Università degli Studi di Roma "Tor Vergata"
Laurea in Informatica

Sistemi Operativi e Reti
(modulo Reti)
a.a. 2023/2024

Livello di applicazione (parte3)

dr. Manuel Fiorelli

manuel.fiorelli@uniroma2.it

https://art.uniroma2.it/fiorelli

Basate sulle slide del libro di testo:
https://gaia.cs.umass.edu/kurose_ross/ppt.php

Livello di applicazione: panoramica

- Principi delle applicazioni di rete
- Web e HTTP
- E-mail, SMTP, IMAP
- DNS: il servizio di directory di Internet
- Applicazioni P2P
- Streaming video e reti di distribuzione di contenti
- Programmazione delle socket programming con UDP e TCP

Problema: risoluzione dei nomi

persone: molti identificatori:

- Nome, codice fiscal, numero della carta di identità

Host e router di Internet:

- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- "nome", ad esempio, cs.umass.edu - usato dagli esseri umani

D: come mappare tra indirizzo IP e nome e viceversa?

Problema: risoluzione dei nomi

persone: molti identificatori:

- Nome, codice fiscal, numero della carta di identità

Host e router di Internet:

- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- "nome", ad esempio, cs.umass.edu - usato dagli esseri umani

D: come mappare tra indirizzo IP e nome e viceversa?

File *hosts* (/etc/hosts nei sistemi POSIX)

Associa un indirizzo IP a uno o più hostname

```
185.300.10.1   host1
185.300.10.2   host2 merlin
185.300.10.3   host3 arthur king
185.300.10.4   timeserver
```

Locale a un nodo, il suo contenuto non deve necessariamente coincidere con quello di altri nodi (ma meglio evitarlo!)

Problema: risoluzione dei nomi

persone: molti identificatori:

- Nome, codice fiscal, numero della carta di identità

Host e router di Internet:

- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- "nome", ad esempio, cs.umass.edu - usato dagli esseri umani

D: come mappare tra indirizzo IP e nome e viceversa?

Anni '70: HOSTS.TXT

- *Mantenuto dal Network Information Center (NIC) presso lo SRI*
- *Reso disponibile su un host designato* (attraverso il protocollo FTP)
- *Installato dagli amministratori di sistema sui singoli nodi*

Problemi:

- *Crescita del file*
- *Traffico generato sull'host dove era pubblicato*

Fonte: https://www.oreilly.com/library/view/dns-on-windows/0596005628/ch01s02s01.html

DNS: Domain Name System

persone: molti identificatori:

- nome, codice fiscale, numero della carta di identità

Host e router di Internet:

- indirizzo IP (32 bit) - usato per indirizzare i datagrammi
- "nome", ad esempio, cs.umass.edu - usato dagli esseri umani

D: come mappare tra indirizzo IP e nome e viceversa?

Domain Name System (DNS):

- *Database distribuito* implementato in una gerarchia di *name server*
- *Protocollo a livello di applicazione* che consente agli host, ai router e ai server DNS di comunicare per *risolvere* i nomi (traduzione nome/indirizzo)
  - *si noti:* funzioni critiche di Internet, implementate come protocollo a livello di applicazione
  - complessità nelle parti periferiche della rete

DNS: servizi, struttura

Servizi DNS

- Traduzione degli hostname in indirizzi IP
- host aliasing
  - nome canonico e alias
- mail server aliasing
- load distribution (*distribuzione del carico di rete*)
  - server Web replicati: più indirizzi IP corrispondono a un solo nome

Q: Perché non centralizzare il DNS?

- un *single point of failure* (*punto di vulnerabilità*)
- volume di traffico
- database centralizzato distante
- manutenzione

R: non scala!

- Solo i server DNS di Comcast: 600B query DNS al giorno
- Solo i server DNS Akamai: 2,2T query DNS al giorno

Pensare al DNS

un enorme database distribuito:

- ~ miliardi di record, ciascuno semplice

gestisce molti *trilioni* di interrogazioni al giorno :

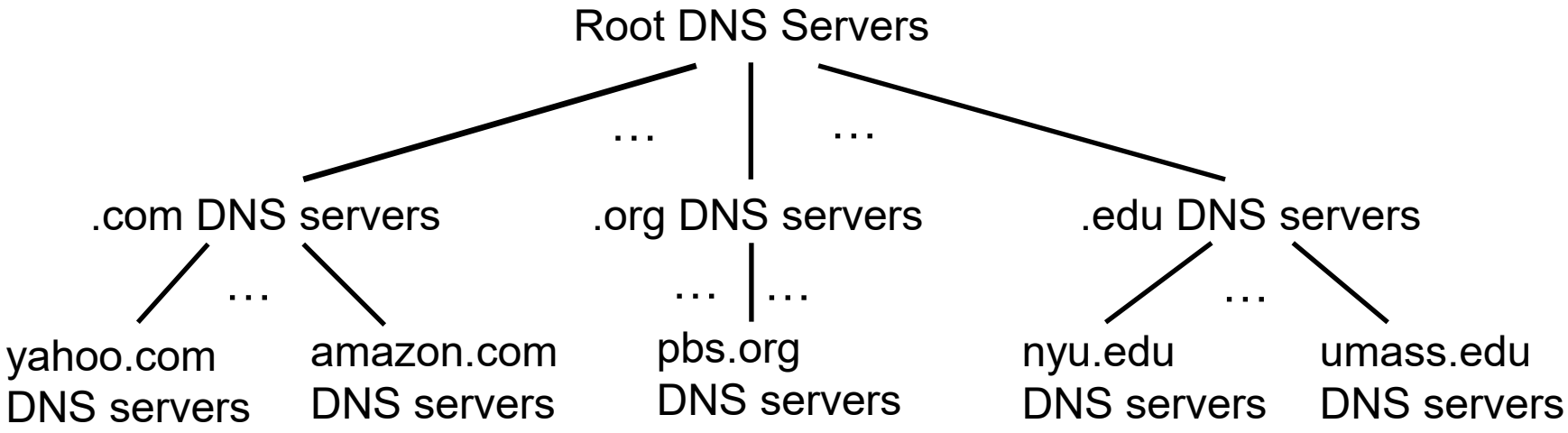
- *molte* più letture che scritture
- *è importante*: quasi tutte le transazioni Internet interagiscono con il DNS - i millisecondi contano!

organizzativamente e fisicamente decentralizzato

- milioni di organizzazioni diverse responsabili d loro *record*

"a prova di proiettile": affidabilità, sicurezza

DNS: un database distribuito e gerarchico



Root
(radice)

Top Level Domain

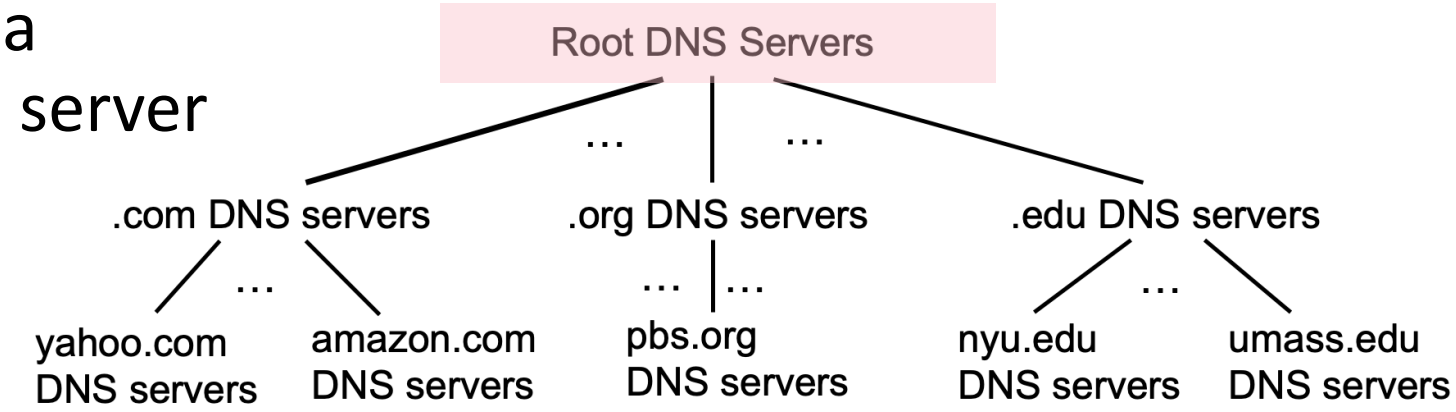
Authoritative
(server autoritativi)

Il client vuole l'indirizzo IP di www.amazon.com; 1ª approssimazione:

- il cliente interroga il root server per trovare il TLD server per .com
- il client interroga il TLD server .com per ottenere il server autoritativo per amazon.com
- il client interroga il server autoritativo per amazon.com per ottenere l'indirizzo IP di www.amazon.com

DNS: root name server

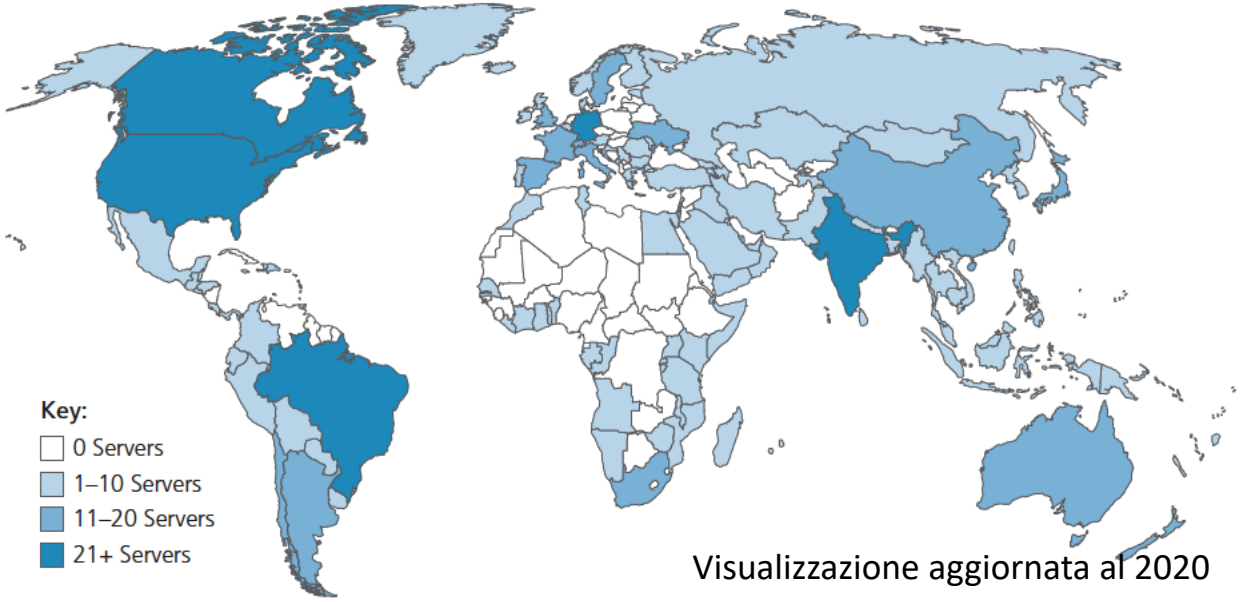
- ufficiale, contatto di ultima istanza da parte dei name server che non sono in grado di risolvere il nome



DNS: root name server

- ufficiale, contatto di ultima istanza da parte dei name server che non sono in grado di risolvere il nome.
  Fornisce gli indirizzi IP dei TLD server
- funzione *incredibilmente importante* di Internet
  - Internet non potrebbe funzionare senza!
  - DNSSEC – offre sicurezza (autenticazione, integrità dei messaggi)
- ICANN (Internet Corporation for Assigned Names and Numbers) gestisce il root DNS domain

13 name "server" logici in tutto il mondo, ogni "server" replicato più volte (~200 server negli USA)
https://www.internic.net/domain/named.root



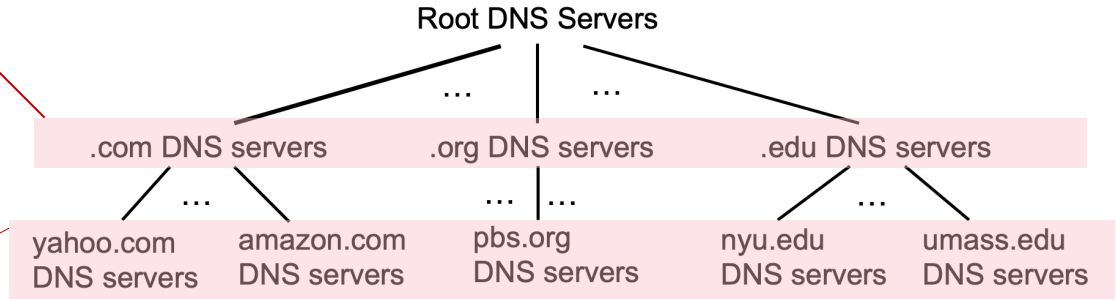
Visualizzazione aggiornata al 2020

Il 20/03/2023 ci sono 1813 istanze gestite da 12 operator, coordinate dallo IANA (fonte: https://root-servers.org/)

Top-Level Domain, and authoritative servers

Top-Level Domain (TLD) DNS server:

- si occupano dei domini .com, .org, .net, .edu, .aero, .jobs, .museums, e di tutti I domani locali di alto livello, quali .cn, .uk, .fr, .ca, .jp
- Network Solutions: gestisce i server TLD per I domini .com e .net
- Educause: gestisce quelli per .edu



DNS server autoritativo:

- server DNS propri di ciascuna organizzazione, che forniscono i mapping ufficiali da hostname a IP per gli host dell'organizzazione
- possono essere mantenuti dall'organizzazione o dal service provider

Name server DNS locali

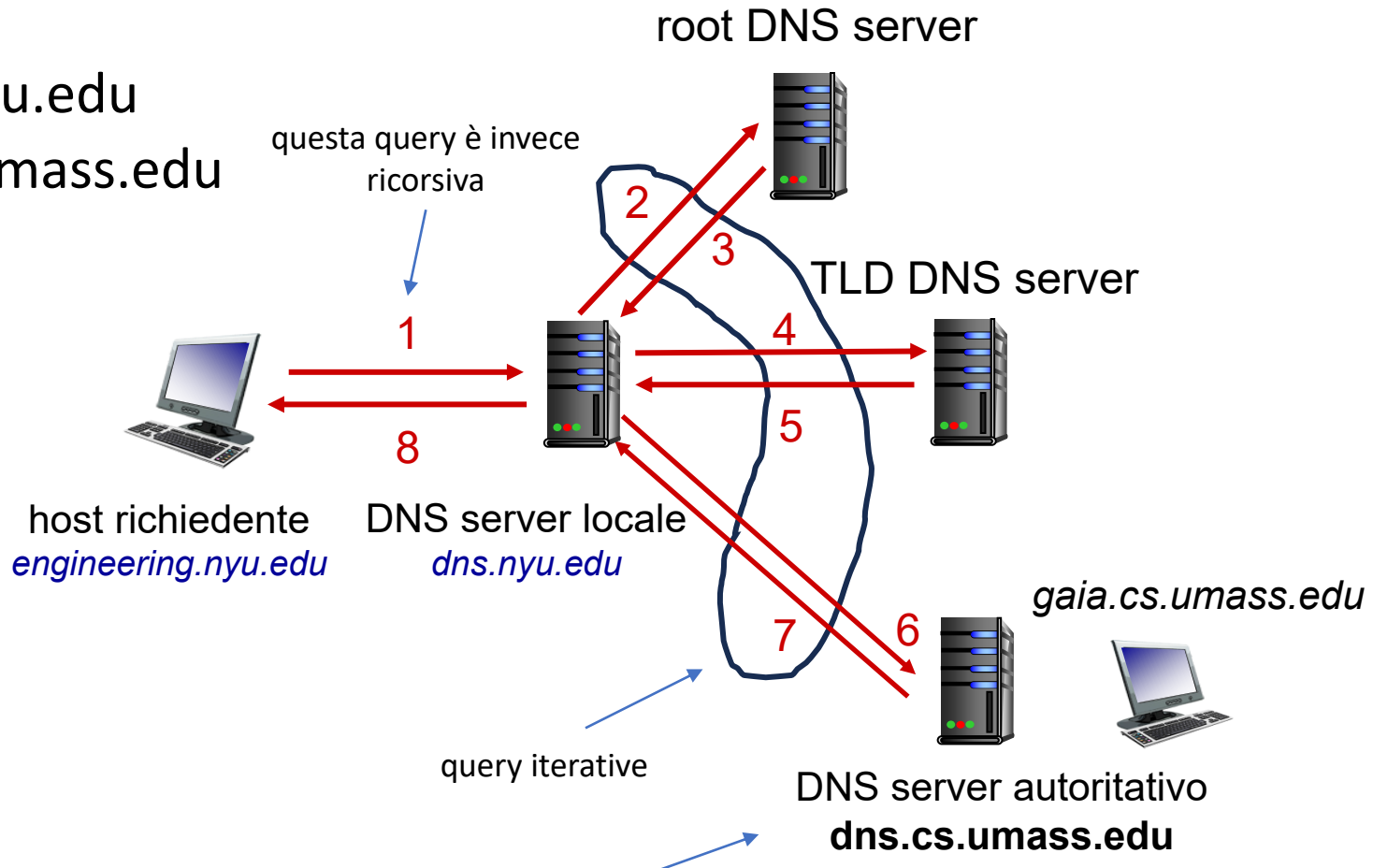
- quando l'host effettua una richiesta DNS, la query viene inviata al suo server DNS *locale* (con funzione di *default name server*)
  - il server DNS locale restituisce una risposta, rispondendo:
    - dalla sua cache locale di coppie nome->indirizzo (possibilmente non aggiornate!)
    - inoltrando la richiesta alla gerarchia DNS per la risoluzione
  - ciascun ISP ha un proprio server DNS locale; per trovare il vostro:
    - MacOS: `% scutil --dns`
    - Windows: `>ipconfig /all`
- il server DNS locale non appartiene strettamente alla gerarchia dei server

DNS: interrogazione iterativa

Esempio: l'host engineering.nyu.edu vuole l'indirizzo IP di gaia.cs.umass.edu

Query iterativa

- Il server contattato risponde con il nome del server da contattare
- "Io non conosco questo nome, ma puoi chiederlo a questo server"



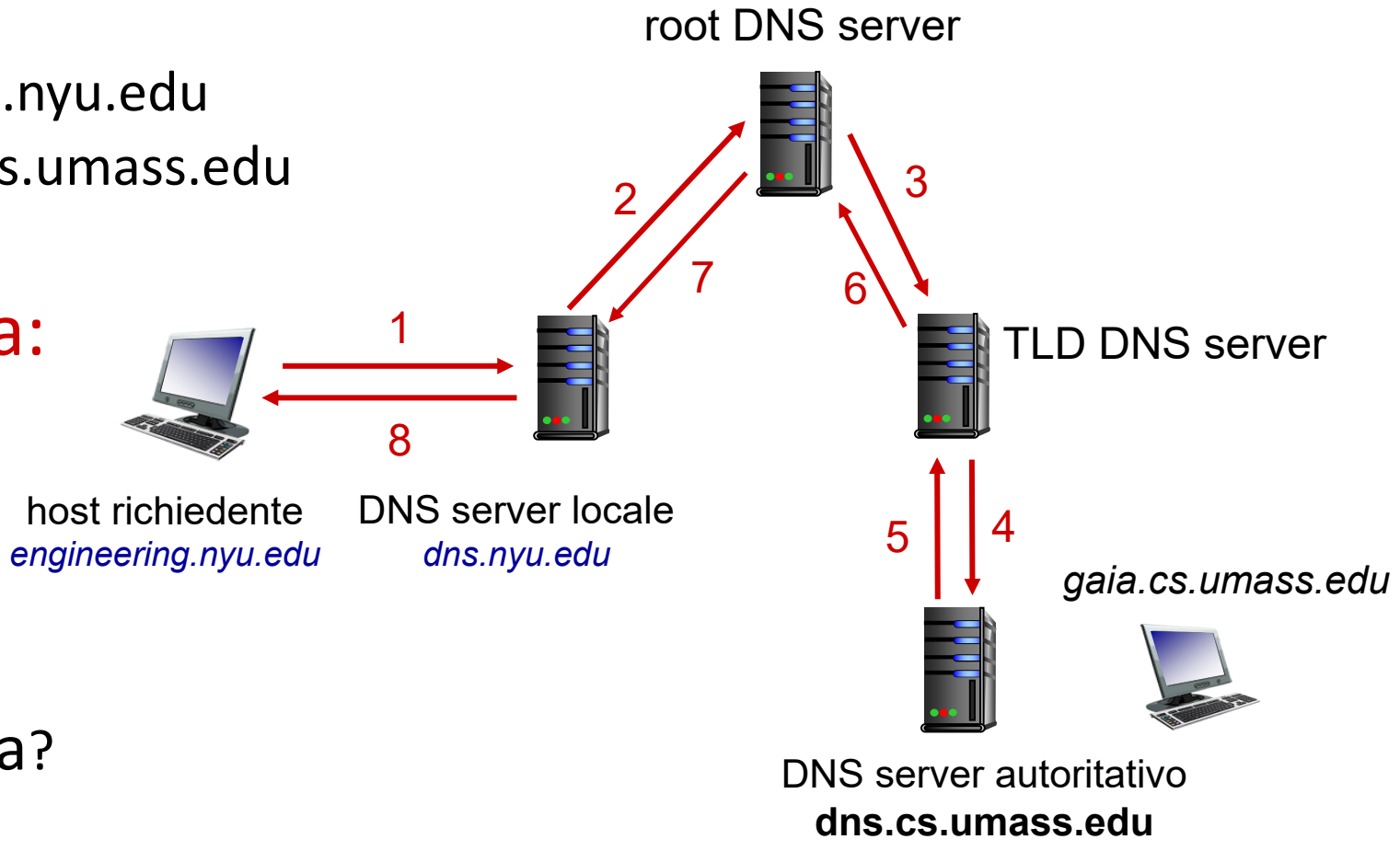
il TLD DNS server potrebbe conoscere in realtà un DNS server intermedio, ad esempio nel caso di domini di terzo livello aventi ciascuno il proprio DNS server autoritativo

DNS: interrogazione ricorsiva

Esempio: l'host engineering.nyu.edu vuole l'indirizzo IP di gaia.cs.umass.edu

Interrogazione ricorsiva:

- Affida il compito di tradurre il nome al server contattato
- carico pesante ai livelli superiori della gerarchia?



DNS: caching e aggiornamento dei record

- una volta che un (qualsiasi) name server impara la mappatura, la mette nella *cache*, e restituisce *immediatamente* il mapping nella cache in risposta a un query
  - il caching migliora i tempi di risposta
  - le voci della cache vanno in timeout (scompaiono) dopo un certo tempo (TTL)
  - i server TLD sono in genere memorizzati nella cache dei server dei nomi locali
- le voci nella cache potrebbero essere *obsolete*
  - se l'host con nome cambia il suo indirizzo IP, potrebbe non essere conosciuto su Internet fino alla scadenza di tutti i TTL!
  - *traduzione nome->indirizzo best-effort!*

Record DNS

DNS: database distribuito che memorizza 7 record di risorsa (RR)

Formato RR: (`name, value, type, ttl`)

type=A

- `name` è l'hostname
- `value` è l'indirizzo IP

type=NS

- `name` è il dominio (ad esempio, foo.com)
- `value` è l'hostname dell'authoritative name server per questo dominio

type=CNAME

- `name` è il nome alias di qualche nome "canonico" (nome vero)
- www.ibm.com è in realtà servereast.backup2.ibm.com
- `value` è il nome canonico

type=MX

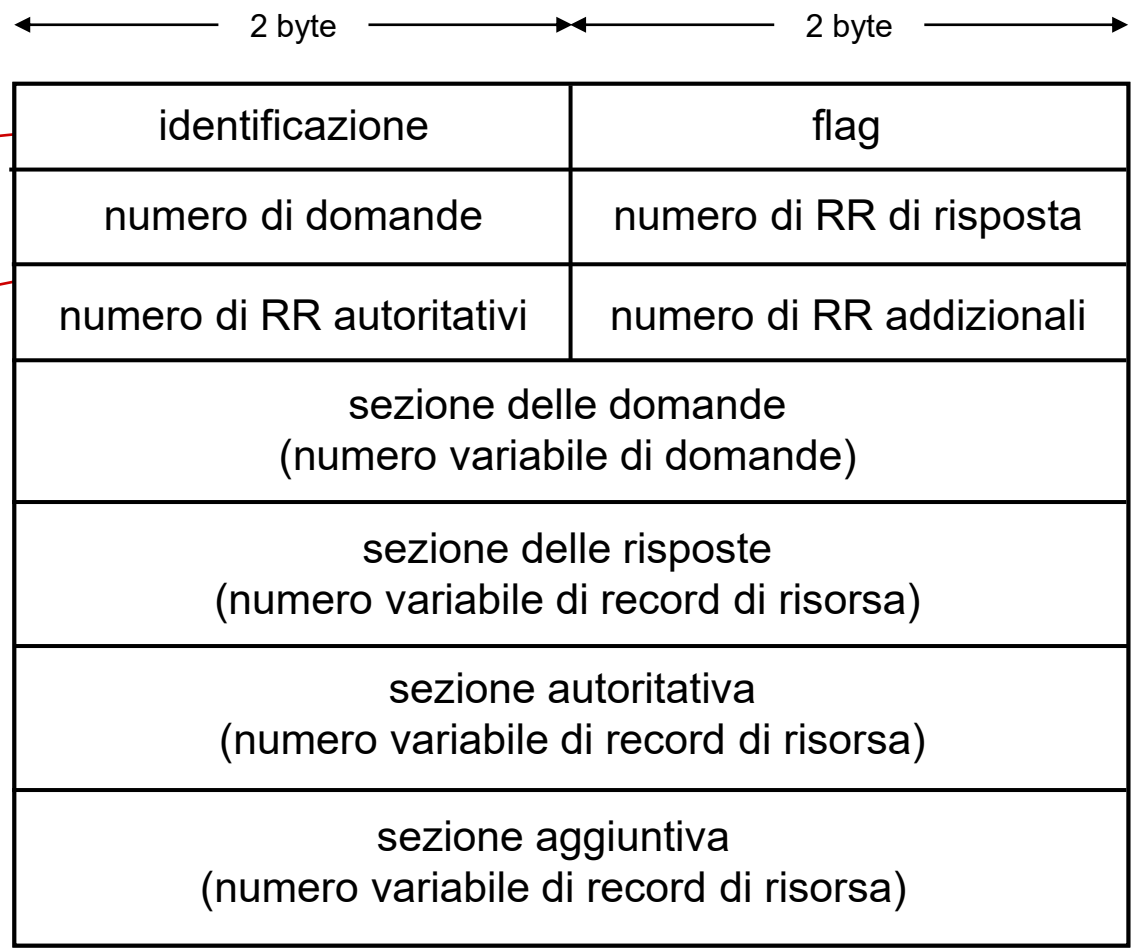
- `value` è il nome del server di posta associato a `name`

Messaggi DNS

domande (query) e messaggi di *risposta* (reply), entrambi con lo stesso formato:

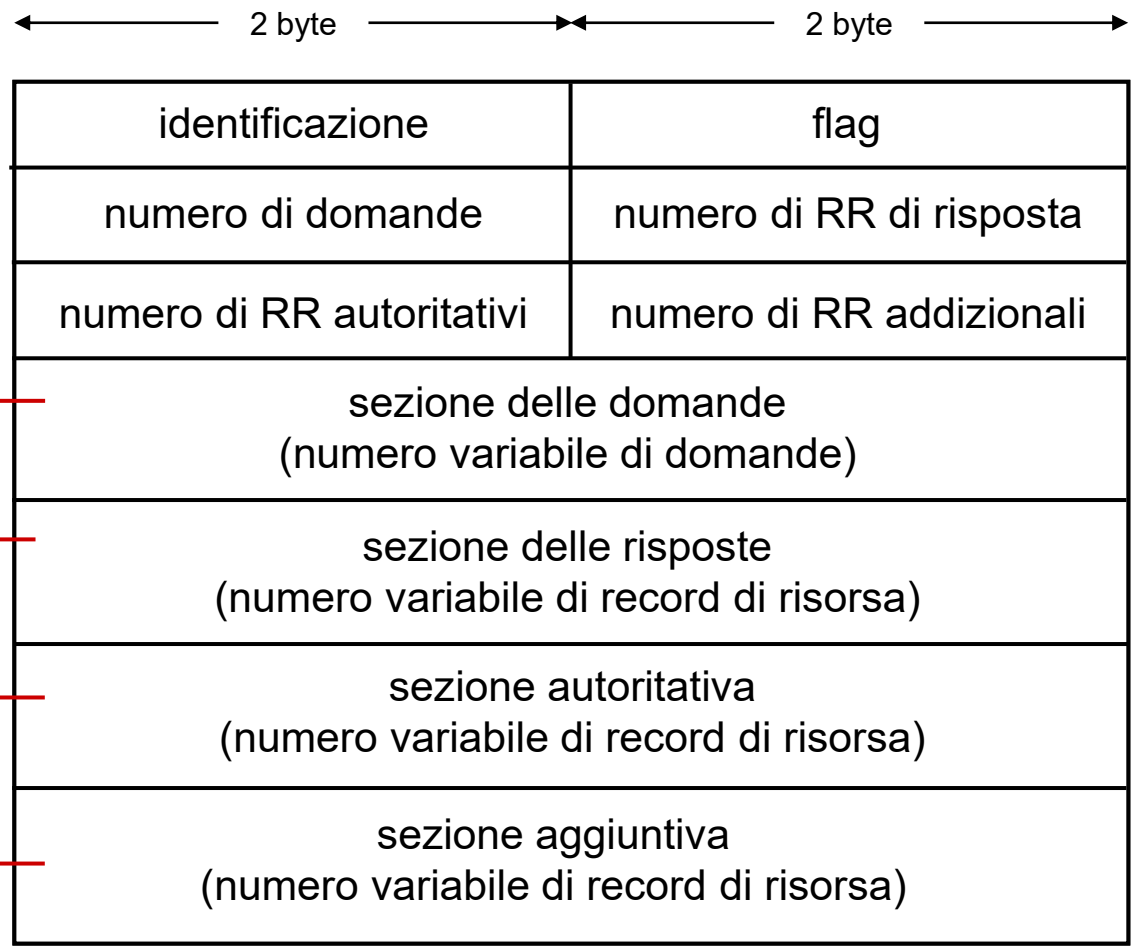
Intestazione del messaggio:

- identificazione: numero di 16 bit per la domanda; la risposta alla domanda usa lo stesso numero
- flag:
  - domanda o risposta
  - richiesta di ricorsione
  - ricorsione disponibile
  - DNS server autoritativo



Messaggi DNS

domande (query) e messaggi di *risposta* (reply), entrambi con lo stesso formato:



campi per il nome richiesto e il tipo di domanda

RR nella risposta alla domanda

record per i server autoritativi (*referral* verso nameserver di livello più basso; vedi RFC 9471: https://datatracker.ietf.org/doc/rfc9471/)

informazioni extra che possono essere usate (es. se la risposta ad una richiesta di tipo MX contiene un hostname, può essere fornita qui la sua traduzione in IP)

Inserire record nel database DNS

Esempio: abbiamo appena avviato la nuova società “Network Utopia”

- Registriamo il nome networkuptopia.com presso il *DNS registrar* (ad esempio, Network Solutions, oppure un altro dei concorrente accreditati dall'ICANN)
  - forniamo al registrar il nome e gli indirizzi IP degli authoritative name server (primario e secondario)
  - il registar inserisce due RR nel TLD server .com:

    `(networkutopia.com, dns1.networkutopia.com, NS)`

    `(dns1.networkutopia.com, 212.212.212.1, A)`
- Inseriamo localmente nell'authoritative server
  - un record A per www.networkuptopia.com
  - un record MX per networkutopia.com

Sicurezza del DNS

Attacchi DDoS (distributed denial-of-service)

- bombare di traffico di root server
  - finora senza successo
  - filtraggio del traffico
  - I server DNS locali mantengono in cachi gli indirizzi IP dei server TLD, consentendo di aggirare i root server
- bombardare i server TLD
  - potenzialmente più pericoloso

Attacco di spoofing

- intercettare le query DNS, restituendo risposte fasulle
  - DNS cache poisoning
  - RFC 4033: DNSSEC - servizi di autenticazione

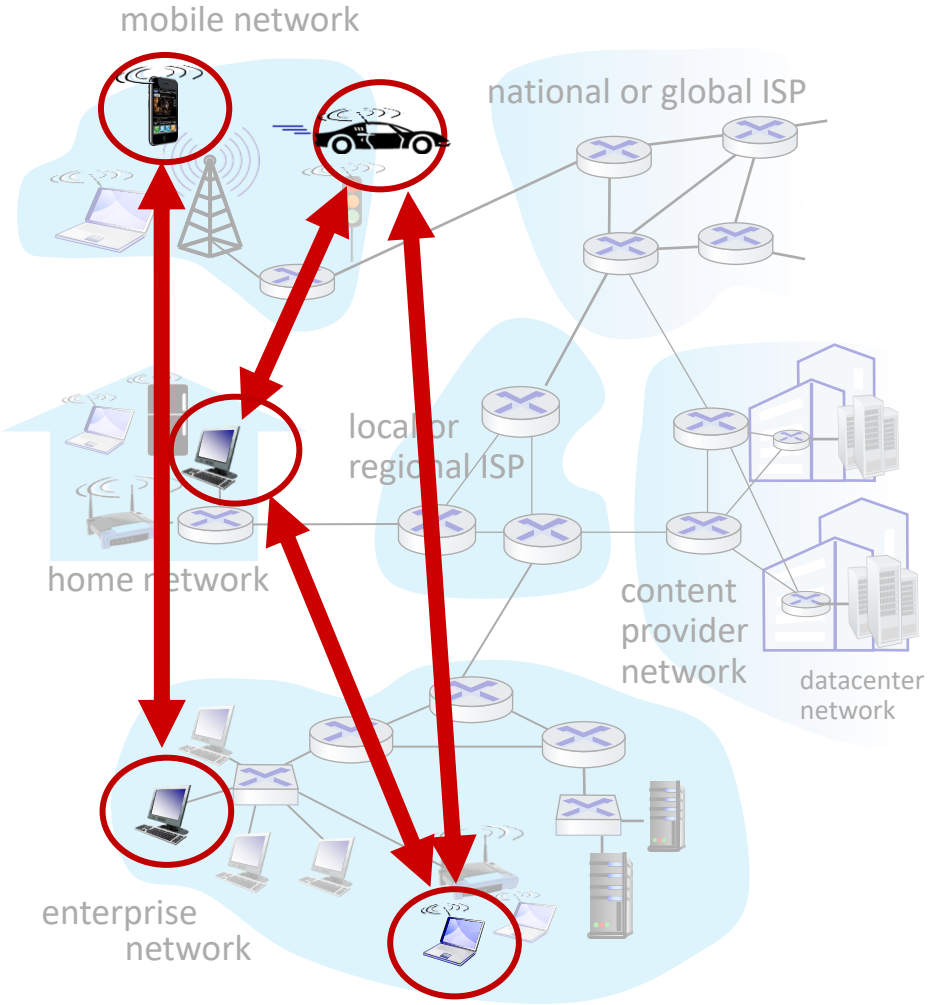
Application Layer: Overview

- Principi delle applicazioni di rete
- Web e HTTP
- E-mail, SMTP, IMAP
- DNS: il servizio di directory di Internet

- Applicazioni P2P
- Streaming video e reti di distribuzione di contenti
- Programmazione delle socket programming con UDP e TCP

Architettura Peer-to-peer (P2P)

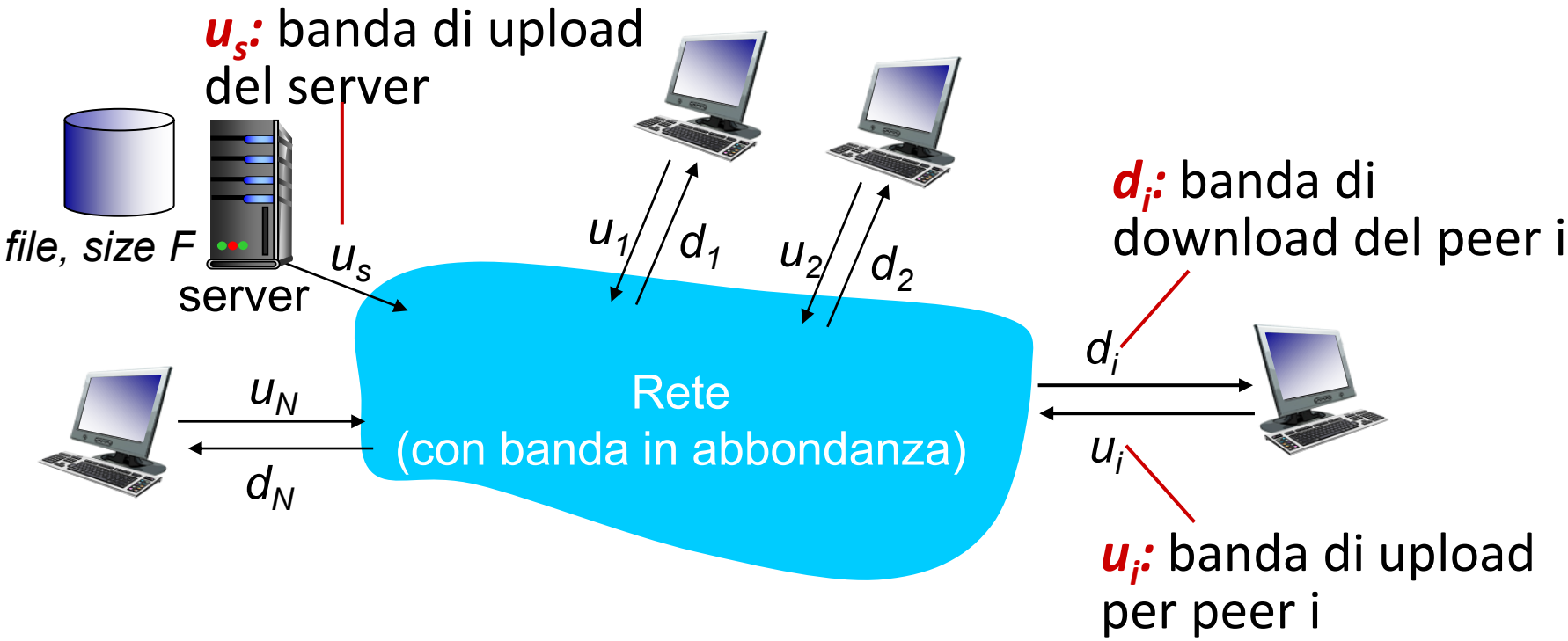
- *nessun* server sempre attivo
- sistemi periferici arbitrari comunicano direttamente
- i peer richiedono un servizio ad altri peer e forniscono un servizio in cambio ad altri peer
  - *scalabilità intrinseca - nuovi peer portano nuova capacità di servizio e nuove richieste di servizio*
- I peer sono connessi a intermittenza e cambiano indirizzo IP
  - gestione complessa
- Esempi: P2P file sharing (BitTorrent), streaming (KanKan), VoIP (Skype)



Distribuzione di file: client-server vs P2P

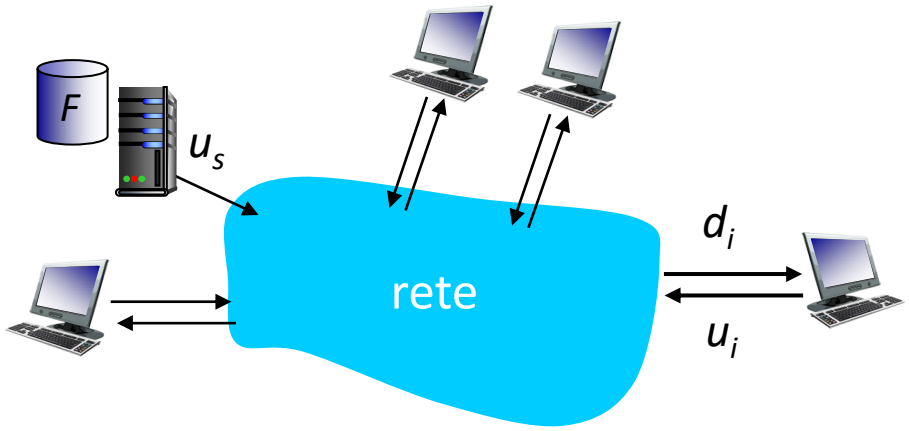
D: quanto tempo per distribuire un file (di dimensione *F*) da un server a *N peer*?

- la capacità di upload/download dei peer è una risorsa limitata



File distribution time: client-server

- *trasmissione via server:* deve inviare (caricare) in sequenza N copie di file:
  - tempo per inviare una copia: F/u_s
  - tempo per inviare N copie: NF/u_s
- *client:* ogni cliente deve scaricare una copia del file
  - d_{min} = banda di download più bassa
  - tempo di download per il client con banda minima è almeno: F/d_{min}



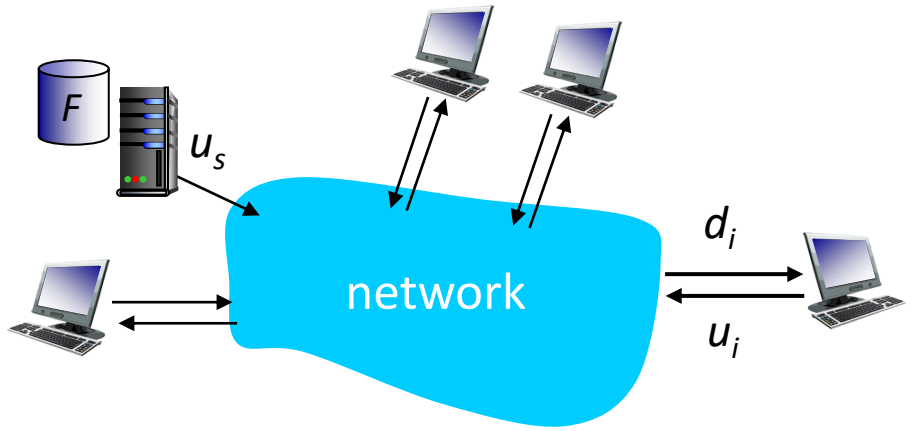
Tempo per distribuire F a N client usando l'approccio client-server

$$D_{c\text{-}s} \geq max\{NF/u_s, F/d_{min}\}$$

aumenta linearmente in N

Distribuzione di file: P2P

- *trasmissione via server:* deve trasmettere almeno una copia del file:
  - tempo per inviare un copia: F/u_s
- *client:* ogni cliente deve scaricare una copia del file
  - Tempo per il client più lento, almeno F/d_{min}
- I *client:* come aggregato devono scaricare NF bit
  - capacità totale di upload (che limita la massima velocità di download) è $u_s + \Sigma u_i$



Tempo per distribuire F a N client usando l'approccio P2P

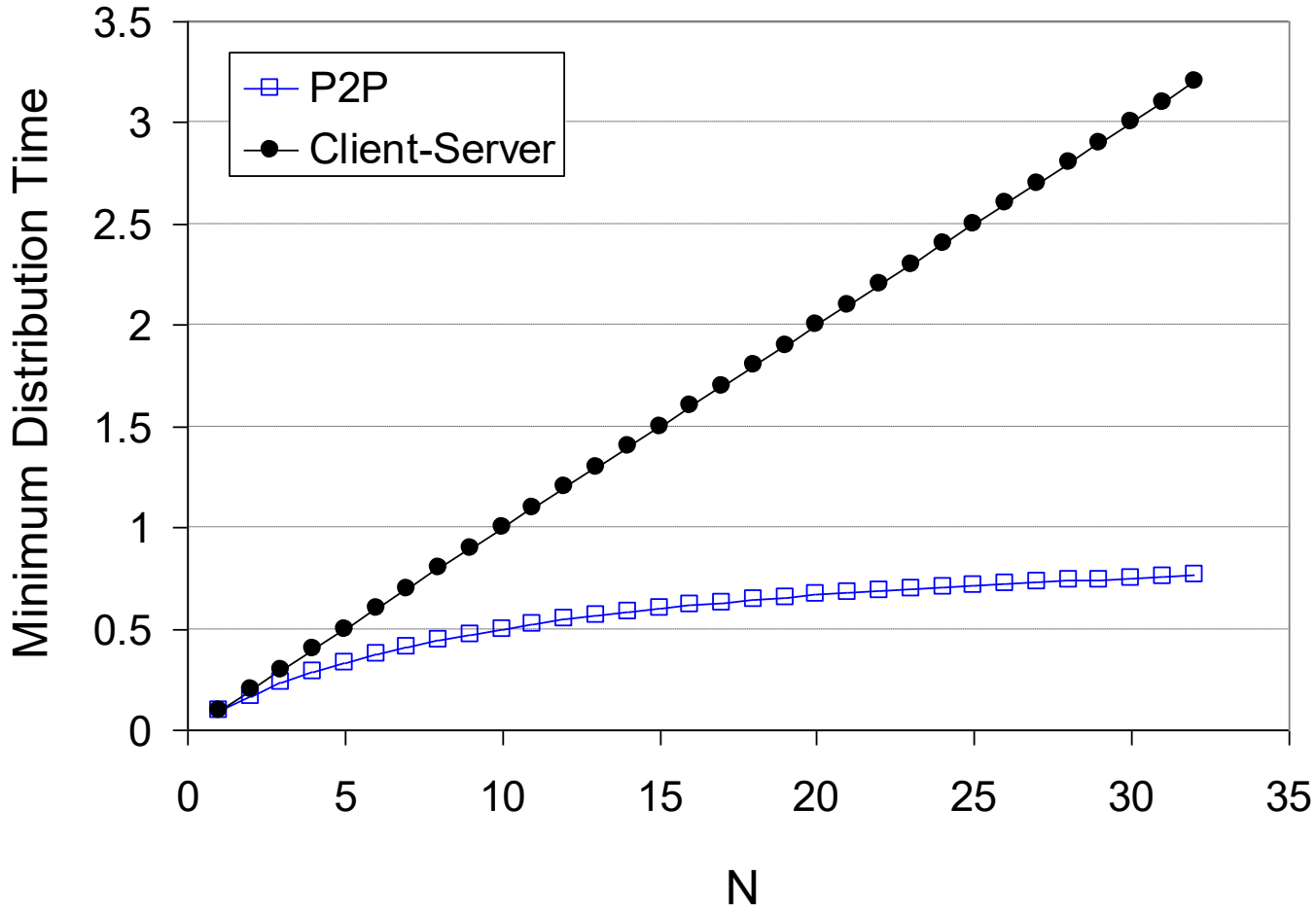
$$D_{P2P} \geq max\{F/u_s, F/d_{min}, NF/(u_s + \Sigma u_i)\}$$

aumenta linearmente in N …

… ma anche questo, dato che ogni peer porta con sé la capacità di servizio

Client-server vs. P2P: example

banda di upload del client = u, F/u = 1 ora, $u_s = 10u$, $d_{min} \geq u_s$

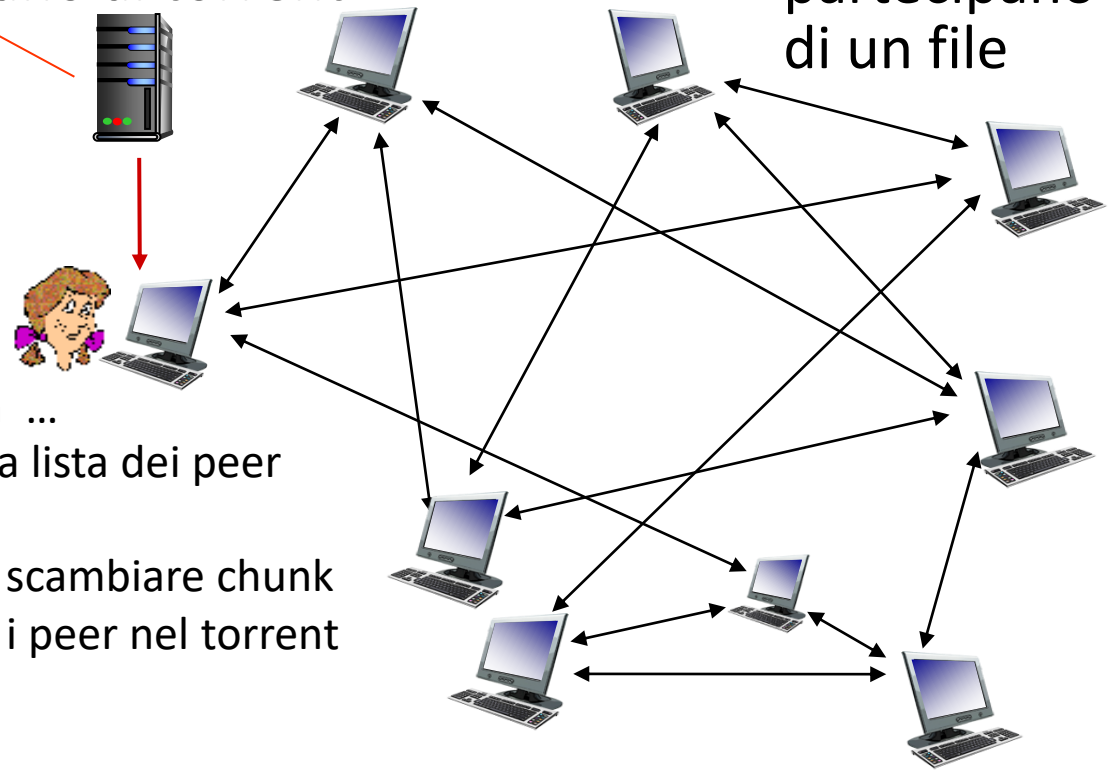


Distribuzione di file P2P: BitTorrent

- file diviso in chunk (parti), in genere di 256 kB
- i peer nel torrent inviano/ricevono chunk del file

tracker: tiene traccia dei peer che partecipano al torrent

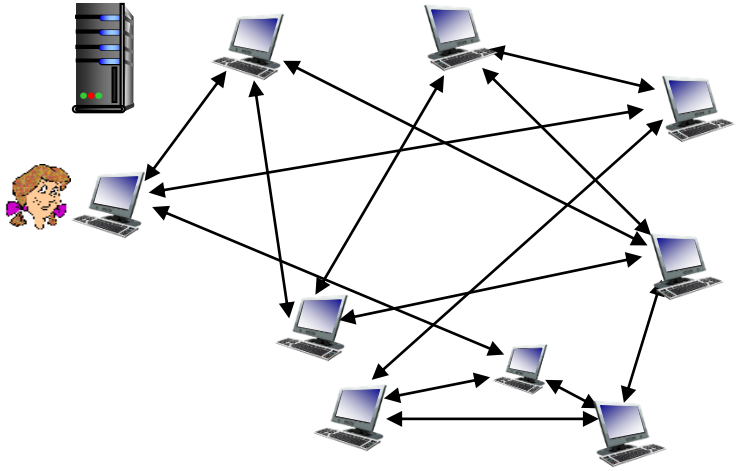
torrent: gruppo di peer che partecipano alla distribuzione di un file



Alice arriva …
… ottiene la lista dei peer dal tracker
… e inizia a scambiare chunk del file con i peer nel torrent

Distribuzione di file P2P: BitTorrent

- Un peer che entra a far parte del torrent:
  - non ha chunk del file, ma li accumulerà nel tempo da altri peer
  - si registra con un tracker, ottenendo la lista di un sottoinsieme dei peer nel torrent (es. 50), stabilisce una connessione con un sottoinsieme di questi, che sono detti peer "vicini" (“neighbors”)
  - informa periodicamente il tracker che è ancora nel torrent



- mentre scarica chunk, un peer invia i chunk già in suo possesso agli altri peer
- un peer può cambiare i peer con cui scambia i chunk
- i peer possono andare e venire
- una volta che un peer ha acquisito l'intero file, può (egoisticamente) lasciare il torrent oppure può (altruisticamente) rimanere nel torrent (come *seeder*)

BitTorrent: richiesta e invio di chunk di file

Richiesta di chunk:

- in ogni momento, peer diversi hanno sottoinsiemi diversi di chunk
- periodicamente, Alice chiede ai peer vicini l'elenco dei chunk in loro possesso
- Alice richiede ai peer i chunk mancanti, adottando la strategia del **rarest first** (**"**prima i più rari"): uniformando la distribuzione dei chunk, migliora la disponibilità globale e aumenta le possibilità di scambio (maggiore throughput)
- Un peer appena entrato può chiedere un blocco in modo casuale (perché vuole avere il prima possibile un blocco da condividere); mentre, quando sta per completare il file, può adottare la strategia end game e richiedere lo stesso blocco a più peer simultaneamente (cancellando le richieste pendenti appena appena riceve un blocco)

Invio di chunk: tit-for-tat ("pan per focaccia")

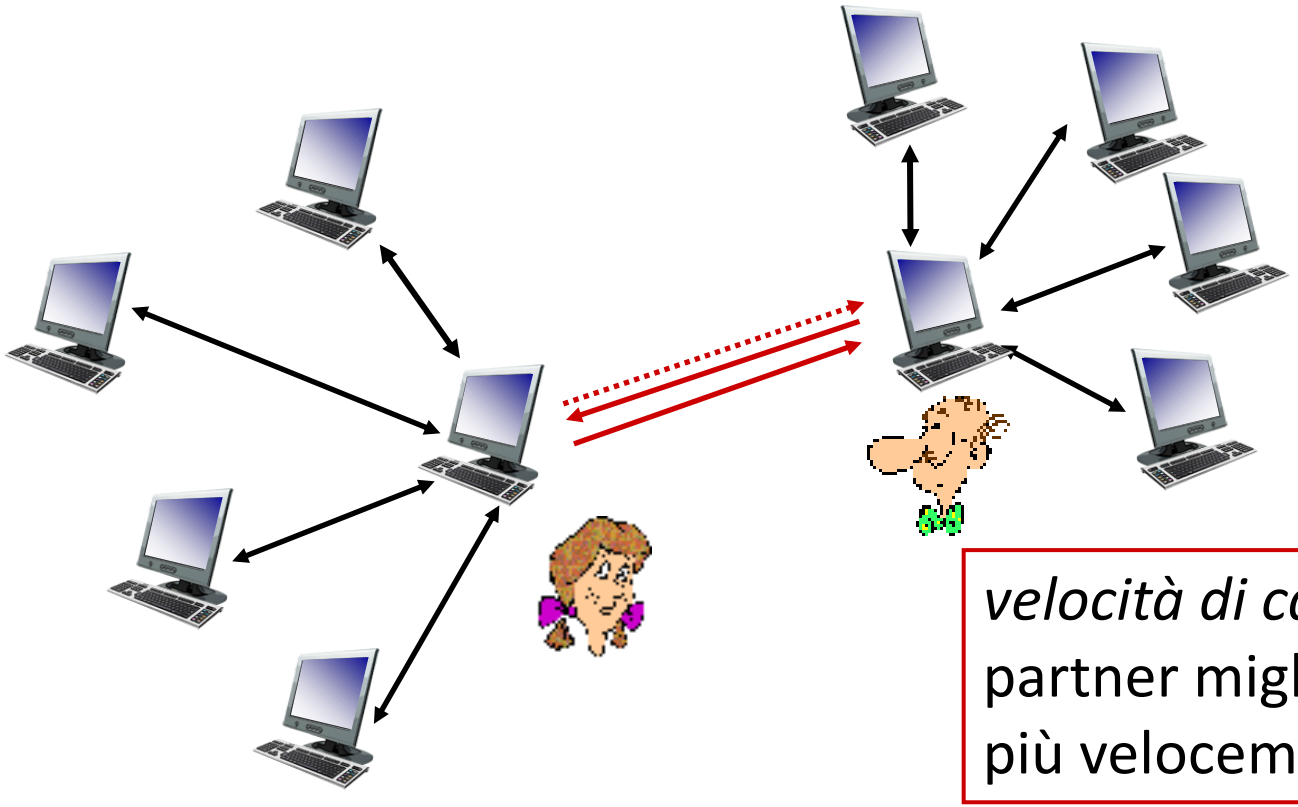
- Alice invia i chunk ai quattro peer vicini che attualmente le inviano i chunk *alla velocità più alta*
  - altri peer sono detti choked ("soffocati" o "limitati") (non ricevono chunk da Alice)
  - rivaluta i primi 4 posti ogni 10 secondi
- ogni 30 secondi: seleziona in modo casuale un vicino, inizia a inviare chunk
  - questo peer è detto “optimistically unchoked” (“non limitato/soffocato in maniera ottimistica”)
  - il nuovo peer scelto può entrare nella top 4

BitTorrent: tit-for-tat

(1) Alice scelte Bob come “optimistically unchoked”

(2) Alice diventa uno dei primi quattro fornitori di Bob; Bob ricambia

(3) Bob diventa uno dei primi quattro fornitori di Alice.



velocità di caricamento più elevata: trovare partner migliori per gli scambi, ottenere file più velocemente!

Livello di applicazione: panoramica

- Principi delle applicazioni di rete
- Web e HTTP
- E-mail, SMTP, IMAP
- DNS: il servizio di directory di Internet
- Applicazioni P2P
- Streaming video e reti di distribuzione di contenti
- Programmazione delle socket programming con UDP e TCP

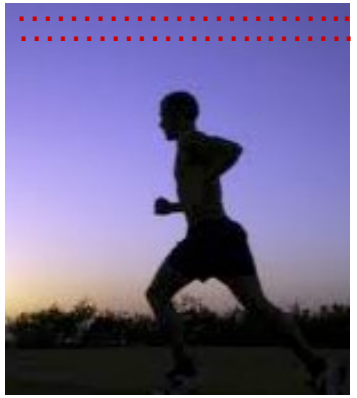
Streaming video e CDN: contesto

- traffico video in streaming :
  grande consumatore di larghezza
  di banda Internet
  - Netflix, YouTube, Amazon Prime: 80% del traffico ISP residenziale (2020)
- *sfida:* scala - come raggiungere ~1B di utenti?
- *sfida:* eterogeneità
  - utenti diversi hanno capacità diverse (ad esempio, cablati o mobili; ricchi di larghezza di banda o poveri di larghezza di banda)
- *soluzione:* infrastruttura distribuita a livello di applicazione

Contenuti multimediali: video

- video: sequenza di immagini visualizzate a tasso costante
  - Esempio: 24 immagini al secondo
- immagine digitale: un array di pixel
  - ogni pixel rappresentato da bit
- codifica: utilizzare la ridondanza *all'interno* e *tra* le immagini per ridurre il numero di bit utilizzati per la codifica dell'immagine
  - spaziale (all'interno di una data immagine)
  - temporale (da un'immagine all'altra)

esempio di codifica spaziale: invece di inviare N valori dello stesso colore (tutti viola), inviare solo due valori: valore del colore (viola) e numero di valori ripetuti (N)



frame *i*

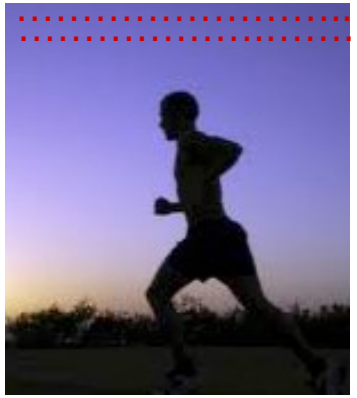
frame *i+1*

esempio di codifica temporale: invece di inviare il frame completo a i+1, invia solo le differenze dal frame i

Multimedia: video

- CBR: (constant bit rate): bit rate costante
- VBR: (variable bit rate): bit rate cambia con la quantità di codifica spaziale e temporale
- esempio:
  - MPEG 1 (CD-ROM) 1.5 Mbps
  - MPEG2 (DVD) 3-6 Mbps
  - MPEG4 (spesso usato in Internet, 64Kbps – 12 Mbps)

esempio di codifica spaziale: invece di inviare N valori dello stesso colore (tutti viola), inviare solo due valori: valore del colore (viola) e numero di valori ripetuti (N)



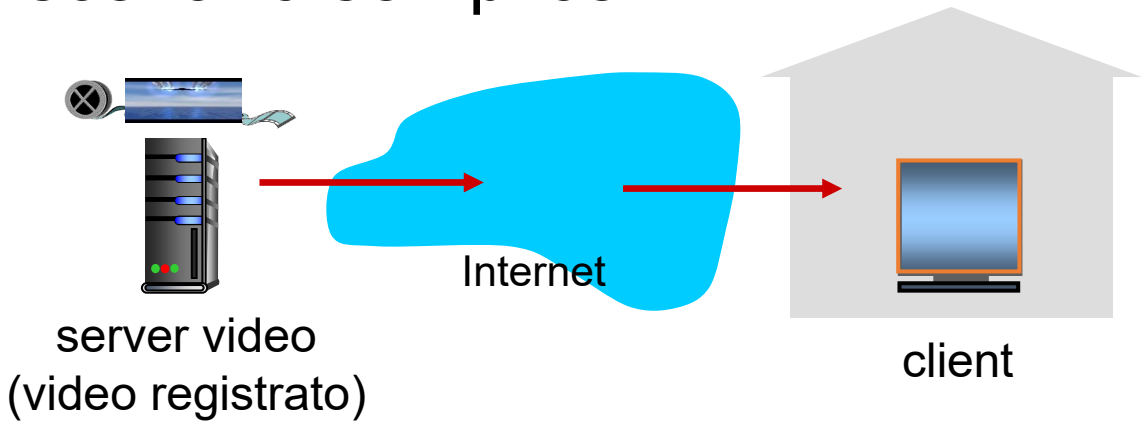
frame *i*

frame *i+1*

esempio di codifica temporale: invece di inviare il frame completo a i+1, invia solo le differenze dal frame i

Streaming video di contenuti registrati

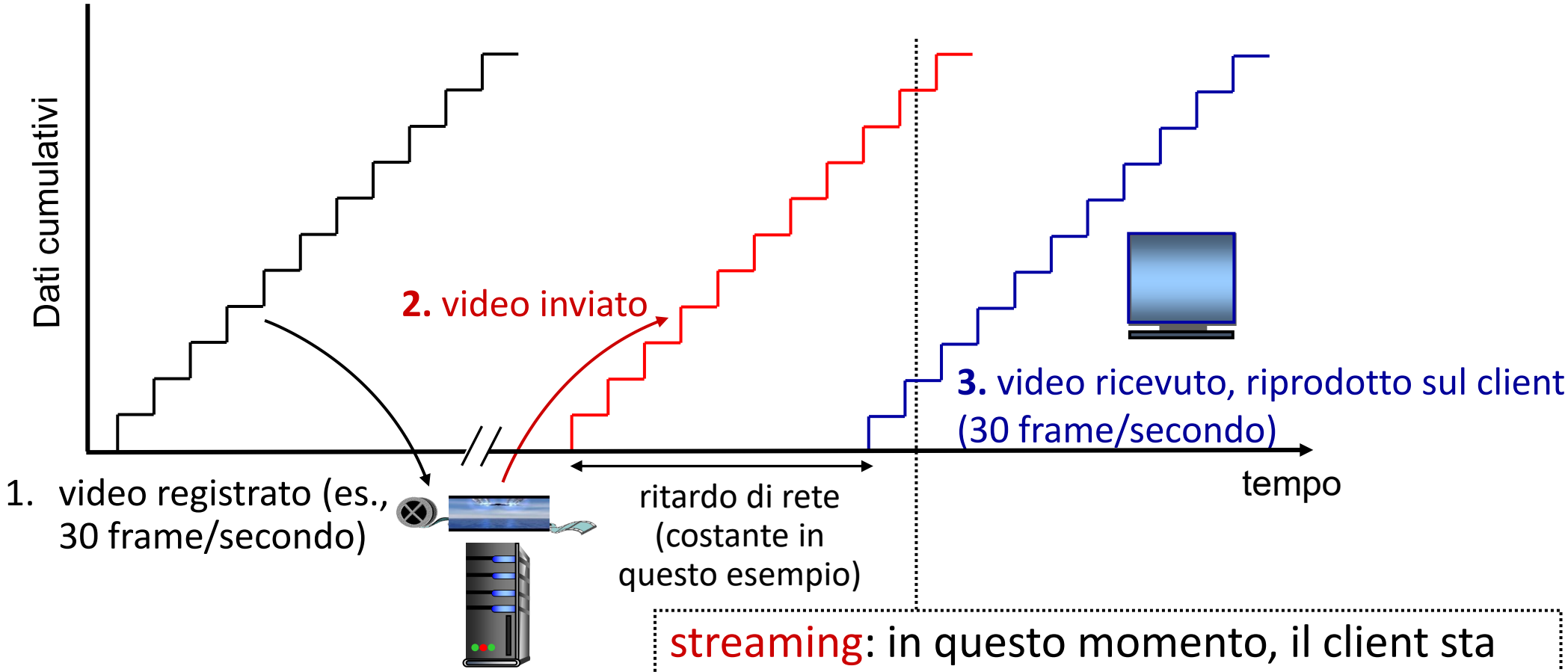
scenario semplice



Sfide principali:

- la larghezza di banda da server a client varia nel tempo, con il variare dei livelli di congestione della rete (rete residenziale, rete di accesso, nucleo della rete, server video)
- la perdita di pacchetti, i ritardi dovuti alla congestione ritardano la riproduzione o comportano una scarsa qualità video.

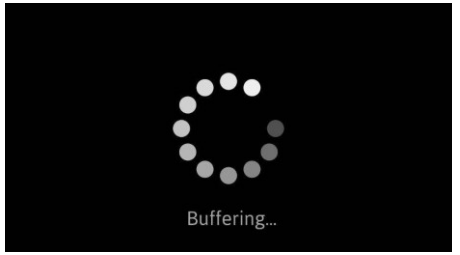
Streaming video di contenuti registrati



streaming: in questo momento, il client sta riproducendo la parte iniziale del video, mentre il server sta ancora inviando la parte successiva del video

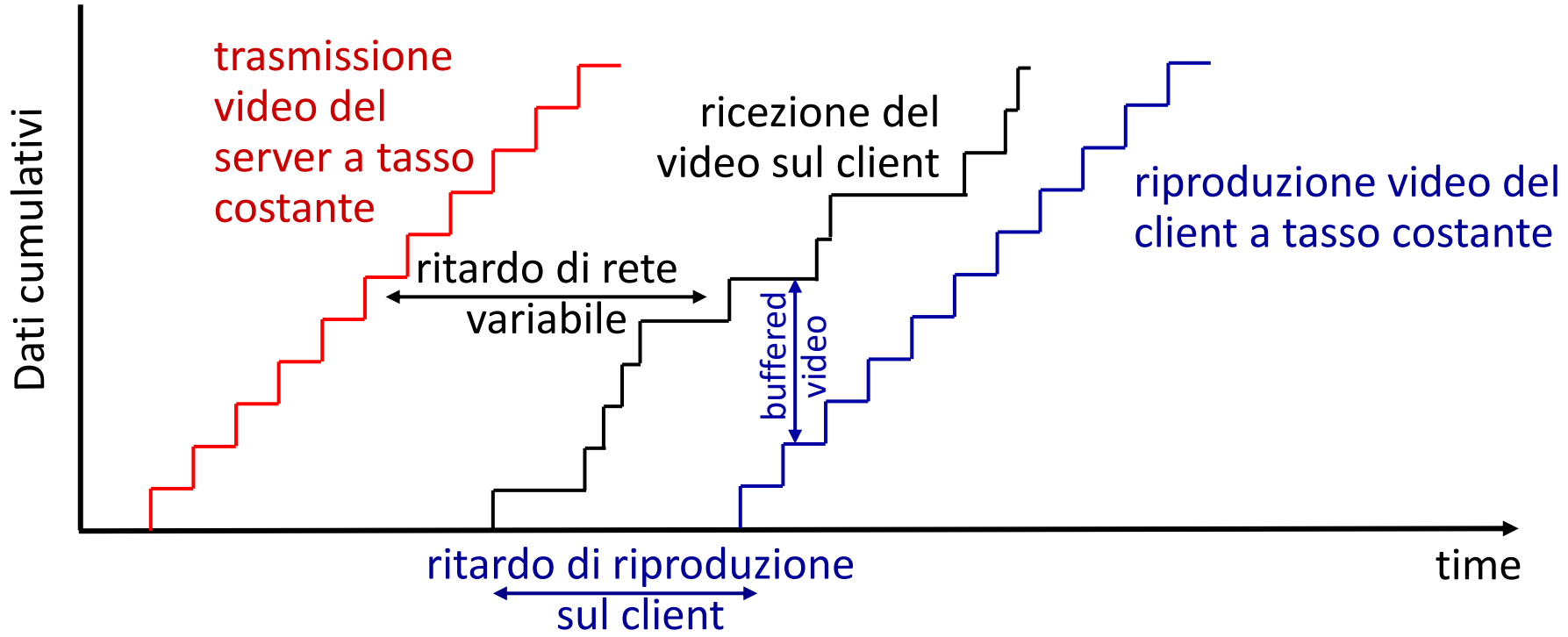
Streaming video di contenuti registrati

- vincolo di riproduzione continua: quando la riproduzione inizia, dovrebbe procedure secondo I tempi di registrazione originali
  - … ma i ritardi di rete sono variabili (jitter), quindi avrà bisogno di un buffer lato client per soddisfare i vincoli di riproduzione continua



- altre sfide:
  - interattività del client: pausa, avanzamento veloce, riavvolgimento, salti attraverso il video
  - i pacchetti video possono essere persi, ritrasmessi

Streaming video di contenuti registrati



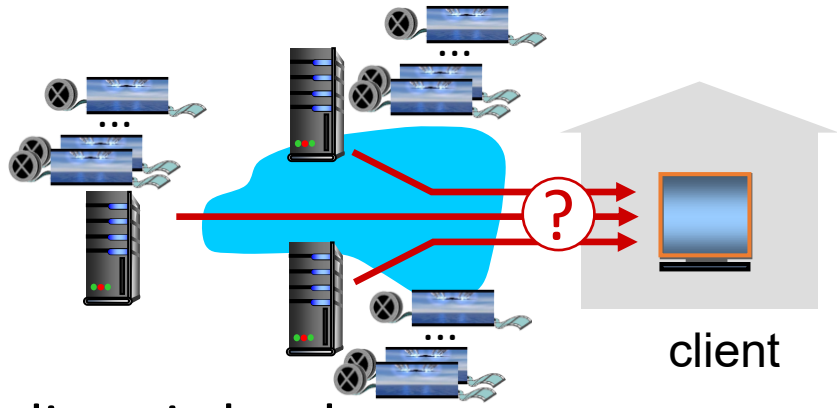
- *buffering lato client e ritardo di riproduzione:* compensare il ritardo aggiunto dalla rete, il jitter (variazione) del ritardo

Streaming multimediale: DASH

*D*ynamic, *A*daptive
*S*treaming over *H*TTP

server:

- divide il file video in più chunk
- ogni chunk è codificata in più versioni, con bit rate differenti
- versioni diverse sono memorizzate in file diversi
- i file sono replicati in vari nodi CDN
- *manifest file (file manifesto):* fornisce gli URL per i diversi chunk

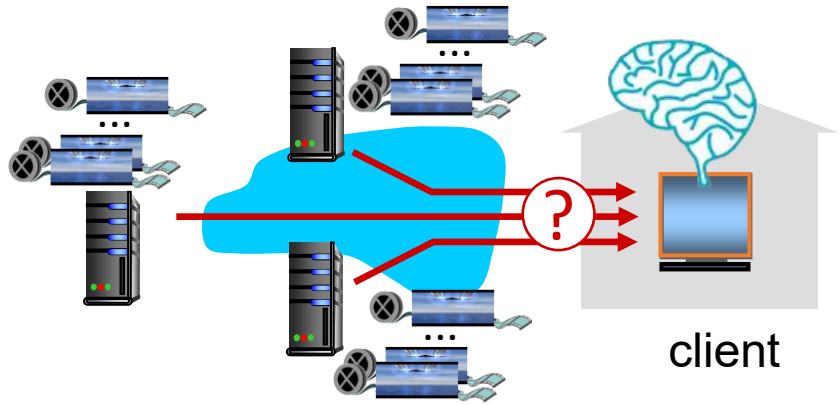


client:

- stima periodicamente la banda da server a client
- consultando il manifesto, richiede un chunk alla volta
  - sceglie versione con il bit rate più alto sostenibile data la larghezza di banda corrente
  - può scegliere versioni con bit rate differenti in momenti diversi (a seconda della larghezza di banda disponibile in quel momento), e da server diversi

Streaming multimediale: DASH

- *"intelligenza"* sul client: il client determina
  - *quando* richiedere un chunk (in modo che non si verifichi la starvation del buffer o l'overflow)
  - *che encoding rate richiedere (qualità più alta quando c'è più larghezza di banda*)
  - *dove* richiedere il chunk (può richiedere dal server che è "vicino" al client o ha banda larga)



Streaming video = codifica + DASH + buffering di riproduzione

Reti per la distribuzione di contenuti - Content distribution networks (CDNs)

sfida: come trasmettere contenuti in streaming (selezionati tra milioni di video) a centinaia di migliaia di utenti simultanei?

- *opzione 1:* unico, enorme data center
  - singolo punto di rottura (single point of failure)
  - punto di congestione della rete
  - percorso lungo (e possibilmente congestionato) verso i clienti lontani

…. molto semplicemente: questa soluzione *non è scalabile*