



UNIVERSITY  
OF TRENTO

Dipartimento di Ingegneria e Scienza dell'Informazione

Corso di Laurea in  
Ingegneria dell'Informazione e Organizzazione d'Impresa

## APPUNTI DI RETI

Dal materiale del prof. F. Granelli

Autore  
Davide Parpinello

Anno accademico 2019-2020

# Indice

<b>1 Roadmap</b>	<b>3</b>
1.1 Internet . . . . .	3
1.2 Ai confini della rete . . . . .	3
1.3 Il nucleo della rete . . . . .	3
1.3.1 Esempio di commutazione di circuito . . . . .	4
1.3.2 Esempio di commutazione di circuito . . . . .	4
1.3.3 Confronto fra commutazione di pacchetto e di circuito . . . . .	4
1.3.4 Struttura gerarchica . . . . .	4
1.4 Ritardi, perdite e throughput nelle reti a comunicazione di pacchetto . . . . .	5
1.4.1 Ritardo di un nodo . . . . .	5
1.5 Livelli di protocollo e i loro modelli di servizio . . . . .	6
1.6 Reti sotto attacco: la sicurezza . . . . .	6
<b>2 Il livello Applicazione</b>	<b>7</b>
2.1 Principi delle applicazioni di rete . . . . .	7
2.2 Web e HTTP . . . . .	8
2.2.1 Connessioni non persistenti . . . . .	8
2.2.2 Connessioni persistenti . . . . .	9
2.2.3 Messaggi HTTP . . . . .	9
2.2.4 Cookies . . . . .	10
2.2.5 Cache web (server proxy) . . . . .	10
2.2.6 HTTP/2.0 . . . . .	10
2.3 FTP . . . . .	10
2.3.1 Comandi comuni . . . . .	11
2.3.2 Codici di ritorno comuni . . . . .	11
2.4 Posta elettronica . . . . .	11
2.5 DNS . . . . .	12
2.5.1 Protocollo DNS . . . . .	13
2.6 Condivisione di file P2P . . . . .	14
2.6.1 Confronto tra architettura server client e P2P . . . . .	14
2.7 Cloud Computing . . . . .	15
2.7.1 CDN . . . . .	15
<b>3 Il livello di Trasporto</b>	<b>16</b>
3.1 Servizi a livello di trasporto . . . . .	16
3.1.1 Demultiplexing senza connessione . . . . .	16
3.1.2 Demultiplexing orientato alla connessione . . . . .	17
3.2 Trasporto senza connessione: UDP . . . . .	17
3.3 Principi del trasferimento dati affidabile . . . . .	18
3.3.1 Rdt 1.0: trasferimento affidabile su canale affidabile . . . . .	18
3.3.2 Rdt 2.0: canale con errori nei bit . . . . .	18

3.3.3	Rdt 2.1 . . . . .	19
3.3.4	Rdt 2.2: un protocollo senza NAK . . . . .	19
3.3.5	Rdt 3.0: canali con errori e perdite . . . . .	19
3.3.6	Pipelining . . . . .	20
3.4	Trasporto orientato alla connessione: TCP . . . . .	20
3.4.1	TCP: controllo di flusso . . . . .	21
3.4.2	Gestione della connessione . . . . .	22
3.5	Principi del controllo di congestione . . . . .	22
3.6	Controllo di congestione in TCP (AIMD) . . . . .	22
3.6.1	Partenza lenta . . . . .	23
3.6.2	Riassunto: controllo di congestione . . . . .	23
3.6.3	Throughput TCP . . . . .	23
3.6.4	Equità di TCP . . . . .	23

# Capitolo 1

## Roadmap

### 1.1 Internet

Internet è costituito da milioni di dispositivi, chiamati **sistemi terminali**, collegati tra loro da collegamenti in rame, fibre ottiche, oppure via radio come onde elettromagnetiche o satelliti.

La frequenza di trasmissione in internet è data dall'ampiezza di banda disponibile.

Sulla rete internet inoltre sono presenti particolari host, denominati **router**, che si occupano di instradare i pacchetti verso la loro destinazione finale.

Nello scambio di messaggi tra host vengono implementati dei **protocolli**, che ne definiscono formato e ordine d'invio, così come le azioni intraprese in fase di trasmissione e/o ricezione di un messaggio o un altro evento.

Internet è un'infrastruttura di comunicazione per applicazione distribuita, viene anche chiamato "rete delle reti" ed è organizzato in modo gerarchico.

### 1.2 Ai confini della rete

Sul bordo della rete internet troviamo applicazioni e sistemi terminali, raggruppati tra loro e connessi tra di loro mediante collegamenti cablati e wireless.

I sistemi terminali (o host) fanno girare diversi programmi applicativi e possono essere organizzati con architettura client/server oppure P2P.

L'accesso a internet può avvenire mediante diversi modi:

- **Accesso residenziale:** viene utilizzato un modem dial-up o DSL.
- **Accesso aziendale:** una LAN collega i sistemi terminali di aziende e università all'**edge router**, i sistemi terminali sono collegati tra loro mediante uno switch ethernet.
- **Accesso wireless:** i terminali vengono collegati mediante **access point**.
- **Reti domestiche:** sono costituite da un modem DSL o via cavo, un router/firewall/NAT, switch ethernet e accesso wireless. Spesso queste funzioni vengono raggruppate in un unico dispositivo (modem/router).

### 1.3 Il nucleo della rete

Al centro della rete si trovano invece router interconnessi tra loro, che creano quindi una rete delle reti.

I dati nel nucleo della rete vengono trasferiti con due modalità differenti:

- **Commutazione di circuito:** è presente un circuito dedicato per l'intera durata della sessione
- **Commutazione di pacchetto:** i messaggi di una sessione utilizzano le risorse su richiesta, di conseguenza potrebbero dover attendere per accedere a un collegamento.

### 1.3.1 Esempio di commutazione di circuito

Consideriamo un file L di 640.000 bit, un bitrate totale C da 1.536 Mbps, TDM con 24 slot/s, 500ms per stabilire la connessione.

Trovo inizialmente la capacità di un singolo slot:

$$C_{1slot} = \frac{C}{24} = 0.064Mbps = 64Kbps$$

Successivamente calcolo il tempo necessario alla trasmissione:

$$T_{tx} = \frac{L}{C_{1slot}} = 10s$$

Infine, calcolo il tempo totale:

$$T_{tot} = 500ms + 10s = 10,5s$$

### 1.3.2 Esempio di commutazione di circuito

I secondi necessari per trasmettere un pacchetto in uscita su un collegamento da R bps sono dati da  $L/R$ , mentre il ritardo  $3L/R$

### 1.3.3 Confronto fra commutazione di pacchetto e di circuito

La commutazione di pacchetto consente un utilizzo della rete da parte di maggiori utenti, ed è ottima per i dati a raffica.

Dal lato negativo, presenta un'eccessiva congestione, causando ritardi e perdite di pacchetti. Sono quindi necessari protocolli per il trasferimento affidabile e per il controllo della congestione.

### 1.3.4 Struttura gerarchica

La rete internet ha una struttura fondamentalmente gerarchica:

- Al centro sono presenti **ISP di livello 1**, che offrono una copertura nazionale e/o internazionale
  - Comunicano tra loro come fossero pari
- **ISP di livello 2:** ISP più piccoli, copertura nazionale/distrettuale.
  - Si può connettere solo ad alcuni ISP di livello 1 e ad altri di livello 2
  - Paga l'ISP di livello 1 che gli fornisce la connettività per il resto della rete
- **ISP di livello 3 e ISP locali (di accesso):**
  - sono le reti "ultimo salto", le più vicine agli host
  - Sono clienti degli ISP di livello superiore che li collegano all'intera internet.

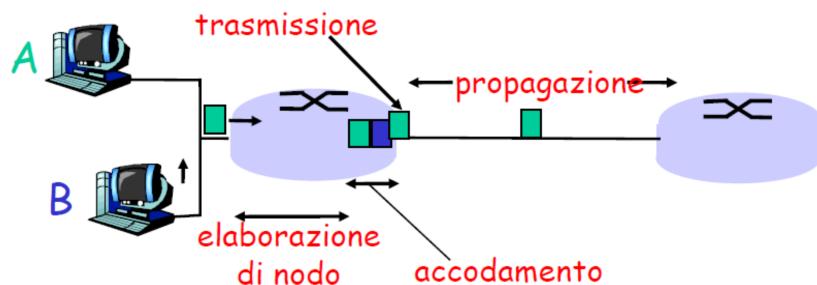
Un pacchetto attraversa un sacco di reti, dal livello più basso fino al principale e poi nuovamente a scendere.

## 1.4 Ritardi, perdite e throughput nelle reti a comunicazione di pacchetto

Nella rete si verificano dei ritardi quando il tasso di arrivo dei pacchetti eccede la capacità di evaderli, con la conseguenza che vengono accodati nei buffer del router in attesa del proprio turno. Se non ci sono buffer liberi i pacchetti vengono scartati e ritrasmessi dal nodo precedente o, in alcuni casi, non venire proprio ritrasmessi.

Quattro cause di ritardo dei pacchetti sono le seguenti:

1. Ritardo di elaborazione sul nodo
  - Controllo errori sui bit
  - Determinazione del canale di uscita
2. Ritardo di accodamento
  - Attesa di trasmissione
  - Livello di congestione del router
3. Ritardo di trasmissione (L/R)
  - $R$  = frequenza di trasmissione del collegamento
  - $L$  = lunghezza del pacchetto
4. Ritardo di propagazione ( $d/s$ )
  - $d$  = lunghezza dl collegamento fisico
  - $s$  = velocità di propagazione del collegamento ( $\sim 2 \cdot 10^8$  m/s)



### 1.4.1 Ritardo di un nodo

Il ritardo di un nodo è dato dalla seguente formula:

$$d_{nodal} = d_{proc} + d_{queue} + d_{trans} + d_{prop} \quad (1.1)$$

dove:

- $d_{proc}$  = ritardo di elaborazione (processing delay)
  - in genere di pochi microsecondi, o anche meno
- $d_{queue}$  = ritardo di accodamento
  - dipende dalla congestione

- $d_{trans}$  = ritardo di trasmissione (transmission delay)
  - significativo sui collegamenti a bassa velocità
- $d_{prop}$  = ritardo di propagazione (propagation delay)
  - da pochi microsecondi a centinaia di millisecondi

### Ritardo di accodamento

- $R$  = frequenza di trasmissione (bps)
- $L$  = lunghezza del pacchetto (bit)
- $a$  = tasso medio di arrivo dei pacchetti

Se calcoliamo  $La/R$  otteniamo l'intensità di traffico:

- Vicino a 0: poco ritardo
- Minore o uguale a 1: traffico consistente
- Maggiore di 1: più lavoro in arrivo di quanto possa essere effettivamente svolto, ritardo medio infinito.

Il **throughput** viene calcolato come la frequenza (bit/unità di tempo) alla quale i bit sono trasferiti tra mittente e ricevente. Può essere istantaneo o medio (in un periodo più lungo).

In internet si considera anche il **collo di bottiglia**, ovvero un collegamento su un percorso punto-punto che vincola un throughput end to end.

## 1.5 Livelli di protocollo e i loro modelli di servizio

Si considerano principalmente 5 livelli di protocollo:

1. **Applicazione:** di supporto alle applicazioni di rete
2. **Trasporto:** trasferimento dei messaggi del livello applicazione tra modulo client e server, connessione tra processi applicativi
3. **Rete:** instradamento dei datagram dall'origine al destinatario
4. **Link (*collegamento*):** instradamento dei datagram attraverso una serie di commutatori di pacchetto
5. **Fisico:** trasferimento dei singoli bit

## 1.6 Reti sotto attacco: la sicurezza

I malintenzionati installano malware negli host attraverso internet. Il malware può raggiungere gli host attraverso virus, worm o cavalli di Troia. Un malware di spionaggio può registrare quanto viene digitato, i siti visitati e informazioni di upload. Gli host infettati possono diventare botnet e essere usati per lo spamming e attacchi DDoS. Un malware è spesso auto-replicante e da un host attaccato può passare ad altri host.

**Analisi di pacchetti** Chiamato anche packet sniffing, quando un'interfaccia di rete legge/registra tutti i pacchetti che la attraversano.

**IP spoofing** Invio di pacchetti con un indirizzo sorgente falso.

**Record-and-playback** Vengono "sniffati" dati sensibili per utilizzarli in un secondo momento.

# Capitolo 2

## Il livello Applicazione

### 2.1 Principi delle applicazioni di rete

Le applicazioni di rete sono costruite con diverse architetture

- Client-Server
  - L'host (o client) interagisce direttamente con il server
  - Più client ci sono più il servizio diminuisce
- Peer-to-Peer (P2P)
  - Non c'è un server sempre attivo
  - Coppie arbitrarie di host comunicano direttamente fra di loro
  - Facilmente scalabile ma difficile da gestire
- Architetture ibride
  - Skype, messaggistica istantanea
  - Connessione client-client, utilizzo del server per la ricerca dell'indirizzo della parte remota
- Cloud computing
  - Un insieme di tecnologie che permettono sia di archiviare dati che elaborarli tramite l'utilizzo di risorse distribuite e virtualizzate in rete
  - Creazione di copie di sicurezza preventive in modo automatico trasferendo tutta l'operatività online
  - Dati memorizzati in server farm
  - Sempre client-server ma basata sulla virtualizzazione

Un processo è un programma in esecuzione su un host. All'interno dell'host due processi comunicano utilizzando schemi di interprocesso, mentre su host differenti comunicano attraverso lo scambio di messaggi.

- **Processo client:** processo che dà inizio alla comunicazione
- **Processo server:** processo che attende di essere contattato

Un processo invia/riceve messaggi mediante la sua socket. La socket è analoga ad una porta mediante la quale un processo che vuole inviare un messaggio lo fa uscire. Questo presuppone l'esistenza di un'infrastruttura esterna che trasporterà il messaggio attraverso la rete fino alla porta del processo di destinazione.

Per il trasporto internet vengono utilizzati i servizi dei protocolli TCP e UDP.

### Servizio TCP

- **Orientato alla connessione:** è richiesto un setup fra i processi client e server
- **Trasporto affidabile** fra i processi d'invio e ricezione
- **Controllo di flusso:** il mittente non vuole sovraccaricare il destinatario
- **Controllo della congestione:** "stroppa" il processo d'invio quando la rete è sovraccaricata
- **Non offre** temporizzazione, garanzie sull'ampiezza di banda minima, sicurezza
- **Più affidabile ma si paga con ritardi di trasferimento**

### Servizio UDP

- Trasferimento dati inaffidabile fra processi d'invio e ricezione
- **Non offre** setup della connessione, affidabilità, controllo di flusso, controllo della congestione, temporizzazione né ampiezza di banda minima e sicurezza
- **Mandare i dati il più velocemente possibile ma senza completa affidabilità**, usato per gaming online e VoIP

## 2.2 Web e HTTP

L'HTTP, o HyperText Transfer Protocol, è un protocollo a livello applicazione per il web, che considera un client che richiede, riceve e visualizza gli oggetti del Web e un server, che invia gli oggetti in risposta ad una richiesta.

Viene utilizzato TCP nel seguente modo:

- Il client inizializza la connessione TCP con il server (crea una socket) tipicamente sulla porta 80
- Il server accetta la connessione TCP dal client
- Avviene lo scambio di messaggi HTTP tra browser e server web
- Chiusura della connessione TCP

HTTP è un protocollo *stateless* (senza stato), cioè il server non mantiene informazioni sulle richieste fatte dal client.

Le connessioni HTTP possono essere non persistenti o persistenti.

**RTT** Round trip time, tempo impiegato da un piccolo pacchetto per andare dal client al server e ritornare al client.

### 2.2.1 Connessioni non persistenti

Viene trasferito un solo oggetto su una singola connessione TCP. Il tempo di risposta è dato da:

- Un RTT per inizializzare la connessione TCP
- Un RTT per inviare la richiesta HTTP e i primi byte
- Il tempo necessario alla trasmissione del file

Quindi totale = 2 RTT + tempo di trasmissione. Le connessioni non persistenti presentano alcuni svantaggi:

- Richiedono 2RTT per ogni oggetto
- Overhead dell'OS per ogni connessione TCP
- I browser aprono spesso connessioni TCP parallele per caricare gli oggetti referenziati

### 2.2.2 Connessioni persistenti

Il server lascia la connessione TCP aperta dopo l'invio di una risposta, i successivi messaggi tra gli stessi client/server vengono trasmessi sulla connessione aperta, il client invia le richieste non appena incontra un oggetto referenziato, un solo RTT per tutti gli oggetti referenziati.

### 2.2.3 Messaggi HTTP

I messaggi HTTP possono essere di richiesta o di risposta.

#### Richiesta HTTP

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
User-agent: Mozilla/4.0
Connection: close Accept-language:fr
```

#### Tipi di metodi

La versione HTTP/1.0 fornisce i seguenti metodi per effettuare le richieste:

- **GET:** richiesta solamente mediante URL
- **POST:** vengono inseriti nel corpo della richiesta anche dati da inviare al server
- **HEAD:** chiede al server di escludere l'oggetto richiesto dalla risposta

Nella versione HTTP/1.1 vengono aggiunti i seguenti metodi:

- **PUT:** include il file nel corpo dell'entità e lo invia al percorso specificato nel campo URL
- **DELETE:** cancella il file specificato nel campo URL

#### Risposta HTTP

```
HTTP/1.1 200 OK
Connection close
Date: Thu, 06 Aug 1998 12:00:15 GMT
Server: Apache/1.3.0 (Unix)
Last-Modified: Mon, 22 Jun 1998 ...
Content-Length: 6821
Content-Type: text/html
```

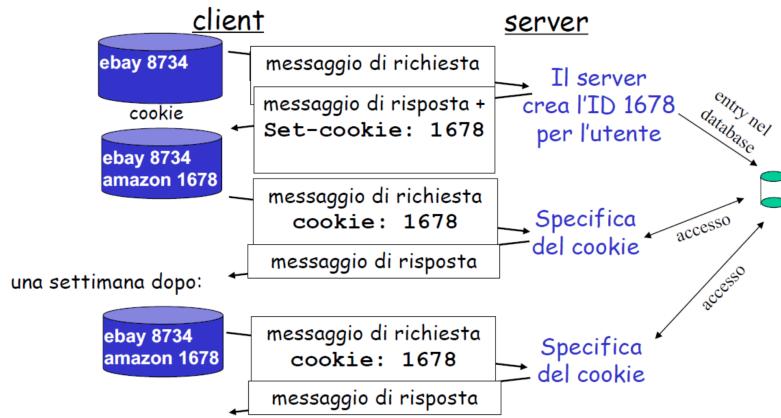
#### Codici di stato della risposta HTTP

Nella prima riga del messaggio di risposta sono inclusi codice di stato e relativa espressione in base all'esito della richiesta. Alcuni codici sono i seguenti:

- **200 OK:** La richiesta ha avuto successo, oggetto inviato nella risposta
- **301 Moved Permanently:** Oggetto trasferito nella nuova posizione indicata da Location

- **400 Bad Request:** il messaggio di richiesta non è stato compreso dal server
- **404 Not Found:** il documento richiesto non si trova su questo server
- **505 HTTP Version Not Supported:** il server non ha la versione di protocollo HTTP indicata

#### 2.2.4 Cookies



I cookies possono essere utilizzati per: autorizzazioni, dati carte per acquisti, raccomandazioni all'utente e possono mantenere lo stato del mittente e del ricevente per più transazioni; i messaggi HTTP trasportano lo stato. I cookies permettono ai siti di imparare molte cose sugli utenti.

#### 2.2.5 Cache web (server proxy)

L'obiettivo di una web cache è quello di soddisfare la richiesta del client senza coinvolgere il server d'origine. La cache opera come client e come server, e consente di ridurre i tempi di risposta alle richieste dei client, riducendo il traffico sul collegamento a internet.

**GET condizionale** Ha l'obiettivo di non inviare un oggetto se la cache ne ha una copia aggiornata. Per questo la cache specifica la data della copia dell'oggetto nella richiesta HTTP: `If-modified-since: <date>` A lato server la risposta non contiene l'oggetto se la copia nella cache è aggiornata: `HTTP/1.0 304 not modified`

#### 2.2.6 HTTP/2.0

È un'evoluzione di HTTP, che mantiene quindi i metodi HTTP, i codici di stato e la semantica migliorando però le prestazioni.

### 2.3 FTP

FTP, o File Transfer Protocol, viene utilizzato per trasferire file con un host remoto. Il server FTP opera solitamente sulla porta 21. Vengono aperte due connessioni, prima una di controllo e successivamente, se andata a buon fine la prima, una seconda per il trasferimento del file. La connessione di controllo è quindi "fuori banda" (out of band). Il server FTP mantiene lo stato

### 2.3.1 Comandi comuni

- USER `username`
- PASS `password`
- LIST elenca i file della directory corrente
- RETR `filename` recupera un file dalla directory corrente
- STOR `filename` memorizza un file nell'host remoto

### 2.3.2 Codici di ritorno comuni

- 331 Username OK, password required
- 125 data connection already open; transfer starting
- 425 Can't open data connection
- 452 Error writing file

## 2.4 Posta elettronica

La posta elettronica è costituita da tre componenti principali: agente utente, server di posta, SMTP (Simple Mail Transfer Protocol).

**Agente utente** Detto anche mail reader, si occupa della composizione, editing e lettura dei messaggi di posta elettronica (esempi: Outlook, Thunderbird). I messaggi in uscita sono memorizzati sul server.

**Server di posta** In esso è contenuta la casella di posta (mailbox) che contiene i messaggi in arrivo per l'utente e la coda dei messaggi da trasmettere.

**Protocollo SMTP** Utilizzato tra i server per inviare messaggi: il client è il server di posta trasmittente e il server il ricevente. Viene utilizzato TCP per trasferire in modo affidabile i messaggi di posta dal client al server sulla porta 25; il trasferimento è diretto. Ci sono tre fasi per il trasferimento: handshaking (saluto), trasferimento dei messaggi e chiusura. L'SMTP utilizza connessioni persistenti.

Esistono poi diversi protocolli di accesso alla posta:

- POP: Post Office Protocol. Utilizzato per autorizzazione e download
- IMAP: Internet Mail Access Protocol. Ha più funzioni e consente di manipolare i messaggi memorizzati sul server.
- HTTP: GMail, Hotmail, ecc...

**POP3** Suddiviso in due fasi: autorizzazione e transazione, è un protocollo senza stato tra le varie sessioni

**IMAP** Mantiene tutti i messaggi sul server, consente all'utente di organizzare i messaggi in cartelle e conserva lo stato tra le varie sessioni (nomi delle cartelle, associazione tra identificatori dei messaggi e nomi delle cartelle).

## 2.5 DNS

Il DNS, Domain Name System è un protocollo a livello applicazione che consente a host, router e server DNS di comunicare per risolvere i nomi dei siti web. Il DNS traduce un hostname come *www.facebook.com* in un indirizzo IP.

**Host aliasing** In alcuni casi, un host può avere più nomi, come nel caso dell'aliasing dei server mail.

Il DNS è un database distribuito implementato in una gerarchia di server DNS. Tipicamente, un server DNS radice viene contattato da un server DNS locale che non può tradurre un nome. A sua volta, il DNS radice contatta un server DNS autorizzato se non conosce la mappatura del nome, la ottiene e la restituisce al server DNS locale.

**Server TLD (top-level domain** Si occupano dei domini .com, .org, .net, .edu, ecc. e di tutti gli altri domini locali di alto livello, quali .uk, .fr, .ca e .jp.

**Server di competenza (authoritative server** Ogni organizzazione dotata di host internet pubblicamente accessibili (server web e di posta) deve fornire i record DNS di pubblico dominio che mappano i nomi di tali host in indirizzi IP; possono essere mantenuti dall'organizzazione o dal service provider.

Il server DNS locale non appartiene strettamente alla gerarchia dei server, ciascun ISP (università, società) ha un server DNS locale. Quando un host effettua una richiesta DNS, la query viene inviata al suo server DNS locale, che opera da proxy e inoltra la query in una gerarchia di server DNS.

Le query DNS possono essere effettuate in due modalità: iterativa o ricorsiva. Nel caso di query iterativa, il server contattato risponde con il nome del server da contattare, nel caso in cui non riesca a risolvere il nome; sarà quindi compito dell'host iniziale contattare quel server. Nel caso di query ricorsiva l'host affida il compito di tradurre il nome al server DNS contattato; in questo caso sarà il server stesso a contattare un secondo server per risolvere il nome, al quale può a sua volta assegnare il compito in modo iterativo.

Una volta che un server DNS impara la mappatura la mette nella cache, dove le informazioni vengono invalidate dopo un certo periodo di tempo. Tipicamente, un server DNS locale memorizza nella cache gli indirizzi IP dei server TLD, quindi i server radice non vengono visitati spesso.

Il DNS è un database distribuito che memorizza i record di risorsa (RR) che hanno il seguente formato: **(name, value, type, TTL)**

- Type = A
  - **name** è il nome dell'host (server)
  - **value** è l'indirizzo IP
- Type = NS
  - **name** è il dominio (**foo.com**)
  - **value** è il nome dell'host del server DNS di competenza di questo dominio
- Type = CNAME
  - **name** è il nome alias di qualche nome canonico (**www.ibm.com**)
  - **value** è il nome canonico (**servereast.backup2.ibm.com**)
- Type = MX
  - **value** è il nome del server di posta associato a **name**

### 2.5.1 Protocollo DNS

Il protocollo DNS è costituito da domande (query) e messaggi di risposta, entrambi con lo stesso formato.

#### Intestazione del messaggio (12 byte - 6 campi)

L'intestazione è costituita da un numero da 16 bit di identificazione della domanda, utilizzato uguale dalla risposta, e da un flag indicante:

- domanda o risposta
- richiesta di ricorsione
- ricorsione disponibile
- risposta di competenza

#### Resto del messaggio

- Campi per il nome richiesto e il tipo di domanda
- RR nella risposta alla domanda
- Record per i server di competenza
- Informazioni extra che possono essere usate

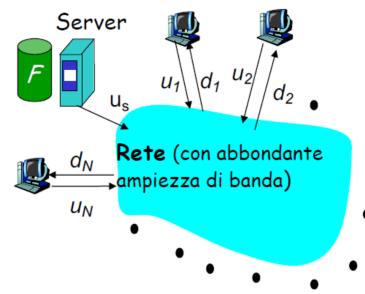
## 2.6 Condivisione di file P2P

Nell'architettura P2P pura non c'è un server sempre attivo, infatti coppie arbitrarie di host (peer) comunicano direttamente tra loro. I peer non devono essere sempre attivi e possono cambiare indirizzo IP.

### 2.6.1 Confronto tra architettura server client e P2P

#### Distribuzione di file: server-client

- Il server invia in sequenza  $N$  copie:
  - ❖  $\text{Tempo} = NF/u_s$
- Il client  $i$  impiega il tempo  $F/d_i$  per scaricare

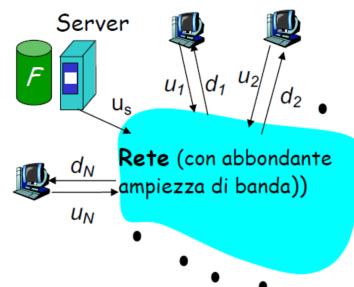


$$\text{Tempo per distribuire } F \text{ a } N \text{ client usando l'approccio client/server} = d_{cs} = \max \left\{ NF/u_s, F/\min(d_i) \right\}$$

aumenta linearmente con  $N$  peer

#### Distribuzione di file: P2P

- il server deve inviare una copia nel tempo  $F/u_s$
- il client  $i$  impiega il tempo  $F/d_i$  per il download
- Devono essere scaricati  $NF$  bit
- Il più veloce tasso di upload è:  $u_s + \sum u_i$



$$d_{P2P} = \max \left\{ F/u_s, F/\min(d_i), NF/(u_s + \sum u_i) \right\}$$

**Tracker** Il tracker tiene traccia dei peer che partecipano alla rete, dove un torrent è un gruppo di peer che si cambiano parti di un file.

Un file condiviso è diviso in più parti da 256 Kb, chiamati *chunk*. Il peer invia le sue parti a 4 vicini, quelli che gli stanno inviando alla frequenza più alta (aggiornata ogni 10 secondi). Successivamente, ogni 30 secondi viene scelto un peer a caso che riceve i chunk; oltre a questi 5 gli altri peer non riceveranno nulla.

**Query flooding** Ciascun peer indica i file che rende disponibili per la condivisione. Il messaggio di richiesta è trasmesso sulle connessioni TCP esistenti, il peer inoltra la richiesta e il messaggio di successo più trasmesso viene inviato sul percorso inverso.

## 2.7 Cloud Computing

L'architettura del cloud computing prevede uno o più server reali, generalmente in architettura ad alta affidabilità e fisicamente collocati presso i datacenter del fornitore del servizio.

### 2.7.1 CDN

Le CDN, o Content Delivery Networks, rappresentano una soluzione comune per la realizzazione di servizi su internet, la quale costruisce una rete "overlay" per la distribuzione di contenuti. Serve per memorizzare i dati il più vicino possibile ai consumatori in modo da ottimizzare le prestazioni di rete, ridurre la latenza ed evitare colli di bottiglia.

# Capitolo 3

## Il livello di Trasporto

### 3.1 Servizi a livello di trasporto

I servizi a livello di trasporto forniscono la comunicazione logica tra processi applicativi di host differenti. I protocolli di trasporto vengono eseguiti nei sistemi terminali: il lato invio scinde i messaggi in segmenti e li passa al livello di rete, il lato ricezione invece riassembra i segmenti in messaggi e li passa al livello applicazione.

- **Livello di rete:** si occupa della comunicazione logica tra host
- **Livello di trasporto:** si occupa della comunicazione logica tra processi, si basa sui servizi del livello di rete e li potenzia

**Demultiplexing** Nell'host ricevente, si occupa di consegnare i segmenti ricevuti alla socket appropriata.

**Multiplexing** Nell'host mittente, raccoglie i dati da varie socket, li incapsula con l'intestazione (usata poi nel demultiplexing).

L'host riceve i datagrammi IP; ogni datagramma ha un indirizzo IP di origine e uno di destinazione, e trasporta 1 segmento a livello di trasporto, ogni segmento ha un numero di porta di origine e un numero di porta di destinazione. L'host usa gli indirizzi IP e i numeri di porta per inviare il segmento alla socket appropriata.

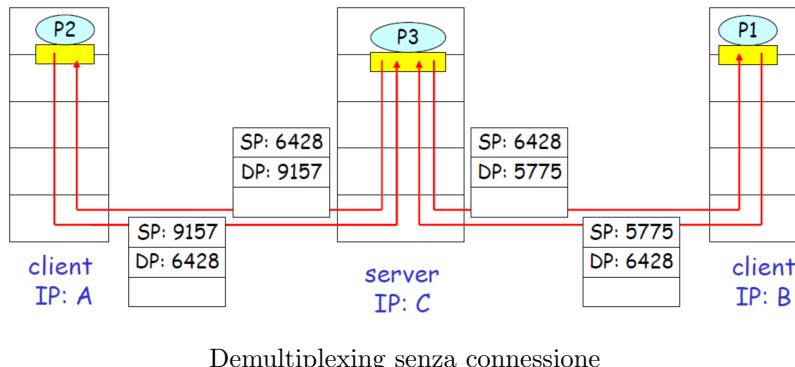
#### 3.1.1 Demultiplexing senza connessione

Il demultiplexing senza connessione utilizza una socket UDP identificata da indirizzo IP di destinazione e numero della porta di destinazione.

Quando l'host riceve il segmento UDP controlla il numero di porta nel segmento e poi lo invia alla socket correlata. Datagram IP con indirizzi IP e/o numeri di porta di origine differenti vengono inviati alla stessa socket.

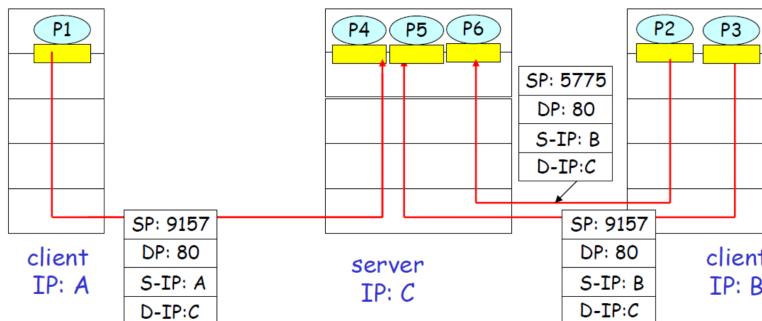


Struttura del segmento TCP/UDP



### 3.1.2 Demultiplexing orientato alla connessione

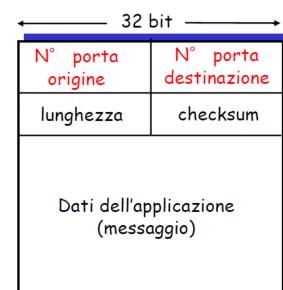
Il demultiplexing orientato alla connessione utilizza una socket TCP identificata da quattro parametri: indirizzo IP e porta d'origine e indirizzo IP e porta di destinazione. L'host ricevente usa tutti e quattro i parametri per inviare il segmento alla socket appropriata. Un host server può supportare più socket TCP contemporanee. I server web hanno socket differenti per ogni connessione client, con HTTP non-persistente si avrà invece una socket differente per ogni richiesta.



## 3.2 Trasporto senza connessione: UDP

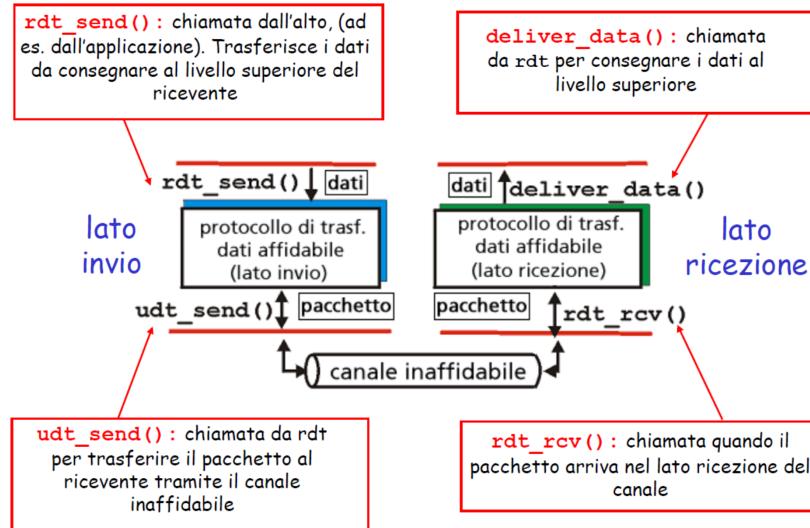
UDP, User Datagram Protocol, è un protocollo di trasporto "senza fronzoli", infatti ha un servizio di consegna best effort (miglior sforzo). Per questo i segmenti UDP possono essere perduti o consegnati fuori sequenza all'applicazione.

Essendo senza connessione non c'è handshaking tra mittente e destinatario, quindi ogni segmento UDP è gestito indipendentemente dagli altri.

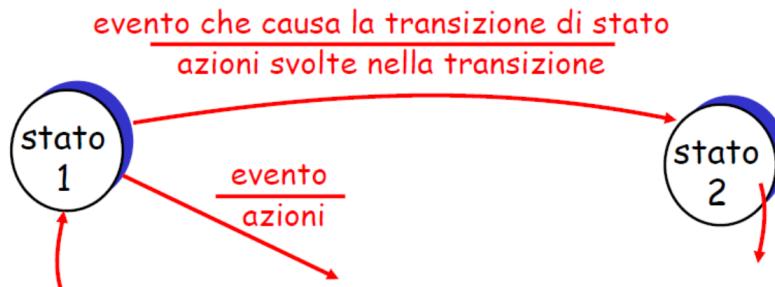


**Checksum UDP** Il checksum UDP serve per rilevare gli "errori" (bit alternati) nel segmento trasmesso, il segmento viene trattato come una sequenza di interi da 16 bit.

### 3.3 Principi del trasferimento dati affidabile



**Stato di Rdt** Lo stato successivo a quello corrente è determinato unicamente dall'evento successivo



#### 3.3.1 Rdt 1.0: trasferimento affidabile su canale affidabile

Il canale sottostante è perfettamente affidabile e è presente un automa distinto per mittente e ricevente.

#### 3.3.2 Rdt 2.0: canale con errori nei bit

Il canale sottostante potrebbe confondere i bit nei pacchetti, si utilizza quindi il checksum per rilevare gli errori. Una volta ricevuto, con la notifica positiva **ACK** il ricevente comunica esplicitamente al mittente che il pacchetto ricevuto è corretto mentre con la notifica negativa **NAK** comunica che il pacchetto contiene errori. Se il mittente riceve un NAK verrà ritrasmesso il pacchetto.

Con Rdt 2.0 sono stati introdotti nuovi meccanismi tra cui il rilevamento di errore e il feedback del destinatario (ACK e NAK).

Rdt 2.0 però ha un difetto fatale: se i pacchetti NAK e ACK sono danneggiati il mittente non saprà cos'è successo; non basta però ritrasmettere la notifica perché sono possibili duplicati. Il mittente ritrasmette quindi il pacchetto aggiungendo un numero di sequenza, e il ricevente lo scarterà se duplicato. Una volta inviato, il mittente aspetta la risposta del destinatario (*stop and wait*).

### 3.3.3 Rdt 2.1

Il mittente aggiunge un numero di sequenza al pacchetto, saranno sufficienti due numeri (0 e 1). Dovrà poi controllare se gli ACK/NAK sono danneggiati e ricordarsi se il pacchetto corrente ha numero di sequenza 0 o 1.

Il ricevente deve invece controllare se il pacchetto ricevuto è duplicato, lo stato indicherà se il numero di sequenza atteso è 0 o 1. Il ricevente non potrà però sapere se il suo ultimo ACK/NAK è stato ricevuto correttamente dal mittente.

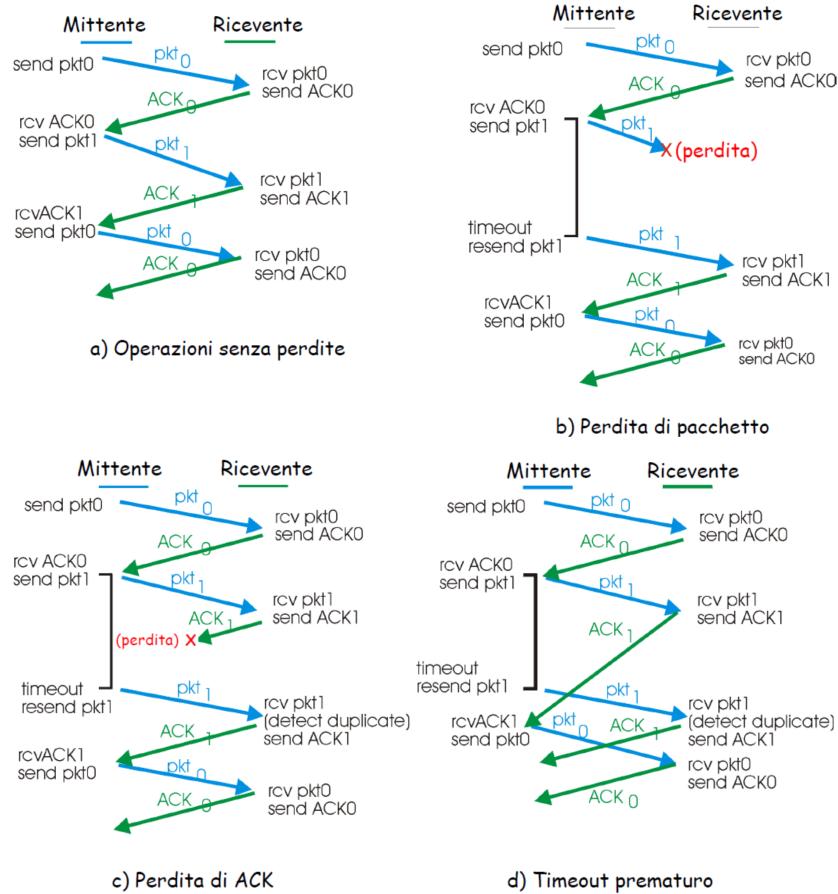
### 3.3.4 Rdt 2.2: un protocollo senza NAK

Ha le stesse funzionalità di Rdt 2.1, utilizzando solamente gli ACK. In sostituzione al NAK il destinatario invierà un ACK per l'ultimo pacchetto ricevuto correttamente, il destinatario invece deve includere esplicitamente il numero di sequenza con l'ACK. Un ACK duplicato presso il mittente determina la stessa azione del NAK, cioè ritrasmettere il pacchetto corrente.

### 3.3.5 Rdt 3.0: canali con errori e perdite

Può succedere che il canale sottostante smarrisca i pacchetti (dati o ACK). Per ovviare a ciò il mittente attende un ACK per un tempo ragionevole, dopodiché, se non ricevuto, ritrasmetterà il pacchetto.

Se il pacchetto (o l'ACK) è solo in ritardo la trasmissione sarà duplicata, ma il problema è già gestita dai numeri di sequenza, che il destinatario specificherà anche nei pacchetti da riscontrare; è necessaria l'introduzione di un contatore.



### 3.3.6 Pipelining

Il mittente ammette più pacchetti in transito ancora da notificare, il loro numero di sequenza deve essere incrementale ed è presente un buffering dei pacchetti presso il mittente e il ricevente. Ci sono due forme generiche di protocolli con pipeline: *Go-Back-N* e *Ripetizione selettiva*.

#### Go-Back-N

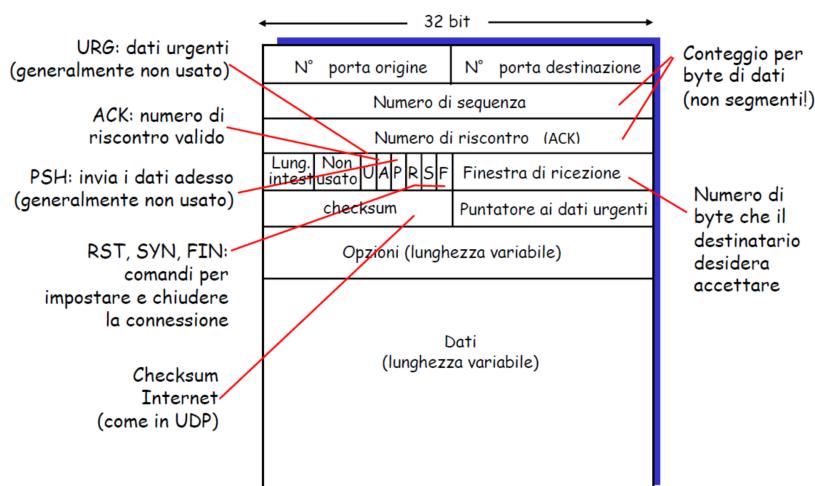
- Il mittente può avere fino a N pacchetti senza ACK in pipeline
- Il ricevente invia solo ACK cumulativi, non dà quindi l'ACK di un pacchetto se c'è un gap
- Il mittente ha un timer per il pacchetto più vecchio senza ACK.

#### Ripetizione selettiva

- Il mittente può avere fino a N pacchetti senza ACK in pipeline
- Il ricevente trasmette gli ACK solo sui singoli pacchetti.
- Il mittente mantiene un timer per ciascun pacchetto che non ha ancora ricevuto un ACK, che alla scadenza farà ritrasmettere solo i pacchetti senza ACK.
- Il ricevente accusa la ricevuta di ciascun singolo pacchetto.

## 3.4 Trasporto orientato alla connessione: TCP

- Implementa l'**Rdt 3.0** con pipelining
- **Connessione punto-punto:** un mittente e un destinatario
- Il flusso di byte è **affidabile** e in sequenza
- Il **controllo di flusso e di congestione** del TCP definiscono la dimensione della finestra di **pipelining**
- **Full-Duplex:** il flusso dei dati è bidirezionale nella stessa connessione e viene definita la dimensione massima del segmento (*MSS*)
- **Orientato alla connessione:** l'handshaking inizializza lo stato di mittente e destinatario prima di scambiare i dati
- **Flusso controllato:** il mittente non sovraccarica il destinatario



Alcuni dettagli sul segmento TCP:

- **Lunghezza dell'intestazione:** serve per sapere se ci sono informazioni nella parte opzionale, in caso negativo sarà di default 20 ( $5 \text{ righe} \cdot 4 \text{ byte} = 20$ )
- **Numero di sequenza:** numero del primo byte del segmento nel flusso di byte
- **ACK:** numero di sequenza del prossimo byte atteso
- Per calcolare il **timeout** si usa una media esponenziale ponderata

Il trasporto TCP crea un servizio di trasferimento dati affidabile sul servizio inaffidabile di IP. Le ritrasmissioni sono avviate da eventi di timeout e ACK duplicati.

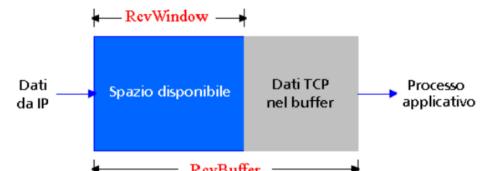
Evento presso il destinatario	Azione del ricevente TCP
Arrivo ordinato di un segmento con numero di sequenza atteso. Tutti i dati fino al numero di sequenza atteso sono già stati riscontrati.	ACK ritardato. Attende fino a 500 ms l'arrivo del prossimo segmento. Se il segmento non arriva, invia un ACK.
Arrivo ordinato di un segmento con numero di sequenza atteso. Un altro segmento è in attesa di trasmissione dell'ACK.	Invia immediatamente un singolo ACK cumulativo, riscontrando entrambi i segmenti ordinati.
Arrivo non ordinato di un segmento con numero di sequenza superiore a quello atteso. Viene rilevato un buco.	Invia immediatamente un ACK duplicato, indicando il numero di sequenza del prossimo byte atteso.
Arrivo di un segmento che colma parzialmente o completamente il buco.	Invia immediatamente un ACK, ammesso che il segmento cominci all'estremità inferiore del buco.

Il timeout spesso è relativamente lungo, si ha quindi un lungo ritardo prima che venga ritrasmesso il pacchetto perduto.

Gli ACK perduti vengono rilevati tramite gli ACK duplicati: il mittente spesso invia molti segmenti, se un segmento viene smarrito è probabile che ci saranno molti ACK duplicati. Se il mittente riceve 3 ACK duplicati per lo stesso dato si suppone che il segmento che segue il dato riscontrato è andato perduto e rispedirà quindi il pacchetto prima che scada il timer (metodo di **trasmissione rapida**).

### 3.4.1 TCP: controllo di flusso

Il lato ricevente della connessione TCP ha un buffer di ricezione, quindi il processo applicativo potrebbe essere rallentato dalla lettura del buffer. Con il controllo di flusso il mittente non vuole sovraccaricare il buffer del destinatario, trasmettendo troppi dati troppo velocemente. Il servizio di corrispondenza delle velocità indica che la frequenza di invio deve corrispondere alla frequenza di lettura dell'applicazione ricevente.



Il destinatario comunica lo spazio disponibile includendo il valore **RcvWindow** nei segmenti, il mittente limita i dati non riscontrati a **RcvWindow** e garantisce che il buffer di ricezione non vada in overflow.

### 3.4.2 Gestione della connessione

La connessione viene gestita mediante un Handshake a tre vie:

1. Il client invia un segmento SYN al server che specifica il numero di sequenza iniziale, non viene inviato nessun dato
2. Il server riceve il SYN e risponde con un segmento SYNACK e successivamente alloca i buffer
3. Il client riceve SYNACK e risponde con un ACK che può anche contenere dati

Per chiudere una connessione:

1. Il client invia un segmento di controllo FIN al server
2. Il server riceve il segmento FIN e risponde con un ACK, chiude la connessione e invia un FIN
3. Il client riceve il FIN e risponde con un ACK
4. Il server riceve l'ACK e chiude la connessione

## 3.5 Principi del controllo di congestione

Per congestione si intende quando troppe sorgenti trasmettono troppi dati a una velocità talmente elevata che la rete non è in grado di gestirli. I sintomi della congestione possono essere pacchetti smarriti (causati da overflow nei buffer dei router) o lunghi ritardi (accodamento nei buffer).

I due principali approcci al controllo di congestione sono:

- **Controllo di congestione punto-punto**
  - Nessun supporto esplicito dalla rete
  - La congestione è dedotta osservando le perdite e i ritardi nei sistemi terminali
  - Metodo adottato da TCP
- **Controllo di congestione assistito dalla rete**
  - I router forniscono un feedback ai sistemi terminali
  - Utilizzato un singolo bit per indicare la congestione
  - Viene comunicata in modo esplicito al mittente la frequenza trasmissiva

## 3.6 Controllo di congestione in TCP (AIMD)

Il controllo di congestione in TCP viene effettuato mediante l'approccio AIMD; ovvero incremento additivo e decremento moltiplicativo.

Consiste nell'aumentare il tasso trasmissivo sondando la rete fino a quando non si verifica una perdita. Secondo l'**incremento adattivo** fa aumentare la CongWin di 1 MSS a ogni RTT in assenza di perdita, mentre secondo il **decremento moltiplicativo** riduce a metà CongWin dopo un evento di perdita. La formula diventa quindi approssimativamente:

$$\text{Frequenza d'invio} = \frac{\text{CongWin}}{\text{RTT}} \text{ byte/sec}$$

CongWin è una funzione dinamica della congestione percepita. Il mittente percepisce la congestione dopo un evento di perdita, quindi un timeout o una ricezione di 3 ACK duplicati, e riduce di conseguenza la frequenza di invio (CongWin).

Vengono utilizzati tre meccanismi: AIMD, partenza lenta e reazione agli eventi di timeout.

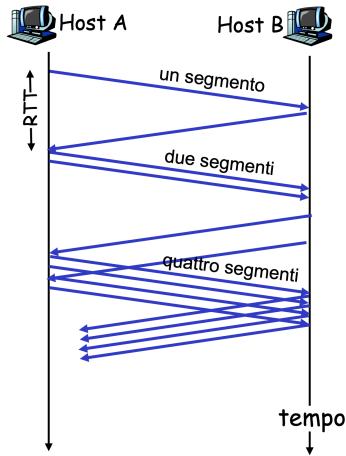
### 3.6.1 Partenza lenta

Quando si stabilisce una connessione ( $CongWin = 1$  MSS) la frequenza aumenta in modo esponenziale fino a quando non si verifica una perdita: infatti  $CongWin$  raddoppia a ogni RTT incrementandolo per ogni ACK ricevuto.

Dopo 3 ACK duplicati (perdita),  $CongWin$  è ridotto a metà e la finestra cresce linearmente. Se si verifica però un timeout,  $CongWin$  viene impostato a 1 MSS crescendo in modo esponenziale fino a una soglia, dopo la quale crescerà linearmente.

La filosofia seguita è quella che 3 ACK duplicati indicano la capacità della rete di consegnare qualche segmento, mentre un timeout prima di 3 ACK duplicati è "più allarmante".

La soglia impostata è variabile, ma in caso di perdita la soglia diventa  $\frac{1}{2}$  di  $CongWin$  appena prima dell'evento.



### 3.6.2 Riassunto: controllo di congestione

- Quando  $CongWin$  è sotto la soglia, il mittente è nella fase di **partenza lenta**; la finestra cresce in modo esponenziale.
- Quando  $CongWin$  è sopra la soglia, il mittente p nella fase di **congestion avoidance**; la finestra cresce in modo lineare.
- Quando si verificano **tre ACK duplicati**, il valore della soglia viene impostato a  $CongWin/2$  e  $CongWin$  viene impostata al valore della soglia.
- Quando si verifica un **timeout**, il valore della soglia viene impostato a  $CongWin/2$  e  $CongWin$  è impostata a 1 MSS.

### 3.6.3 Throughput TCP

Il throughput medio di TCP varia in funzione della dimensione della finestra e di RTT. Se, dopo una perdita, la finestra è  $W$ , il throughput sarà  $W/RTT$ . Dopo la perdita la finestra diventerà  $W/2$  e il throughput di conseguenza diventa  $W/2RTT$ . Il throughput medio è quindi  $0.75 W/RTT$ .

### 3.6.4 Equità di TCP

**Equità** se  $K$  sessioni TCP condividono lo stesso collegamento con ampiezza di banda  $R$  (collo di bottiglia), ogni sessione dovrà avere una frequenza trasmittiva media pari a  $R/K$ .

TCP infatti è equo perché con due connessioni in concorrenza l'incremento additivo ha una pendenza pari a 1, mentre il decremento moltiplicativo riduce il throughput in modo proporzionale.

Proprio per questo le applicazioni multimediali spesso usano UDP, poiché non vogliono che il loro tasso trasmittivo venga ridotto ma tollerano la perdita di pacchetti.