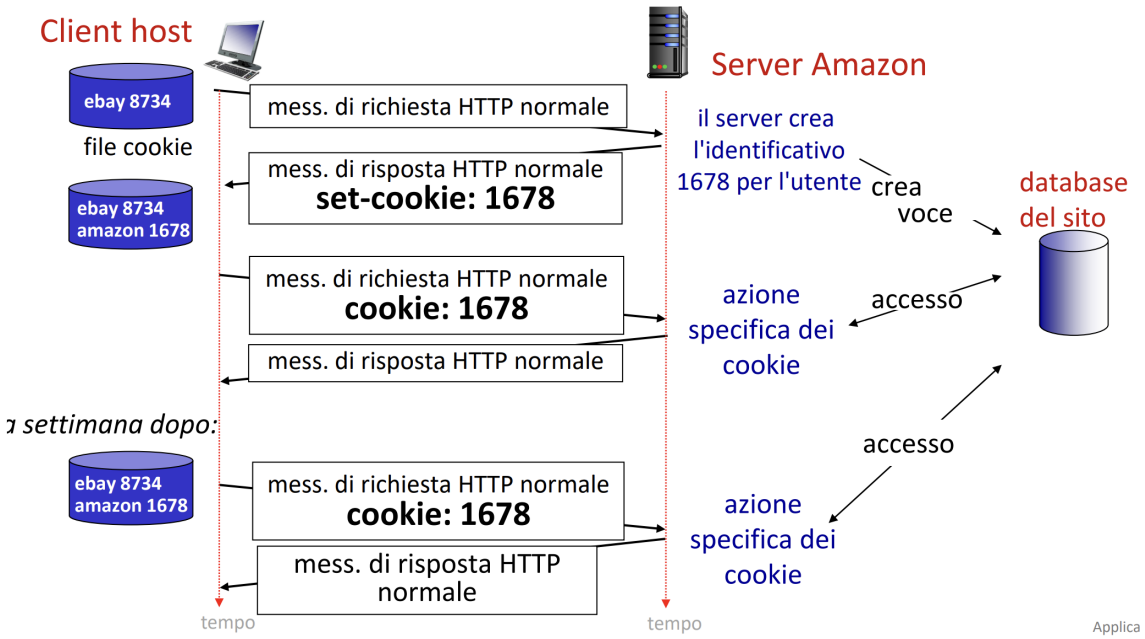
Lezione 5

Mantenere stato utente/server : i cookie

- Ricorda : l'iterazione HTTP GET/risposta è senza stato.
- nessuna nozione di scambio di messaggi HTTP in più fasi per completare una "transazione" Web.
    - non è necessario che il client o il server tengano traccia dello "stato" dello scambio in più fasi
    - tutte le richieste HTTP sono indipendenti l'una dall'altra
    - non è necessario che il client né il server siano in grado di "recuperare" da una transazione quasi completa ma mai completata.
- I siti web e il browser client usano i cookie per mantener dello stato tra le transazioni.
- Ci sono quattro componenti :
    1. Una riga di intestazione nel messaggio di *risposta* HTTP.
    2. Una riga di intestazione nel messaggio di *richiesta* HTTP.
    3. Un file cookie mantenuto sul sistema terminale dell'utente e gestito dal browser dell'utente.
    4. Un database sul sito.
- ***ESEMPIO :***
    - Susan usa il browser dal portatile , visita uno specifico sito di commercio elettronico per la prima volta
    - quando la richiesta HTTP iniziale arriva al sito , il sito crea
        - un identificativo unico
        - una voce nel proprio database , indicizzata dal numero identificativo
    - il server ritorna una risposta che include l'intestazione Set-cookie , che contiene l'identificativo unico e che sarà aggiunto

al file dei cookie

- le successive richieste del browser di Susan per questo sito conterranno l'identificativo in una intestazione cookie.



Cookie HTTP : commenti

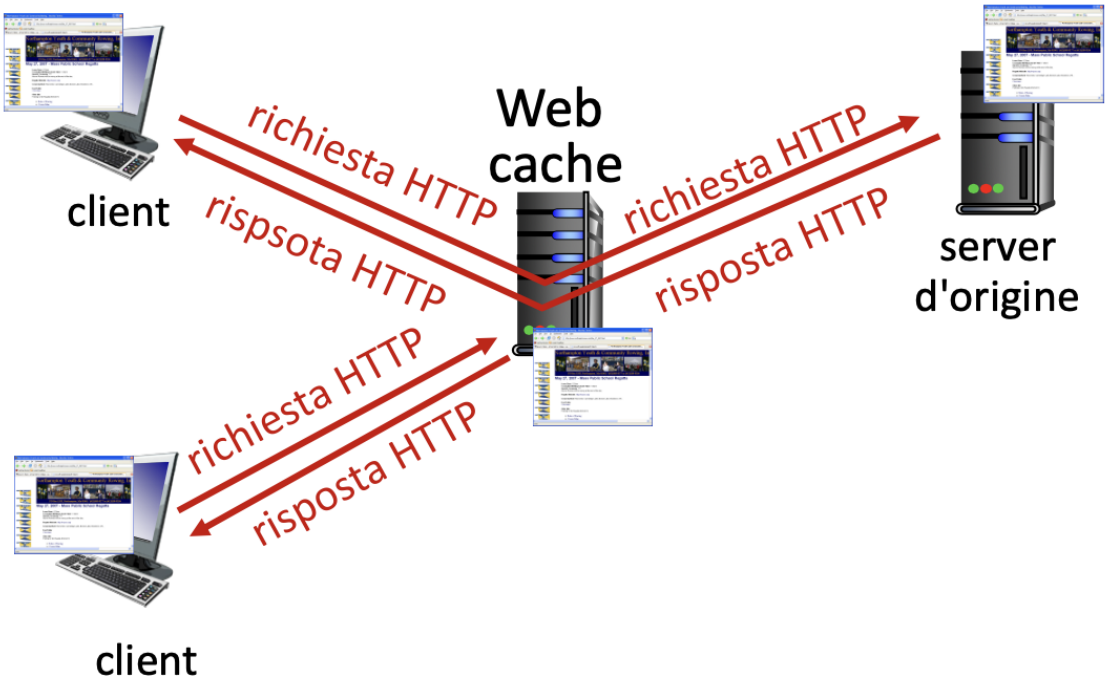
- I cookie possono essere usati per :
  - autorizzazioni
  - carrello degli acquisti
  - raccomandazioni
  - stato della sessione dell'utente
- *Nota , cookie e privacy:*
  - I cookie consentono ai siti di *imparare* molto di voi
  - cookie persistenti di terze parti (cookie di tracciamento) consentono il tracciamento di una identità comune attraverso siti web multipli.
- Come mantenere lo stato?
  - *presso gli endpoint del protocollo* : mantenere lo stato presso il trasmettitore e il ricevitore attraverso multiple transazioni
  - *nei messaggi :* i cookie trasportano lo stato nei messaggi HTTP

Cookie : tracciare il comportamento di navigazione di un utente

- I cookie possono essere usati per :
  - tracciare il comportamento degli utenti su un dato sito (Cookie di prima parte)
  - tracciare il comportamento degli utenti su più siti (Cookie di terze parti) senza neppure che l'utente abbia mai scelto di visitare il sito del tracker.
  - il tracciamento può essere *invisibile* all'utente :
    - piuttosto che un annuncio visualizzato che attiva HTTP GET al tracker , potrebbe essere un collegamento invisibile
  - disabilitato per impostazione predefinita nei browser Firefox e Safari
  - eliminazione graduale dei cookie di terze parti nel browser Chrome , inizialmente bloccati per l'1% degli utenti a partire da Gennaio 2024 , con l'obiettivo di estendere il blocco a tutti nel terzo trimestre del 2024.

Web cache

- **Obiettivo :** soddisfare la richiesta del client senza coinvolgere il server d'origine (origin server).
- l'utente configura il browser per usare una Web cache (locale)
- Il browser trasmette tutte le richieste HTTP alla cache
  - se l'oggetto è nella cache: la cache fornisce l'oggetto al client
  - altrimenti la cache richiede l'oggetto al server d'origine, memorizza ("cache") l'oggetto ricevuto, e infine lo restituisce al client

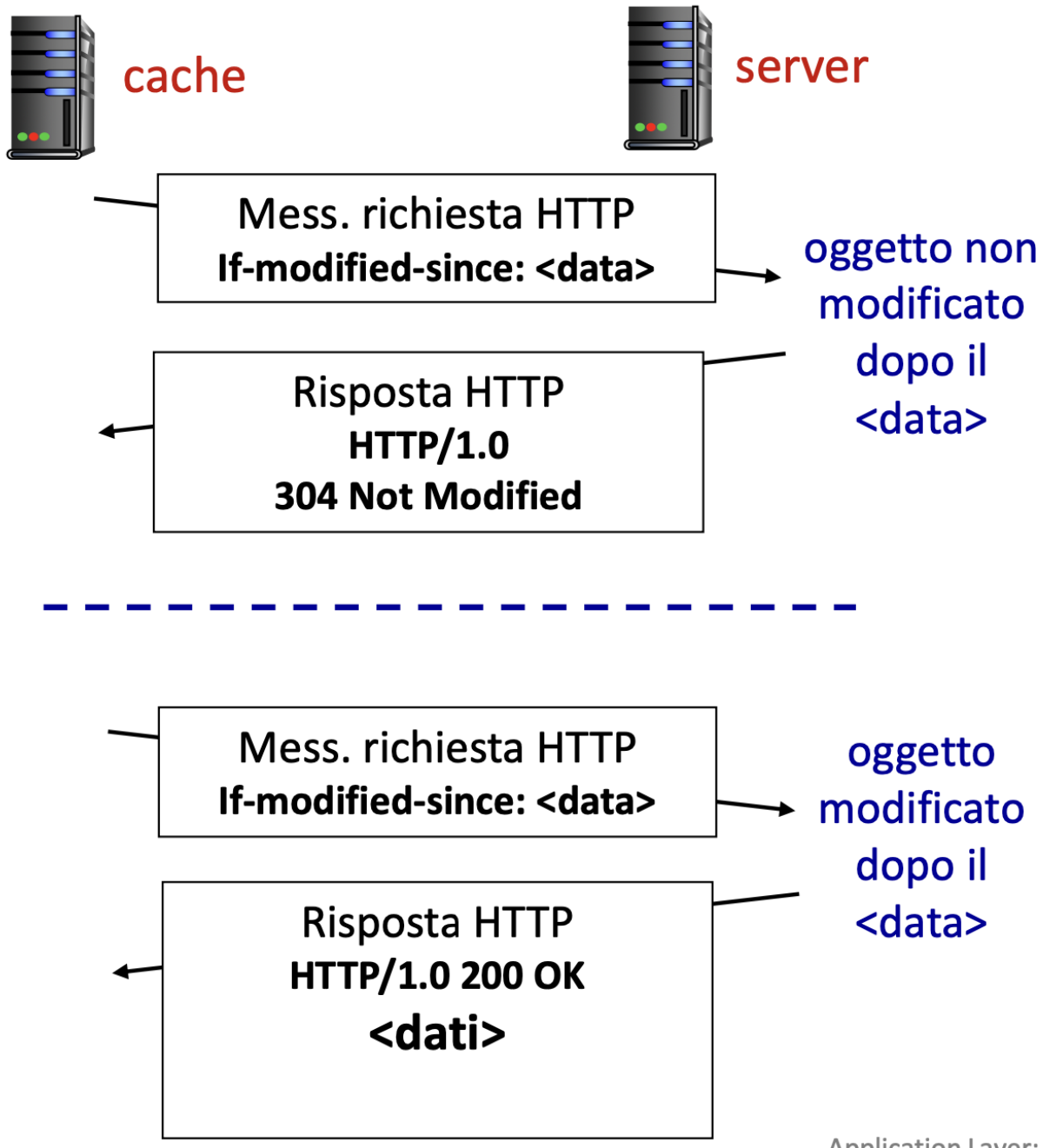


Web cache (server proxy)

- la cache opera come client (per il server d'origine) e come server (per il client originale)
- Il server comunica alla cache la cache consentita dell'oggetto nell'intestazione della risposta:
  - `Cache-Control : max-age=<seconds>`
    `Cache-Control : no-cache`
- Perché il web caching?
  - riduce i tempi di risposta alle richieste dei client
    - la cache è più vicina ai client
  - riduce il traffico sul collegamento di accesso a Internet istituzionale
  - Internet è ricca di cache
    - consente ai provider “scadenti” di fornire dati con efficacia

GET condizionale

- ***Obiettivo :*** non inviare un oggetto se la cache ha una copia aggiornata dell'oggetto
    - Nessun ritardo di trasmissione dell'oggetto (o uso delle risorse di rete)
- client: specifica la data della copia dell'oggetto nella richiesta HTTP
  `If-modified-since: <data>`
- server: la risposta non contiene l'oggetto se la copia nella cache è aggiornata:
  `HTTP/1.0 304 Not Modified`



Nota sul caching

Il caching può essere effettuato da:

- una web cache, ossia uno speciale tipo di proxy, cui il browser invia le richieste invece che indirizzarle all'origin server.
- oppure, dal browser stesso, che conserva una copia degli oggetti richiesti in precedenza

In entrambi i casi, occorre prestare attenzione al problema dell'aggiornamento degli oggetti: vedi riga di intestazione Cache-Control e GET condizionale.

HTTP/2

Obiettivo principale: diminuzione del ritardo nelle richieste HTTP a più oggetti

HTTP1.1: ha introdotto GET multiple in pipeline su una singola connessione TCP

- il server risponde in ordine (FCFS: first-come-first-served scheduling) alle richieste GET
- con FCFS, oggetti piccoli possono dover aspettare per la trasmissione (head-of-line (HOL) blocking $[\text{blocco in testa alla coda}]$) dietro a uno o più oggetti grandi
- il recupero delle perdite (ritrasmissione dei segmenti TCP persi) blocca la trasmissione degli oggetti

HTTP/2: $[\text{RFC } 7540, 2015]$ maggiore flessibilità del server nell'invio di oggetti al client:

- metodi, codice di stato, maggior parte dei campi di intestazione inalterati rispetto a HTTP 1.1
- ordine di trasmissione degli oggetti richiesti basata su una priorità degli oggetti specificata dal client (non necessariamente FCFS)
- invio push al client di oggetti aggiuntivi, senza che il client li abbia richiesti
- dividere gli oggetti in frame, intervallare i frame per mitigare il blocco HOL

Da HTTP/2 a HTTP/3

HTTP/2 su una singola connessione TCP significa:

- il recupero dalla perdita di pacchetti blocca comunque tutte le trasmissioni di oggetti
- come in HTTP 1.1, i browser sono incentivati ad aprire più connessioni TCP parallele per ridurre lo stallo e aumentare il throughput complessivo
- nessuna sicurezza su una connessione TCP semplice
- HTTP/3: aggiunge sicurezza, controllo di errore e congestione per oggetto (più pipelining) su UDP
    - ulteriori informazioni su HTTP/3 trattando il livello di trasporto

EMAIL

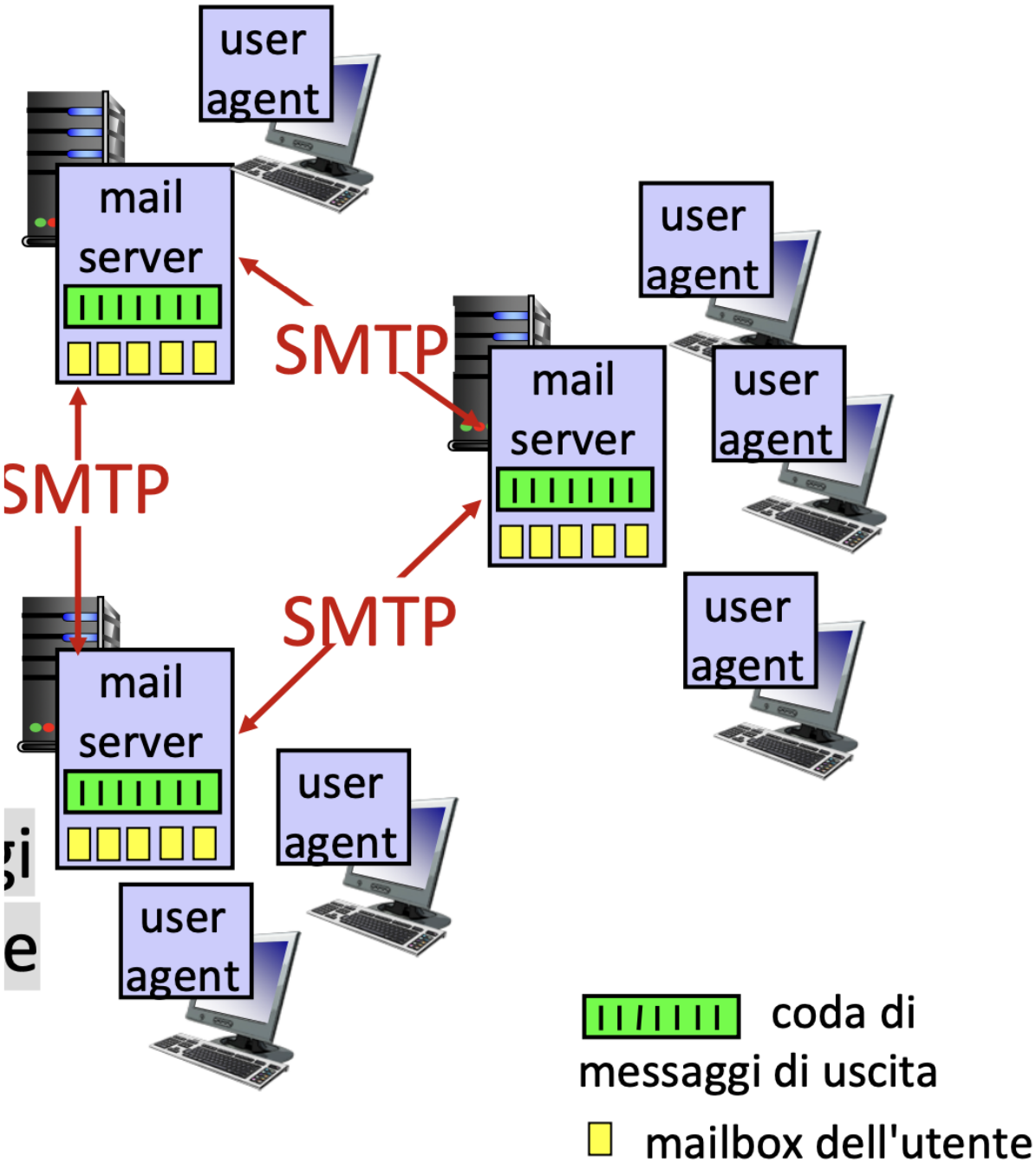
Tre componenti principali:

- user agents (o agenti utenti)
- mail servers (o server di posta)
- simple mail transfer protocol: SMTP

User Agent

- detto anche “mail reader”
- composizione, editing, lettura dei messaggi
- esempi: Outlook, client di posta dell'iPhone

- i messaggi in uscita o in arrivo sono memorizzati sul server



Mail servers

mail server :

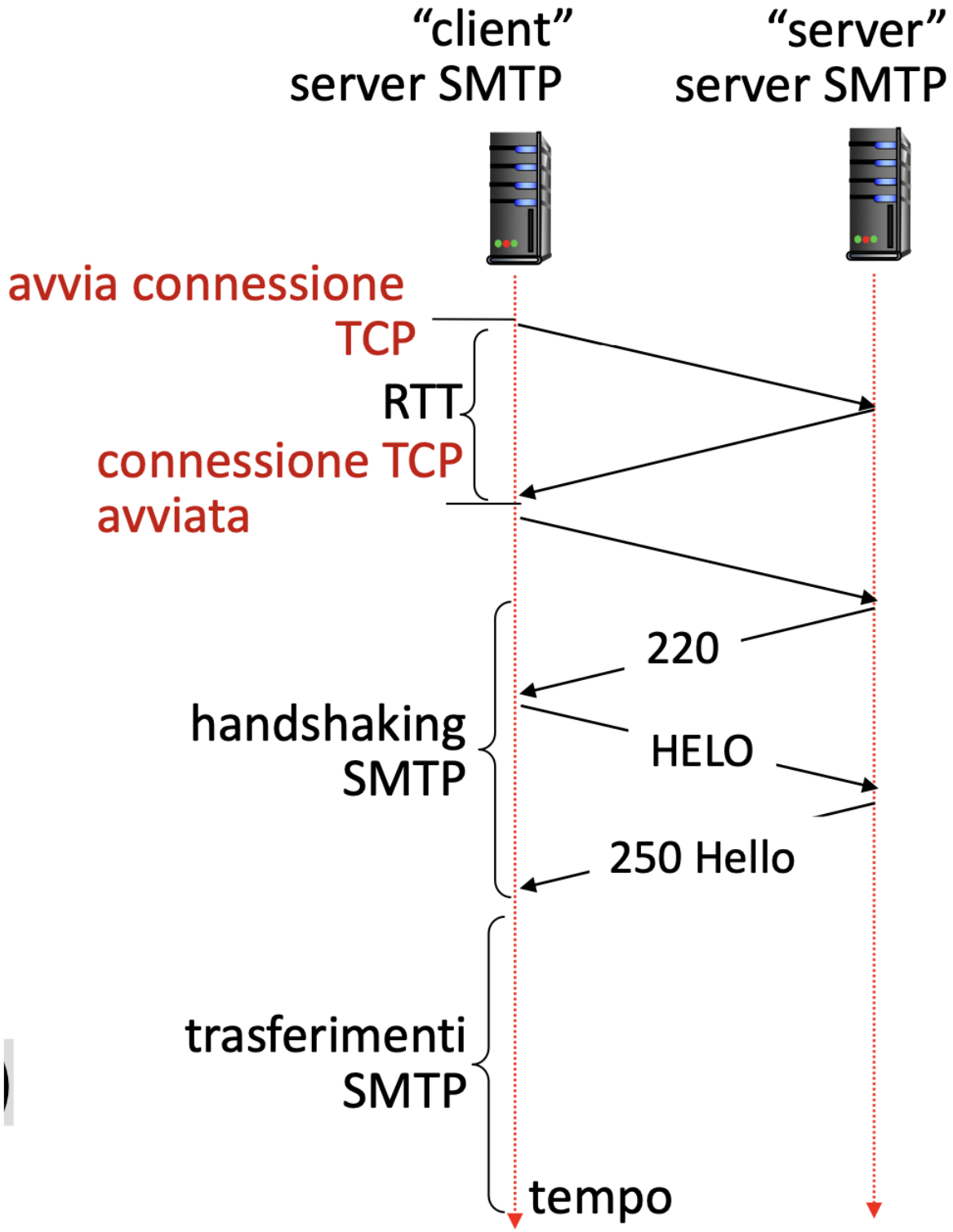
- *mailbox* (casella di posta) contiene i messaggi in arrivo per l'utente
- *coda di messaggi* da trasmettere

Protocollo SMTP tra mail server per inviare messaggi email

- *client :* mail server trasmittente
- *server :* mail server ricevente

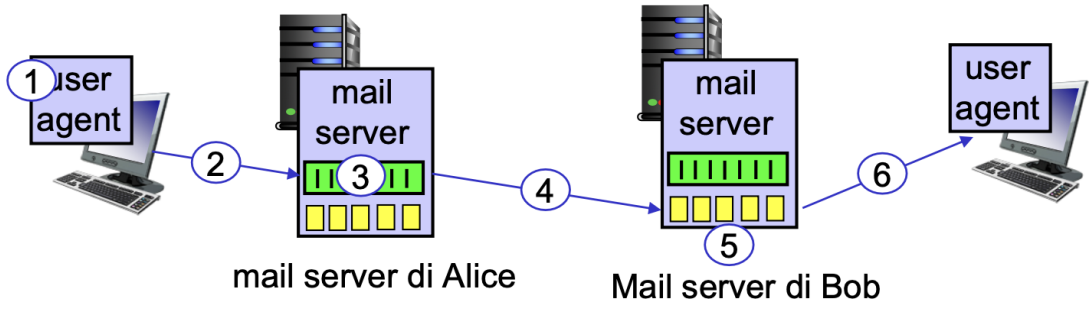
SMTP RFC

- usa TCP per trasferire in modo affidabile i messaggi di posta elettronica dal client (mail server che avvia la connessione) al server, porta 25
  - Trasferimento diretto: il server trasmittente al server ricevente
- Tre fasi per il trasferimento
  - handshaking (saluto)
  - trasferimento dei messaggi
  - chiusura
- Interazione comando/risposta (come HTTP)
  - comandi: testo ASCII a 7 bit
  - risposta: codice di stato e espressione



- Scenario :
  1. Alice usa il suo user agent per comporre il messaggio da inviare “a” ("to") bob@someschool.edu
  2. Lo user agent 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 mail server di Bob
4. il client SMTP invia il messaggio di Alice sulla connessione TCP
5. il mail server di Bob pone il messaggio nella casella di posta di Bob
6. Bob invoca il suo user agent per leggere il messaggio



Esempio di interazione SMTP

```
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
```

SMTP : note finali

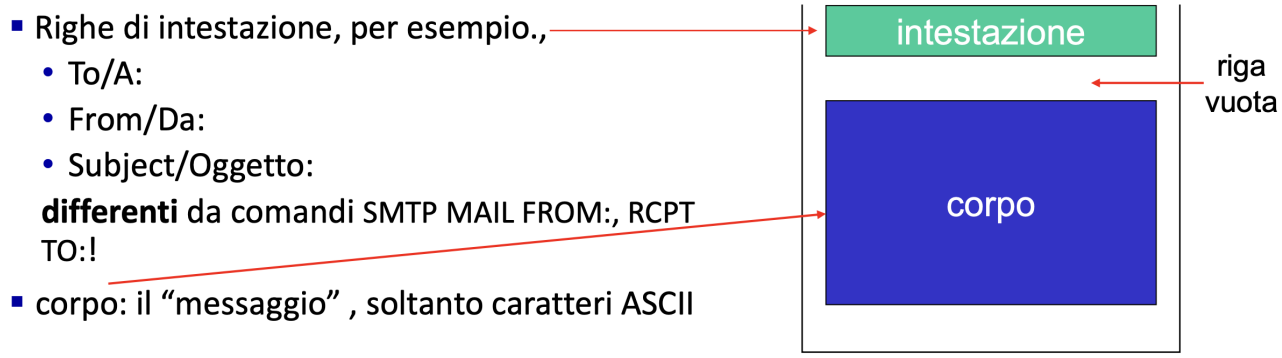
confronto con HTTP :

- HTTP : client push
- SMTP : client push

- Entrambi hanno un'iterazione comando/risposta in ASCII , codici di stato
- HTTP : ciascun oggetto è incapsulato nel suo messaggi di risposta
- SMTP : più oggetti vengono trasmessi in un unico messaggio
- SMTP usa connessione persistenti
- 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

Formato dei messaggi di posta elettronica

SMTP: protocollo per scambiare messaggi di posta elettronica, definito nell'RFC 5321 (come RFC 7231 definisce HTTP)
RFC 2822 definisce la sintassi dei messaggi di posta elettronica (come HTML definisce la sintassi per i documenti web)



Protocolli di accesso alla posta

- SMTP : consegna/memorizzazione sul server del destinatario
- protocollo di accesso alla posta : ottenere i messaggi dal server
  - IMAP : Internet Mail Access Protocol (RFC 3501) , messaggi memorizzati sul server , consente di recuperare , cancellare e archiviare i messaggi memorizzati sul server.

- HTTP : gmail , Hotmail , ecc... consente interfaccia Web sopra a STMP (per l'invio) e IMAP (o POP) per il recupero delle email.