corrente

- nessun ritardo di trasmissione dell'oggetto
 - minore utilizzo del link
- **cache:** il client specifica la data della copia memorizzata nella cache nella richiesta HTTP
If-modified-since: <date>
 - **server:** la risposta non contiene alcun oggetto se la copia nella cache è aggiornata:
HTTP/1.0 304 Not Modified



HTTP/2

Obiettivo principale: riduzione del ritardo nelle richieste HTTP multioggetto

HTTP1.1: ha introdotto **più GET in successione** su una singola connessione TCP (pipelined, richiesta GET senza aspettare risposta alla richiesta precedente)

- il server risponde *in ordine* (FCFS) alle richieste GET
- con FCFS, un oggetto piccolo potrebbe dover attendere la trasmissione (**blocco head-of-line (HOL)**) se dopo oggetti di grandi dimensioni
- La perdita di un segmento TCP causa lo stallo del trasferimento di un oggetto

HTTP/2

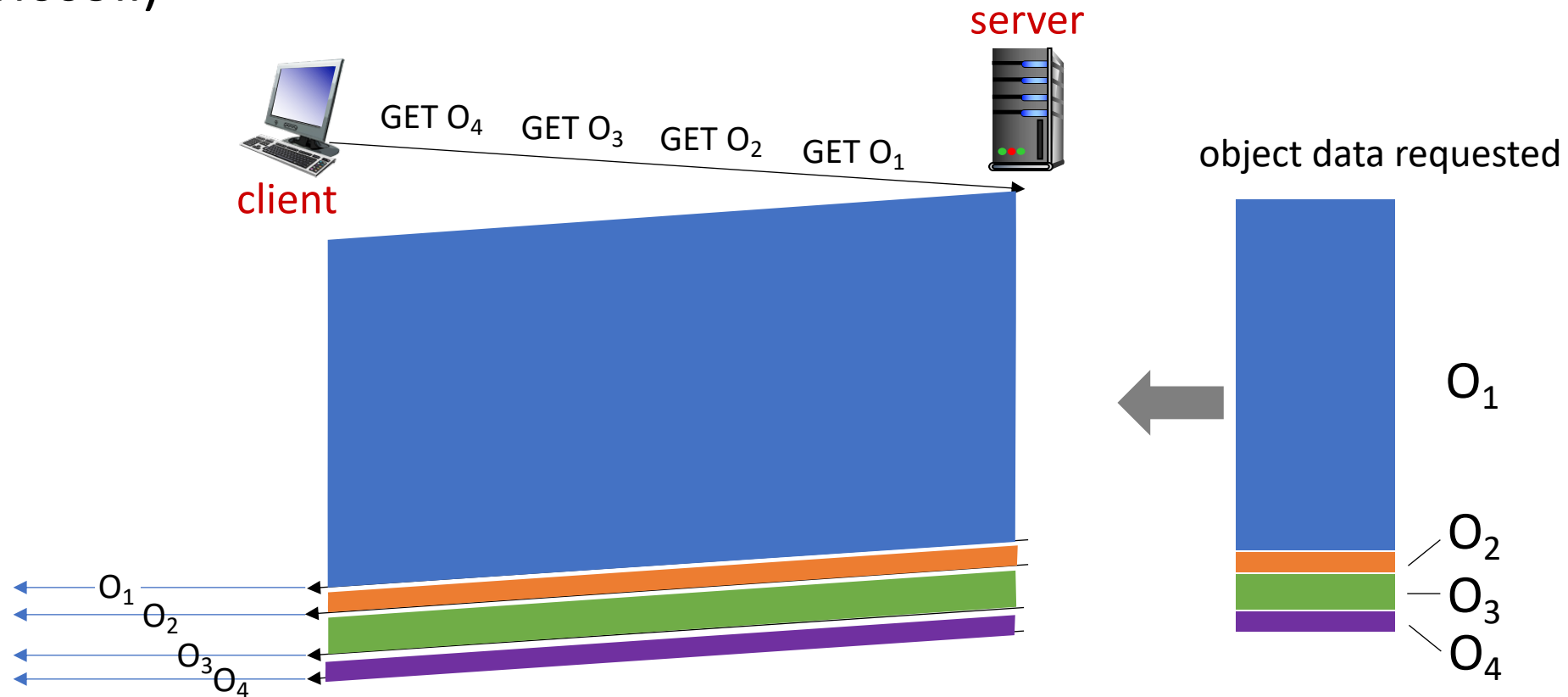
Obiettivo principale: riduzione del ritardo nelle richieste HTTP multioggetto

HTTP/2: [RFC 7540, 2015] maggiore flessibilità al *server* nell'invio di oggetti al client:

- metodi, codici di stato, la maggior parte dei campi di intestazione invariati rispetto a HTTP 1.1
- ordine di trasmissione degli oggetti richiesti in base alla priorità dell'oggetto specificata dal client (non necessariamente FCFS)
- *è possibile inviare* oggetti non richiesti al client
- gli oggetti vengono divisi in frame, schedulati in modo da mitigare il blocco HOL

HTTP/2: attenuazione del blocco HOL

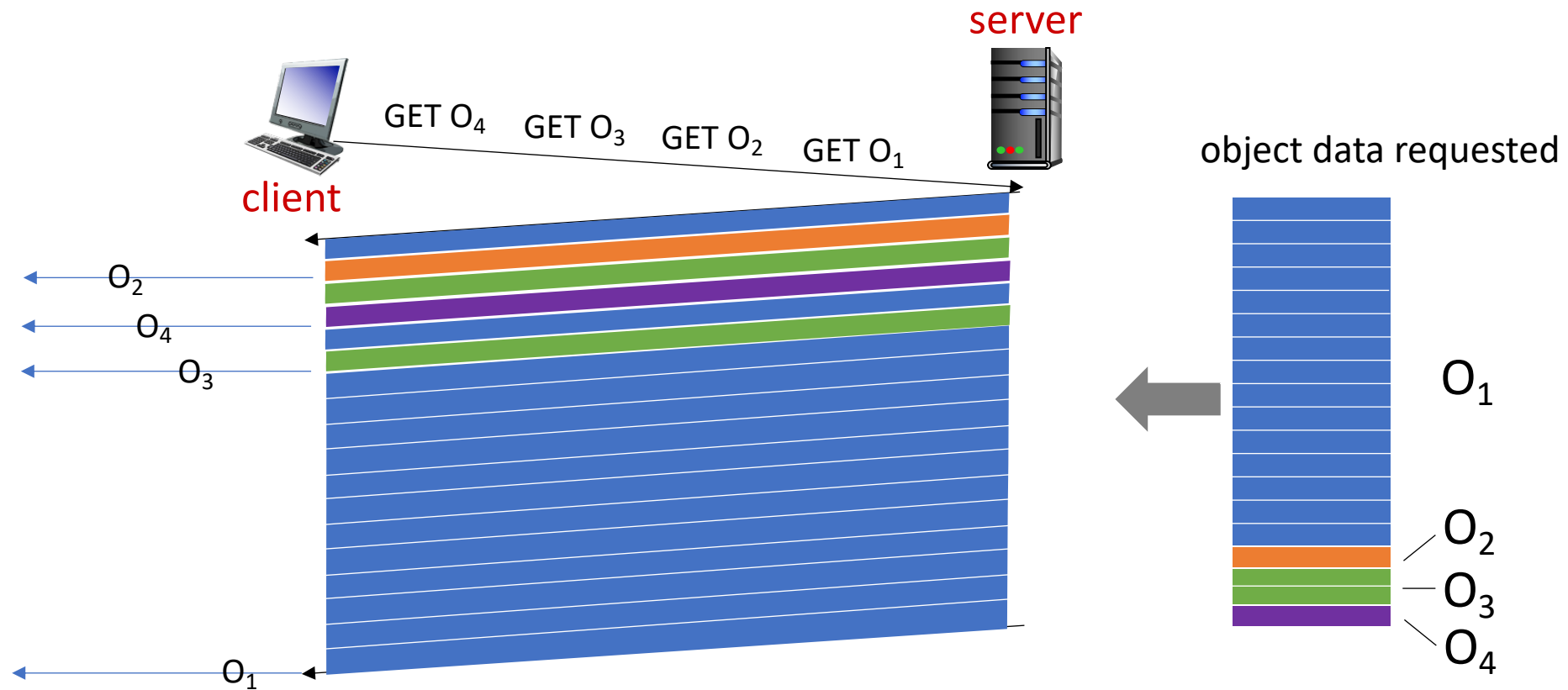
HTTP 1.1: il client richiede 1 oggetto grande (ad es. file video e 3 oggetti più piccoli)



oggetti consegnati nell'ordine richiesto: O₂, O₃, O₄ aspettano dietro O₁

HTTP/2: attenuazione del blocco HOL

HTTP/2: oggetti divisi in frame, trasmissione frame interlacciata



O₂, O₃, O₄ consegnati rapidamente, O₁ un po' più tardi

da HTTP/2 a HTTP/3

Obiettivo principale: riduzione del ritardo nelle richieste HTTP multioggetto

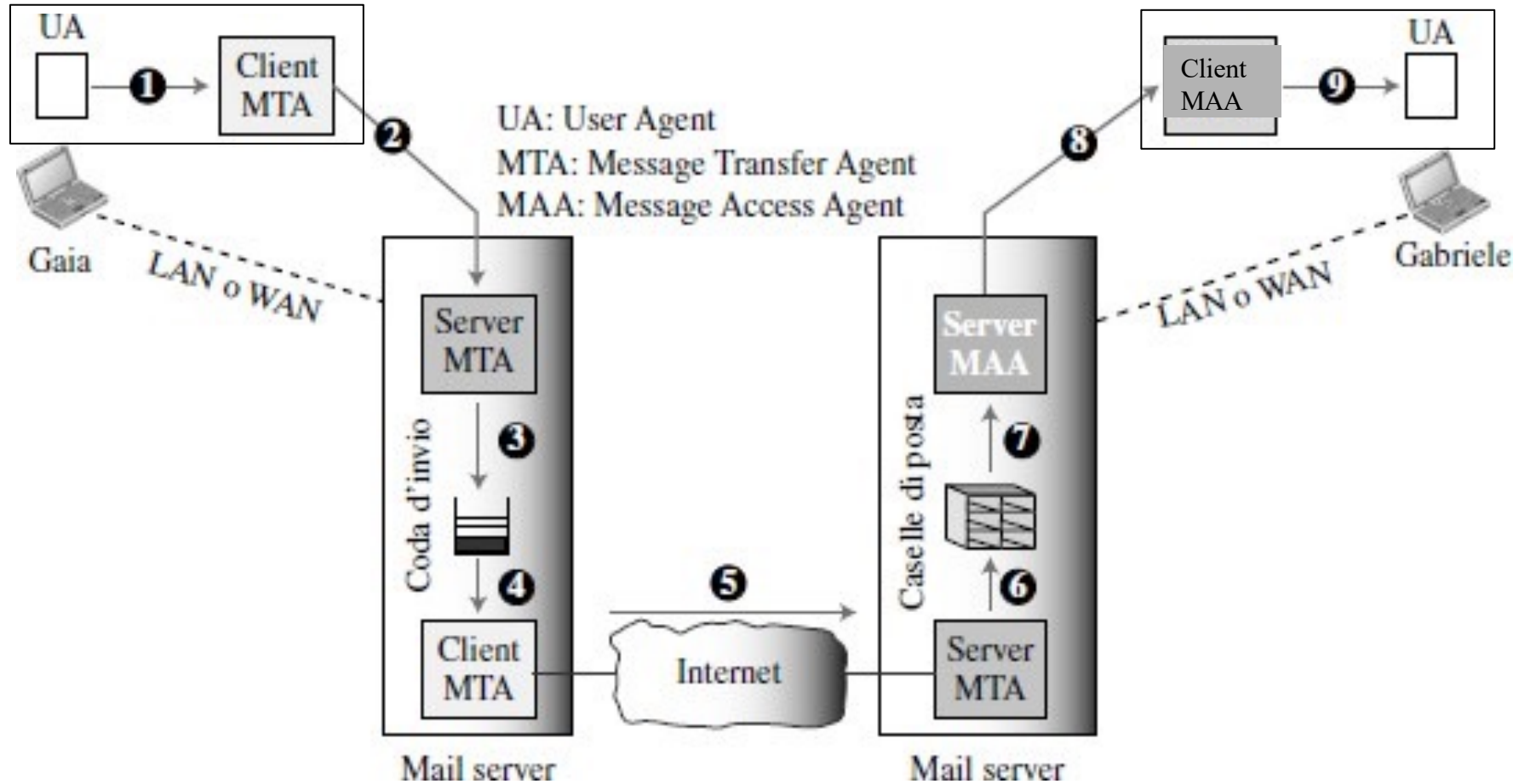
HTTP/2 su singola connessione TCP significa che:

- il ripristino dalla perdita di pacchetti blocca ancora tutte le trasmissioni di oggetti
 - come in HTTP 1.1 (e .0), i browser sono incentivati ad aprire più connessioni TCP parallele per ridurre lo stallo e aumentare il throughput complessivo
- nessuna sicurezza sulla connessione TCP “vanilla”
- **HTTP/3:** aggiunge sicurezza, controllo degli errori e della congestione per oggetto (più pipelining) su UDP
 - lo vedremo meglio quando studieremo il livello di trasporto

Livello di applicazione: sommario

- Principi delle applicazioni di rete
- Web e HTTP
- Posta elettronica, SMTP, IMAP
- Domain Name System: DNS
- Applicazioni P2P
- streaming video e content distribution networks
- programmazione socket con UDP e TCP

Posta elettronica: scenario classico



Tre componenti principali:

1. User agent: usato per scrivere e inviare un messaggio o leggerlo
2. Message Transfer Agent: usato per trasferire il messaggio attraverso Internet
3. Message Access Agent: usato per leggere la mail in arrivo

User agent

- ☐ Lo user agent viene attivato dall'utente o da un timer: se c'è una nuova email informa l'utente
- ☐ detto anche “mail reader”
- ☐ composizione, editing, lettura dei messaggi di posta elettronica
- ☐ esempi:
 - ☐ Eudora, Outlook, Thunderbird
 - ☐ Mail, Pine, Elm
- ☐ i messaggi in uscita o in arrivo sono memorizzati sul server
- ☐ Il messaggio da inviare viene passato al Mail Transfer Agent
- ☐ Come comunicano i Mail Transfer Agent?

Message Transfer Agent

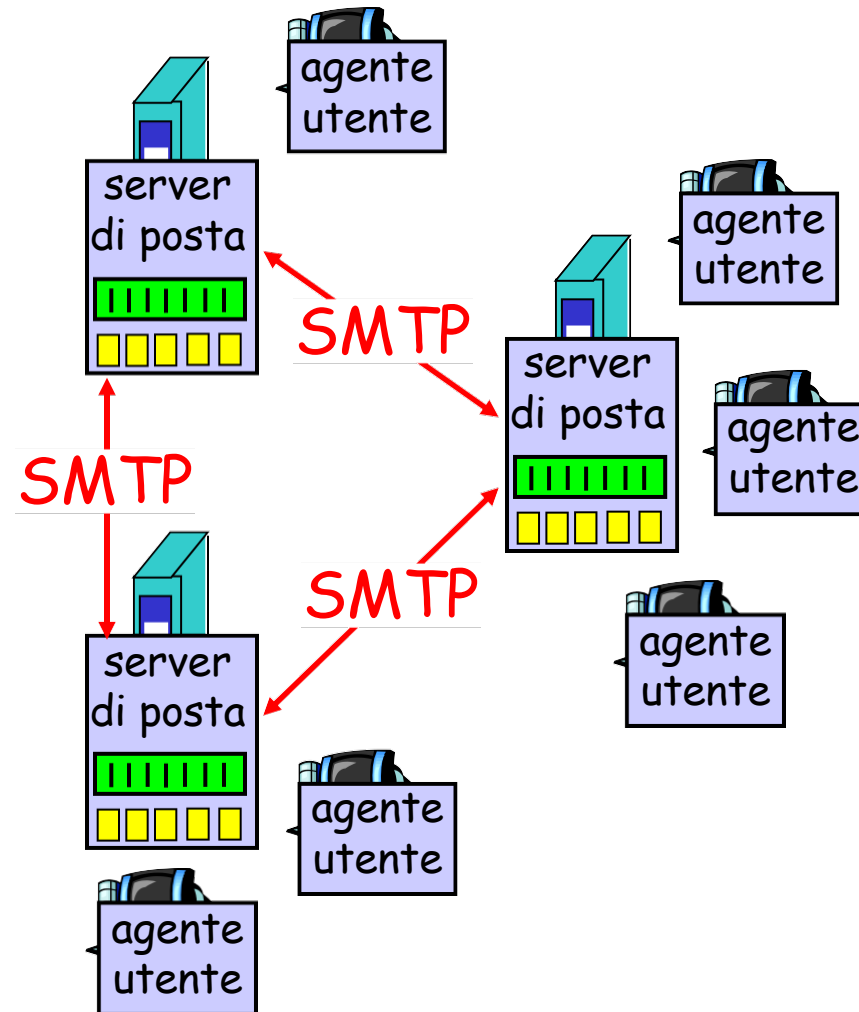
Come comunicano gli MTA?

Server di posta

- Casella di posta (mailbox) contiene i messaggi in arrivo per l'utente
- Coda di messaggi da trasmettere (tentativi ogni x minuti per alcuni giorni)

Protocollo SMTP (Simple Mail Transfer Protocol)

- tra server di posta per inviare messaggi di posta elettronica
- Tra agente utente del mittente e il suo server di posta

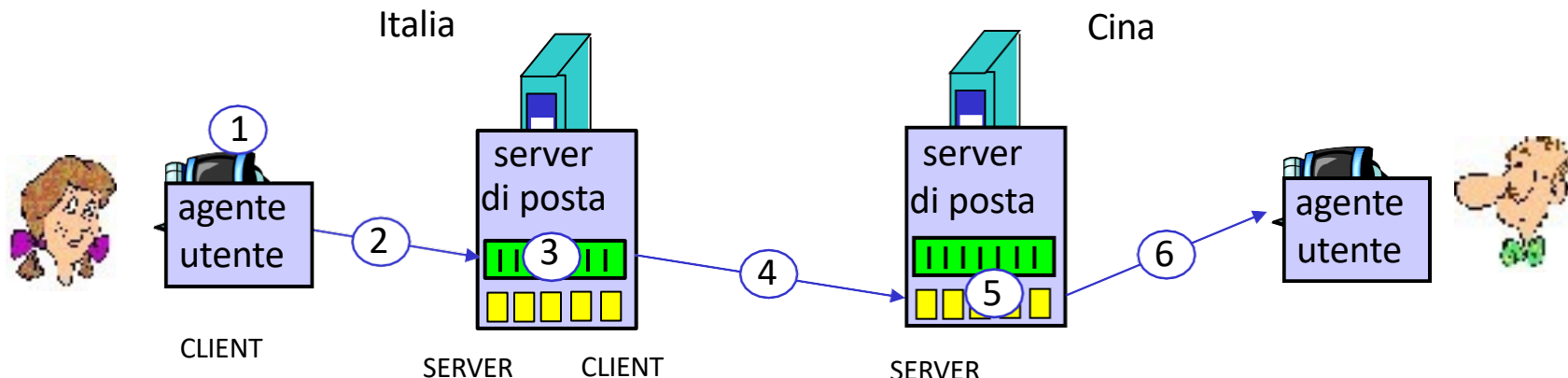


Posta elettronica: SMTP [RFC 5321]

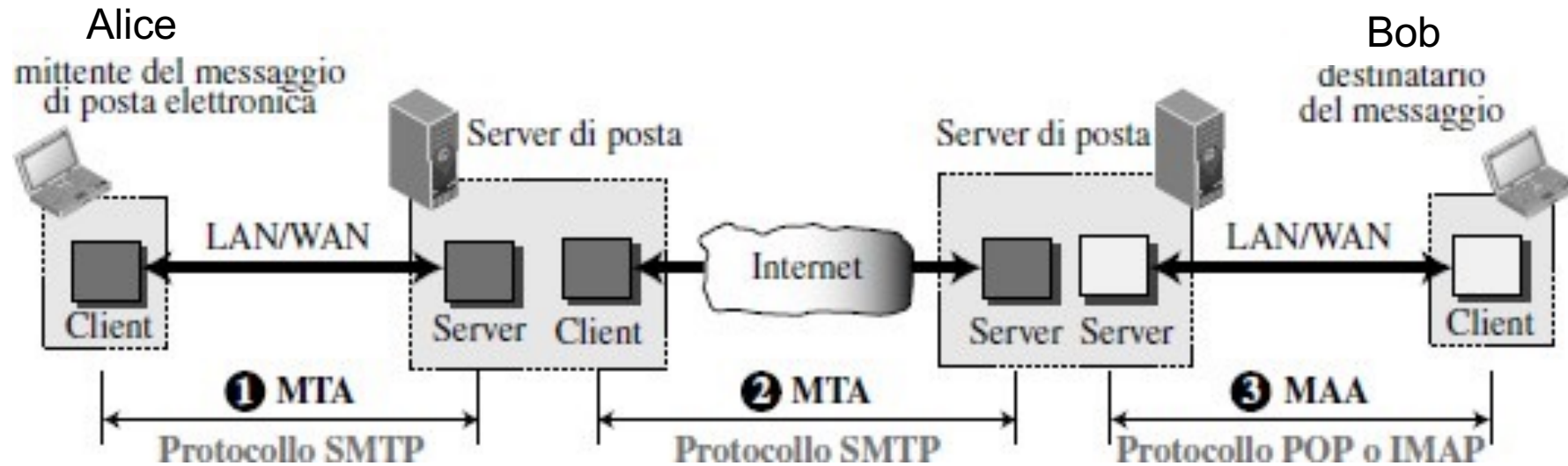
- usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client al server, porta 25
- trasferimento diretto: dal server trasmittente al server ricevente (no server intermedi)
- tre fasi per il trasferimento
 - ❖ Handshaking
 - ❖ trasferimento di messaggi
 - ❖ chiusura
- interazione comando/risposta
 - ❖ **comandi**: testo ASCII
 - ❖ **risposta**: codice di stato ed espressione
- i messaggi devono essere nel formato ASCII

Scenario: Alice invia un messaggio a Bob

- 1) Alice usa il suo agente utente per comporre il messaggio da inviare "a" bob@some school.edu
 - 2) L'agente utente di Alice invia un messaggio al server di posta di Alice; il messaggio è posto nella coda di messaggi
 - 3) Il lato client di SMTP apre una connessione TCP con il server di posta di Bob
 - 4) Il client SMTP invia il messaggio di Alice sulla connessione TCP
 - 5) Il server di posta di Bob riceve il messaggio e lo pone nella casella di posta di Bob
 - 6) Bob invoca il suo agente utente per leggere il messaggio
- N.B.: nessun server intermedio



Protocolli utilizzati



Scambio di messaggi al livello di protocollo

- Il client SMTP (che gira sull'host server di posta in invio) fa stabilire una connessione sulla porta 25 verso il server SMTP (che gira sull'host server di posta in ricezione)
- Se il server è inattivo il client riprova più tardi
- Se il server è attivo, viene stabilita la connessione
- Il server e il client effettuano una forma di handshaking (il client indica indirizzo email del mittente e del destinatario)
- Il client invia il messaggio
- Il messaggio arriva al server destinatario grazie all'affidabilità di TCP
- Se ci sono altri messaggi si usa la stessa connessione (**connessione persistente**), altrimenti il client invia richiesta di chiusura connessione

Esempio di interazione SMTP

Client: crepes.fr

Server: hamburger.edu

La seguente transazione inizia appena si stabilisce la connessione TCP

S: 220 hamburger.edu

C: HELO crepes.fr

S: 250 Hello crepes.fr, pleased to meet you

C: MAIL FROM: <alice@crepes.fr>

S: 250 alice@crepes.fr... Sender ok

C: RCPT TO: <bob@hamburger.edu>

S: 250 bob@hamburger.edu ... Recipient ok

C: DATA

S: 354 Enter mail, end with "." on a line by itself

C: Do you like ketchup?

C: How about pickles?

C: .

S: 250 Message accepted for delivery

C: QUIT

S: 221 hamburger.edu closing connection



Si ripete da qui
per mail multiple

Messaggio di posta

Esempio di interazione SMTP:

- sudo apt-get install postfix (con configurazione locale)

Si possono mandare email solo a \$USER@localhost (dove \$USER è un utente nella macchina).

- telnet localhost 25
- Riceverete la risposta **220** dal server
- Digitate i comandi HELO, MAIL FROM, RCPT TO, DATA, QUIT

Questo vi consente di inviare messaggi di posta elettronica senza usare il client di posta

- Dopo aver visto la programmazione socket potete implementare il vostro client di posta che manda email al server SMTP postfix

Esempio

```
35      # Connect to mail server
36      client_socket.connect((mail_server, 25))
37
```

⋮

```
59
60      # Get initial message from server
61      recv_and_check(220)
62
63      # Send greeting
64      send('HELO {}\n'.format(my_address.split('@')[1]))
65      recv_and_check()
66
67      # Set our address
68      send('MAIL FROM: {}\n'.format(my_address))
69      recv_and_check()
70
71      # Set their address
72      send('RCPT TO: {}\n'.format(their_address))
73      recv_and_check()
```

SMTP: note

- SMTP usa connessioni persistenti (ripete i passi da MAIL FROM:)
- SMTP richiede che il messaggio (intestazione e corpo) sia nel formato ASCII a 7 bit
- Il server SMTP usa CRLF.CRLF per determinare la fine del messaggio

Confronto con HTTP:

- HTTP e SMTP vengono utilizzati per **trasferire file** da un host all'altro
- HTTP: **pull** (gli utenti scaricano i file e inizializzano le connessioni TCP)
- SMTP: **push** (Il server di posta spedisce il file e inizializza la connessione TCP)
- HTTP: ciascun oggetto è incapsulato nel suo messaggio di risposta
- SMTP: più oggetti vengono trasmessi in un unico messaggio

Formato dei messaggi di posta elettronica

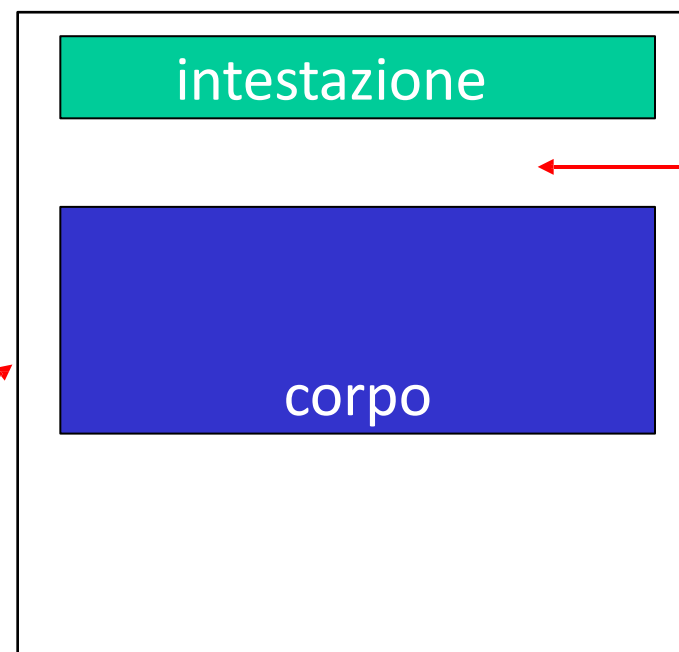
SMTP: protocollo per scambiare messaggi di posta elettronica

RFC 822: standard per il formato dei messaggi di testo:

- Righe di intestazione, per esempio
 - ❖ To:
 - ❖ From:
 - ❖ Subject:

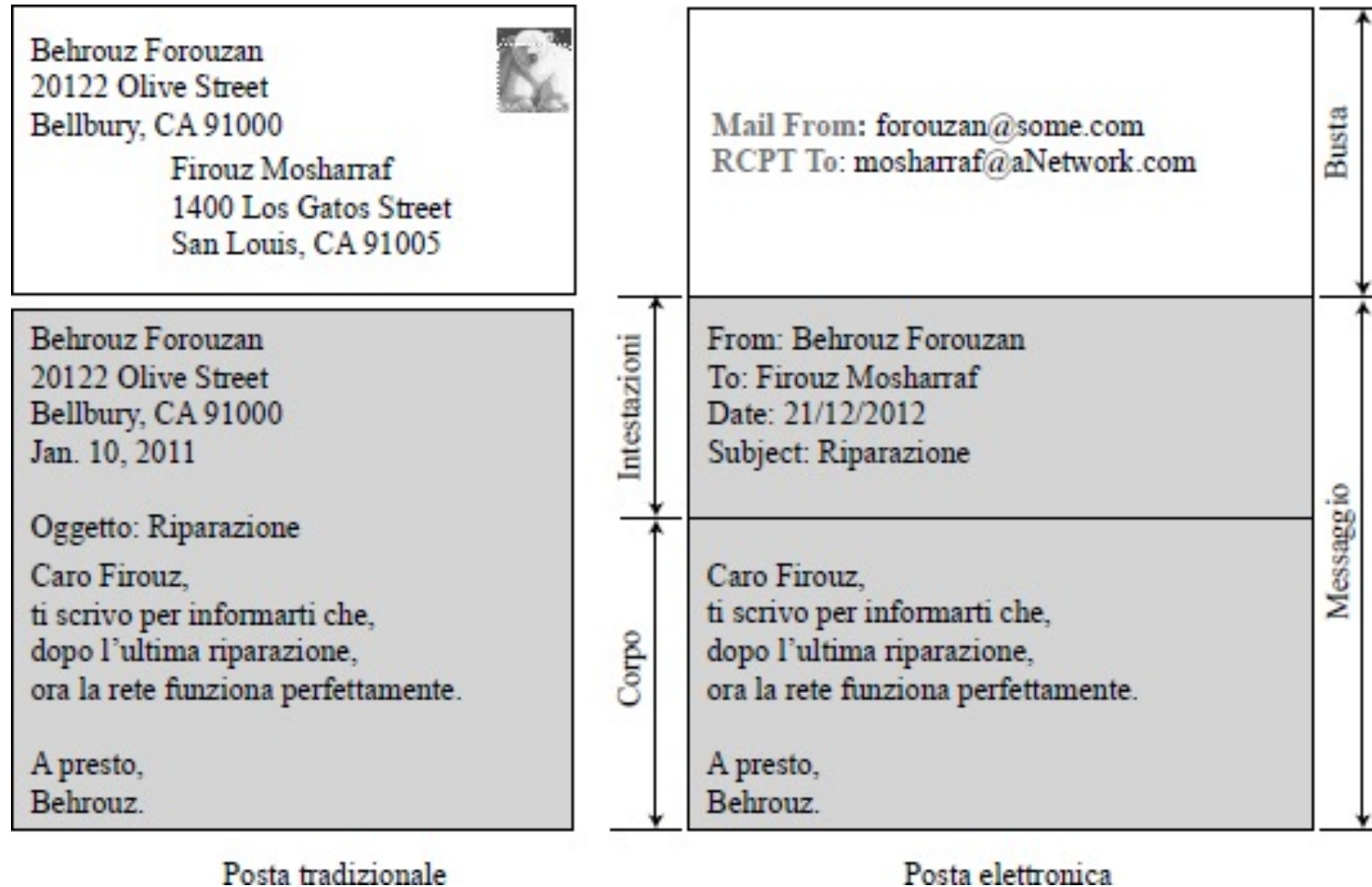
differenti dai comandi SMTP !
- corpo
 - ❖ il “messaggio”, soltanto caratteri ASCII

To	indirizzo di uno o più destinatari.
From	indirizzo del mittente.
Cc	indirizzo di uno o più destinatari a cui si invia per conoscenza.
Bcc	blind Cc: gli altri destinatari non sanno che anche lui riceve il messaggio.
Subject	argomento del messaggio.
Sender	chi materialmente effettua l'invio (ad es. nome della segretaria).

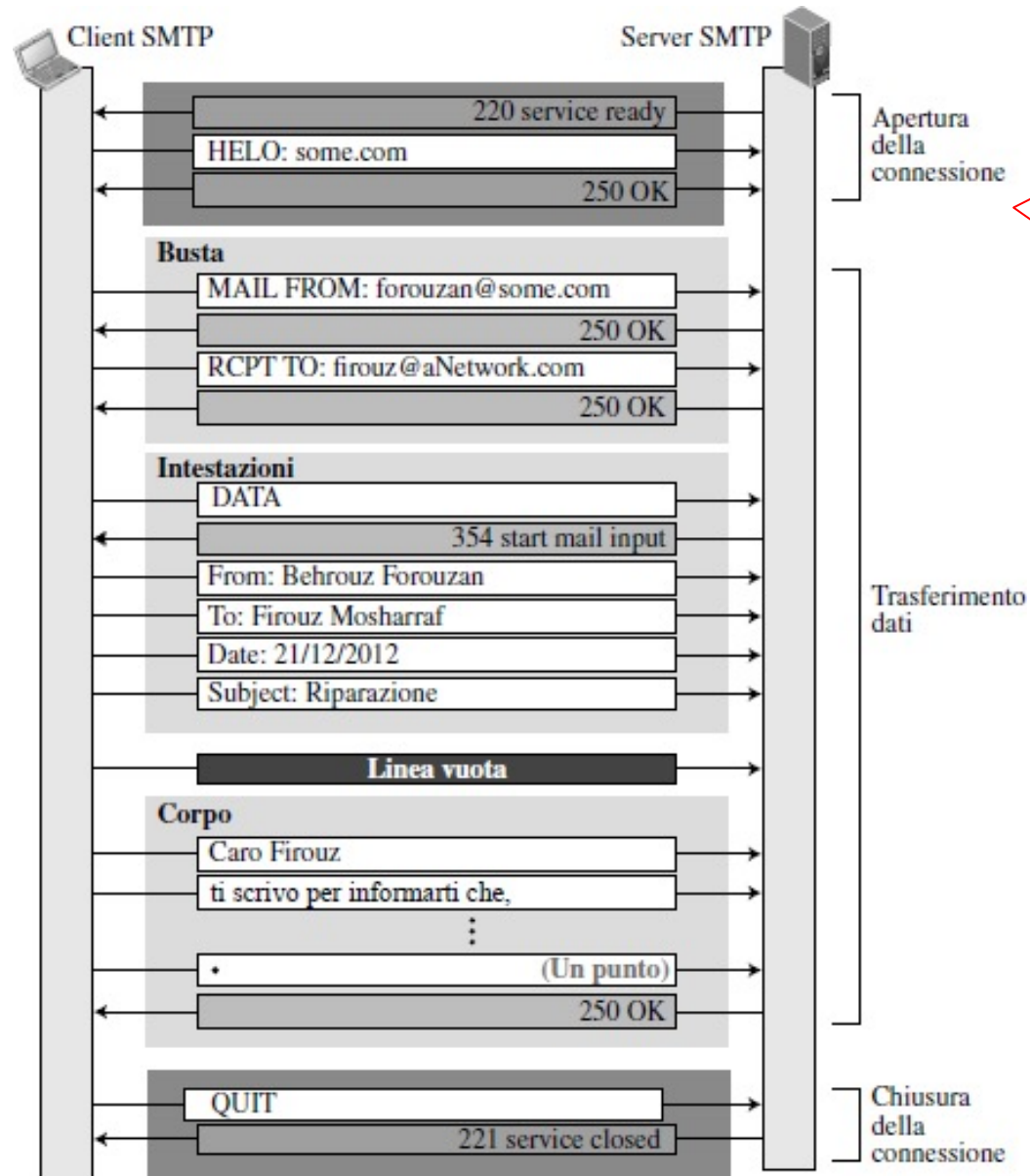


riga
vuota
(CRLF)

Esempio formato messaggio



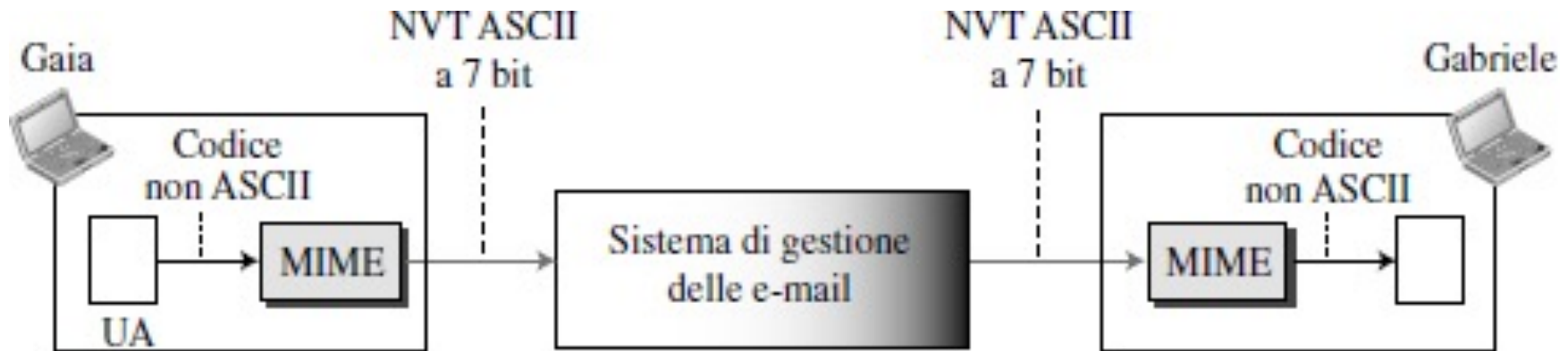
Esempio: fasi trasferimento



Apertura connessione SMTP è successiva all'apertura della connessione TCP

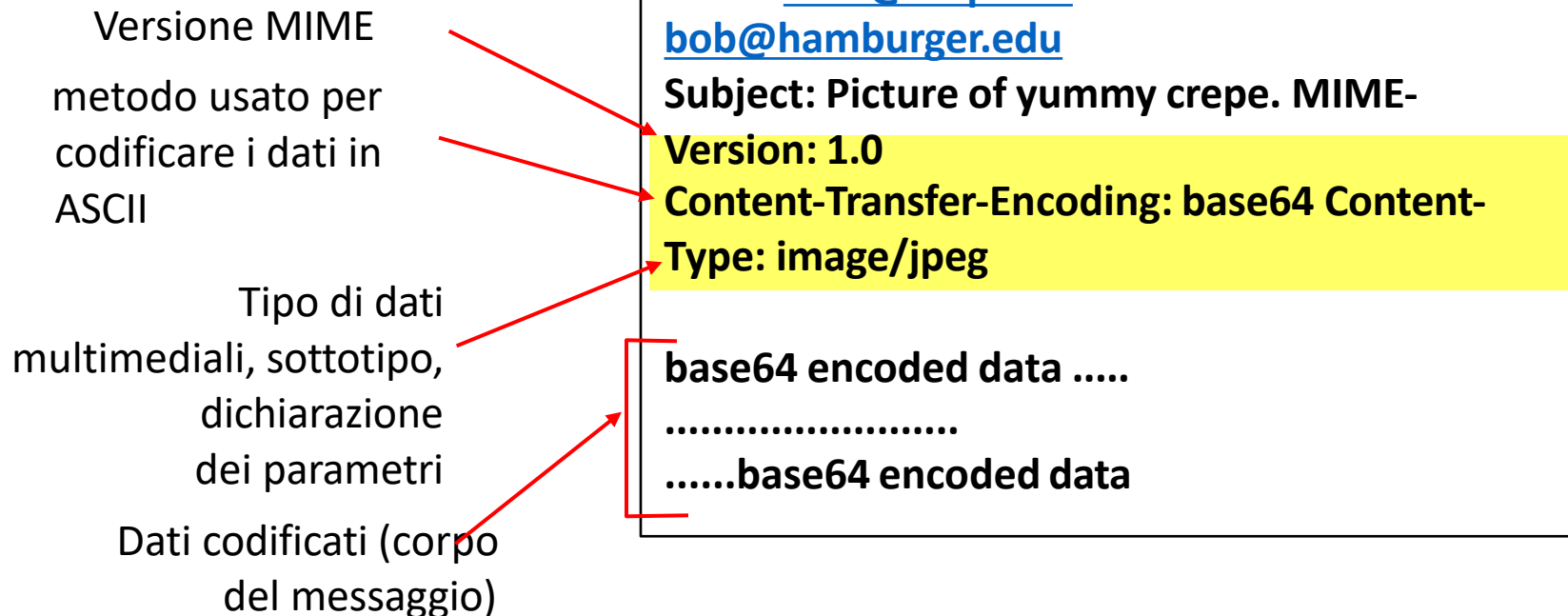
Protocollo MIME

- Come si possono inviare messaggi in formati non ASCII?
- Bisogna convertire i dati!



Formato del messaggio: estensioni di messaggi multimediali

- Per inviare contenuti diversi dal testo ASCII si usano intestazioni aggiuntive
- MIME (Multipurpose Internet Mail Extension): estensioni di messaggi di posta multimediali, RFC 2045, 2046
- Alcune righe aggiuntive nell'intestazione dei messaggi dichiarano il tipo di contenuto MIME



Formato del messaggio ricevuto

- Un'altra classe di righe di intestazione viene inserita dal server di ricezione SMTP
- ES. Il server di ricezione aggiunge **Received:** in cima al messaggio, che specifica il nome del server che ha inviato il messaggio (from), il nome del server che lo ha ricevuto (by), e l'orario di ricezione

Received: from crepes.fr by hamburger.edu; 12 Oct 98 15:27:39 GMT

From: alice@crepes.fr **To:**

bob@hamburger.edu

Subject: Picture of yummy crepe.

MIME-Version: 1.0

Content-Transfer-Encoding: base64

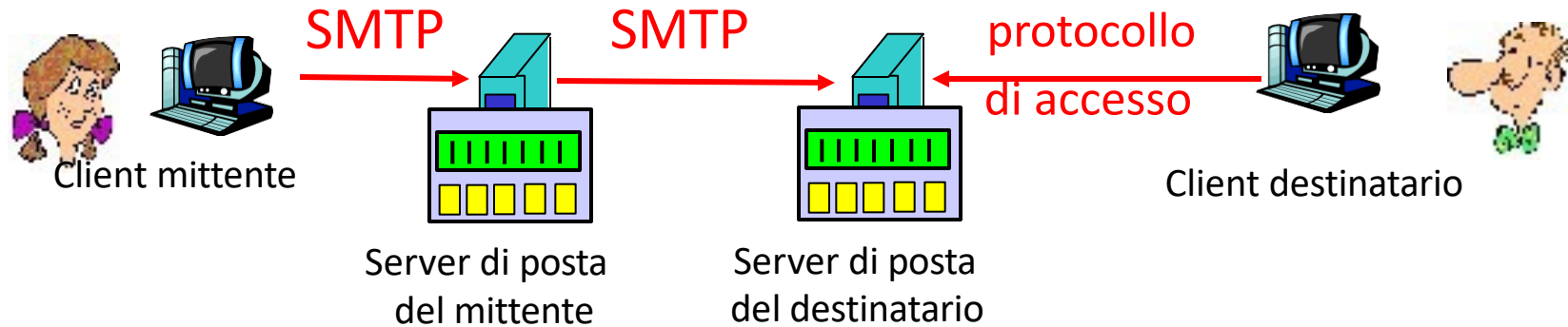
Content-Type: image/jpeg

base64 encoded data

.....

.....base64 encoded data

Protocolli di accesso alla posta



- SMTP: consegna/memorizzazione sul server del destinatario
- SMTP non può essere usato dall'agente utente del destinatario perché è un protocollo push, mentre l'utente deve eseguire un'operazione pull
- Protocollo di accesso alla posta: ottenere i messaggi dal server
 - ❖ POP3: Post Office Protocol – versione 3 [RFC 1939]
 - autorizzazione (agente <--> server) e download
 - ❖ IMAP: Internet Mail Access Protocol [RFC 1730]
 - più funzioni (più complesse)
 - manipolazione di messaggi memorizzati sul server
 - ❖ HTTP: gmail, Hotmail, Yahoo! Mail, ecc.


Protocollo POP3

- RFC 1939
- POP3 permette al client ricevente la posta di aprire una connessione TCP verso il server di posta sulla porta 110
- Quando la connessione è stabilita si procede in 3 fasi
 - 1. Autorizzazione:** L'agente utente invia nome utente e password per essere identificato
 - 2. Transazione:** L'agente utente recupera i messaggi
 - 3. Aggiornamento:** Dopo che il client ha inviato il QUIT, e quindi conclusa la connessione, vengono cancellati i messaggi marcati per la rimozione

Protocollo POP3: comandi

Fase di autorizzazione


- Comandi del client:
 - ❖ `user`: dichiara il nome dell'utente
 - ❖ `pass`: password
- Risposte del server
 - ❖ `+OK`
 - ❖ `-ERR`



```
S: +OK POP3 server ready
C: user rob
S: +OK
C: pass hungry
S: +OK user successfully logged on
```

Fase di transazione, client:

- `list`: elenca i numeri dei messaggi
- `retr`: ottiene i messaggi in base al numero
- `dele`: cancella
- `quit`



```
C: list
S: 1 498
S: 2 912
S: .
C: retr 1
S: <message 1 contents>
S: .
C: dele 1
C: retr 2
S: <message 1 contents>
S: .
C: dele 2
C: quit
S: +OK POP3 server signing off
```

POP3 e IMAP

Ancora su POP3

- ☐ Il precedente esempio usa la modalità “scarica e cancella”
- ☐ Bob non può rileggere le e-mail se cambia client
- ☐ Modalità “scarica e mantieni”: mantiene i messaggi sul server dopo averli scaricati
- ☐ POP3 è un protocollo senza stato tra le varie sessioni
- ☐ POP3 non fornisce all'utente alcuna procedura per creare cartelle remote ed assegnare loro messaggi, l'utente può crearle solo localmente al suo computer

IMAP: Internet Mail Access Protocol

- Cosa succede con POP3 se si accede alla mail da computer diversi?
 - ❖ Il server non mantiene le cartelle create localmente al proprio programma di posta
- IMAP
 - ❖ Mantiene tutti i messaggi in un unico posto: il server
 - ❖ Consente all'utente di organizzare i messaggi in cartelle
 - ❖ IMAP conserva lo stato dell'utente tra le varie sessioni:
 - I nomi delle cartelle e l'associazione tra identificatori dei messaggi e nomi delle cartelle

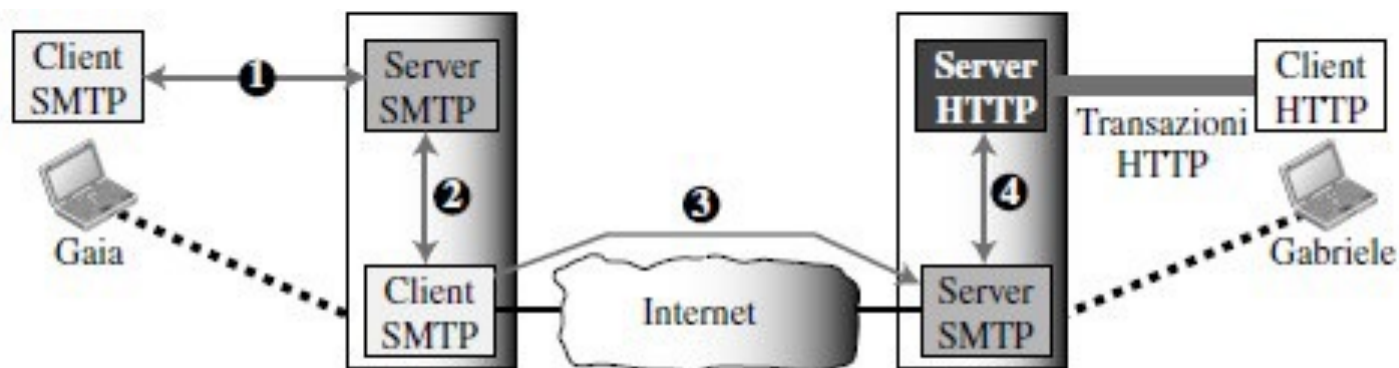
IMAP: Internet Mail Access Protocol

- RFC 3501
- Un server IMAP associa a una cartella (INBOX) ogni messaggio arrivato al server
- Il protocollo IMAP fornisce comandi agli utenti per
 - ❖ Creare cartelle e spostare messaggi da una cartella all'altra
 - ❖ Effettuare ricerche nelle cartelle remote
- I server IMAP conservano informazioni di stato sull'utente da una sessione all'altra
 - ❖ nomi cartelle
 - ❖ associazioni messaggi-cartelle

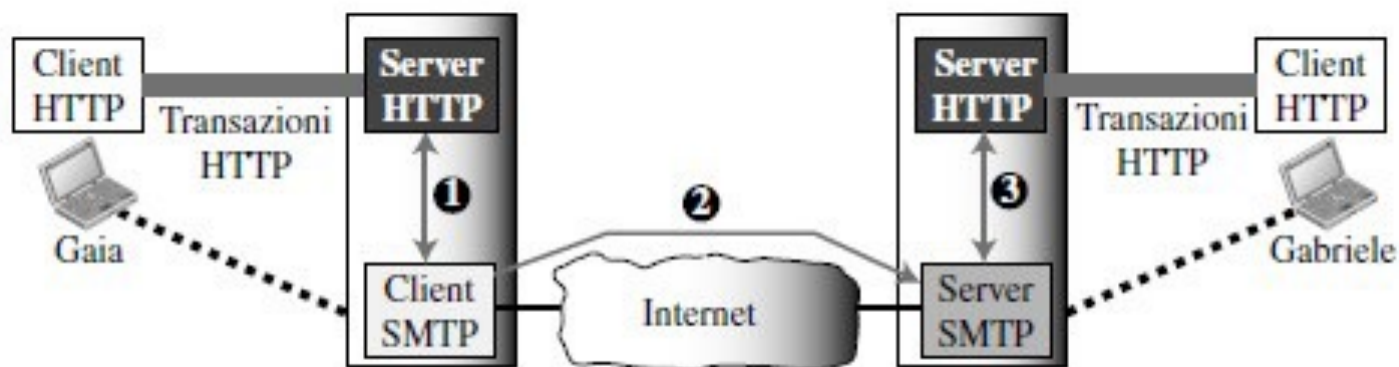
HTTP

- Alcuni mail server forniscono accesso alla mail via web (ovvero mediante il protocollo HTTP)
- Agente utente: web browser
- L'utente comunica con il suo mailbox mediante HTTP
- Il ricevente accede al suo mailbox mediante il protocollo HTTP
- SMTP rimane il protocollo di comunicazione tra mail server

webmail



Caso I: solo il destinatario utilizza HTTP



Caso II: il mittente e il destinatario utilizzano HTTP