*Parte 1*

**- DSL (Digital subscriber line)**

\* Usa la linea telefonica, quando arriva a ISP(fornitore di servizi Internet) data va a internet, telefono a telefono net

\* Ogni endpoint ha la sua linea che lo connette all'isp

\* Up: < 1 Mbps - Down: < 10 Mbps

**- Cable Network**

\* HFC, Hybrid fiber coax

\* Gli endpoint sharano la linea per arrivare a isp (data + TV = due frequenze, stesso cavo)

\* Up: < 2Mbps - Down: < 30 Mbps

**- Wireless Access Networks**

\* LAN → Private, 100 ft, 802.11b/g/n

\* WAN → Cellular operator, 10 Km, 3G 4G..

**- Packet Transmission Delay** = Tempo per trasmettere L-bit pacchetti attraverso la linea = L (bits) / R (bits/s)

**- Mezzo Fisico** (aka Physical Link, ciò che giace fra sender/receiver)

\* Guided Media (mezzo solido)

+ Twisted Pair (TP, doppino) → Due cavi di rame isolati (v5 fino 1 Gbps, v6 fino 10 Gbps)

+ Cavo Coassiale → 2 rame concentrici, bidirezionale, HFC, vari canali

+ Fibra ottica → Porta luce, ogni impulso è un bit, error rate basso, immune agli EMP

\* Unguided Media (segnale libero)

+ Radio → Trasportato dallo spettro elettromagnetico, bidirezionale, affetto da ostruzione

**- Network Core Functions**

\* Routing → Definisce il percorso preso dal pacchetto

\* Forwarding → Da router a quello successivo

**- Packet Switching**

\* Store and forward

+ Intero pacchetto deve arrivare a router per essere trasmesso

+ End/end delay = 2L/R (zero delay propagazione)

\* Queuing and loss

+ Se la trasmissione host → router è maggiore di router → internet i pacchetti sono accodati

+ Se il buffer del router si filla pacchetti in eccesso droppati, ritrasmessi da source, node o nessuno

+ dnodal = dproc + dqueue + dtrans + dprop

1. dproc = nodal processing (controlla errori bit, determina output speed)

2. dqueue = delay queue (La/R, a = average packet arrival rate, → 0 bene, → 1 male)

3. dtrans = transmission delay (L/R)

4. dprop = propagation delay (d/s → d = lunghezza mezzo fisico, s = veloc. prop. = 2\*10^8 m/s)

**- Circuit Switching** (telefono)

\* Risorse dedicate

+ Un canale è dedicato ad un host

+ Se il canale non è utilizzato è "sprecato"

**- Packet vs Circuit Switching**

\* Circuit

+ 1 Mbps, 10 utenti, ognuno 100 Kbps → Problema = spreco risorse

\* Packet

+ 1 Mbps, 35 utenti, probabilità 10 allo stesso tempo bassa → Problema = congestione

**- Struttura di Internet**

\* Network → ISP/RegionalNetwork → ISP tier 1 interconnessi tramite IXP (Internet exchange point)

\* Content provider (Google ecc..) possono creare una propria net per connettersi a internet, bypass ISP

**- Throughput**

\* Rate (bits/s) di trasmissione fra client e server

+ Instantáneo

+ Medio

\* Il throughput sarà sempre il rate minore fra quello del client e del server (bottleneck)

**- Internet Protocol Stack**

\* Levels

+ Application → Es: FTP, SMTP, HTTP..

+ Presentation (ISO/OSI) → Applicazione interpreta il significato dei dati, Es: encryptio, compression..

+ Session (ISO/OSI) → Synchronization, recovery of data

+ Transport → Comunicazione interprocesso, Es: TCP, UDP..

+ Network → Routing da sorgente a destinazione, Es: IP..

+ Link → Data transfer fra hardware vicino, Es: Ethernet, 802.111 (Wi-Fi)..

+ Physical → Cavo

\* Encapsulation

+ Application → Message

+ Transport → Segment

+ Network → Datagram [ROUTER]

+ Link → Frame [SWITCH, ROUTER]

**- Network Security**

\* Internet inizialmente era una cosa "intima", tutti si fidavano di tutti, ora no

\* Malwares

+ Virus → Autoreplicante, ricevuto tramite email

+ Worm → Autoreplicante, riceve file da eseguire

+ Spyware → Registra tastiera, siti visitati..

\* Botnet & DoS

\* Packet Sniffing

\* IP Spoofing → Manda pacchetti con IP sorgente arbitrario

*### Parte 2*

**- Application Architectures**

\* **Client-Server**

+ Server → Sempre on, IP permanente

+ Client → Parla con server, non sempre on, IP dinamico, non comunicano fra loro

+ Tempo di distribuzione di file (size F) a N client

- Server → N \* F / UploadCapacity

- Client → F / MinCliDownRate

- Totale → Max{Server, Client}

\* **Peer to peer** (P2P)

+ Non esiste un server sempre on

+ Tutti gli host parlano fra loro

+ I peer si scambiano servizi, più peer presenti, più "forza" ha la rete

+ IP dinamico

+ Tempo di distribuzione di file (size F) a N peer

- "Server" → Almeno deve caricare una copia → F / UploadCapacity

- "Client" → Ogni client deve scaricare una copia → F / MinCliDownRate

- Clients → Tot download = N \* F bits → MaxUpRate = UploadCapacity + sum(each\_peer)

- Totale → Max{"Server", "Client", N \* F / Clients}

+ File Sharing (BitTorrent)

- I file sono divisi in chunk da 256 Kb

- Ogni peer manda/riceve i chunk di un file con tutta la lista dei peer del file

- Un tracker gestisce le liste di peer

- Richiedere chunk

\* Ogni peer ha una lista di chunk di un file

\* Periodicamente client A richiede la lista dei chunk ai peer

\* Client A richiede i chunk mancanti ai peer, i più rari per primi

- Mandare chunk (tit-for-tat)

\* Client A manda chunk alla top 4 per velocità dei peer che gli sharano i file

+ Agli altri niente (choke)

+ Rivalutato ogni 10 secondi

\* Ogni 30 secondi scegli a caso un altro peer a cui mandare chunk

+ Speri l'unchoke

+ Potrebbe joinare la top 4

**- Comunicazione fra processi**

\* Same host → Inter process communications

\* Different host → Messaggi

+ P2P → Gli host hanno applicazione sia server che client

+ Client/Server → Client (processo che inizializza comunicazione), server (processo wait connection)

**- Sockets** (Porta fra processo applicazione e protocollo di trasporto)

oggetto software che permette l'invio e la ricezione di dati, tra host remoti o tra processi locali

\* Host A pusha messaggio a sua socket

\* Socket pusha messaggio a mezzo di trasporto esterno

\* Messaggio arriva ad altra socket che lo "spacchetta"

\* Programmazione

+ UDP

- Non esiste handshake

- Sender → Ad ogni pacchetto attacho IP e porta destinazione

- Receiver → Ad ogni pacchetto estraggo IP e porta sorgente

- Client

1) Creo la socket UDP

2) Input del messaggio

3) Attacco al messaggio IP e porta destinazione poi invio tramite socket

4) Leggo la risposta dalla socket

5) Chiudo socket

- Server

1) Creo la socket UDP

2) Bindo la socket ad una porta locale

In loop:

3) Leggo quello che mi arriva dalla socket salvandomi IP e porta sender

4) Rispondo

+ TCP

- Il server e la sua socket devono essere raggiungibili

- Quando il client crea la socket specificando IP e porta destinazione la connessione parte

- Quando il server riceve la richiesta di connessione crea una socket apposta per quel client

- Client

1) Creo la socket TCP con porta destinazione e mi connetto all'IP destinazione

2) Mando il messaggio e mi arriva la risposta

- Server

1) Creo la socket di benvenuto e mi metto in ascolto

In loop:

2) Quando mi arriva connessione creo socket dedicata

3) Leggo quello che mi è stato inviato

4) Chiudo la socket del client, non quella di benvenuto

**- Addressing Processes**

\* Per mandare messaggi ho bisogno di

+ Indirizzo IP host ricevente

+ Porta su cui il processo ricevente è bindato

- **Definizione dei** **messaggi fra applicazioni**

\* Tipo di messaggio → Richiesta/risposta

\* Sintassi messaggio → Campi del messaggio e loro disposizione

\* Semantica messaggio → Significato dei campi

\* Regole di invio richiesta/risposta

**- Servizi di trasporto necessari**

\* Integrità dei dati → File 100% integri, audio/video anche no

\* Timing → Videogiochi low delay

\* Throughput → Un film in 4K lo guardi se hai un buon throughput, una pagina web la scarichi in ogni caso

\* Sicurezza → Encryption, dati sani..

**- Protocolli**

\* **TCP**

+ Trasporto affidabile - lento

+ Flow control → Sender non intasa receiver

+ Congestion control → Se congestione il sender rallenta

+ Connection oriented → Connessione fra host A e host B

+ No sicurezza

+ No throughput minimo garantito

\* **UDP**

+ Trasporto inaffidabile - veloce

+ No flow control

+ No congestion control→

+ No connessione

+ No sicurezza

+ No throughput minimo garantito

**- Sicurezza TCP**

\* **SSL** (Secure Socket Layer - livello applicazione) versione precedente del TLS

metodi di trasmissione sicura di dati: negoziazione tra il mittente e il destinatario di una chiave di crittografia con cui vengono criptati i dati trasmessi, scambio delle chiavi e cifratura

+ Encryption

+ Dati sani

+ Autenticazione degli endpoint

**- RTT** (Round trip time) → Tempo che ci impiega un pacchetto di fare avanti e indietro fra client e server

**- HTTP**

caricare pagine web utilizzando link ipertestuali

\* Livello applicazione

\* Client (Browser richiede, riceve, visualizza) - Server (risponde con oggetti richiesti)

\* TCP → Connessione a porta 80, accettata, scambio messaggi, connessione chiusa

\* Stateless → Server non mantiene info sulle vecchie richieste del client

\* Connessione

+ Non-persistent → Un oggetto per ogni connessione TCP (pagina con 2 immagini = 3 connessioni)

- Tempo di risposta → 1 RTT init connection + 1 RTT richiedo file + trasmetto file

- Contro → 2 RTT per oggetto, sovralavoro per ogni connessione TCP..

+ Persistent → Oggetti multipli per una connessione TCP

\* Messages

+ Request

- Formato

```

Request line → Metodo URL Versione\r\n

Header lines → Header\_field\_name .. Value\r\n

\r\n

Body

```

- Metodi

\* GET

\* POST

\* HEAD → Lascia oggetto richiesto senza risposta

\* PUT (1.1) → Upload il file definito nel body nell'URL field

\* DELETE (1.1) → Deleta il file dell'URL field

- Input forms

\* POST method → Il form input è uppato al server tramite il body

\* URL method → Usa GET e in URL field crafta la query di richiesta

+ Response

- Formato

```

Status line → Protocol Status\_code Status\_phrase\r\n

Header lines

\r\n

Data

```

- Status codes

\* 200 OK → Ok, data nel messaggio

\* 301 Moved Permanently → Object spostato, nuova location in messaggio

\* 400 Bad Request → Il server non ha capito

\* 404 Not Found

\* 505 HTTP Version not SUpported

\* **Cookies**

dati sugli utenti memorizzati sul computer e utilizzati per migliorare la navigazione (info personalizzate)

1) Client richiede qualcosa a server

2) Server risponde e nell'header di risposta mette "set-cookie: N" e salva in suo DB

3) Client nella prossima richiesta incorpora cookie in header line e browser salva cookie

\* **Web-caches** (proxy server)

memorizza copie di documenti richiesti dagli utenti. Riduce tempo risposta e traffico

+ Funzionamento

1) Client manda richiesta alla cache

2) Se risorsa esiste → viene ritornata

else → contatti il server madre

+ La cache è installata dall'ISP (es: uni, ISP regionale..)

+ Solitamente 4/10 del traffico soddisfatto da cache (Latenza = 0.4 \* tempo\_cache + 0.6 \* tempo\_internet)

+ Conditional GET

1) Cache richiede risorsa con header → If-modificed-since: data

2) Server risponde: if non modificata → 304 Not Modified

else → 200 OK

**- E-Mail**

\* Componenti

+ User agent (mail reader) → Outlook, Thunderbird..

+ Mail servers

- Mailbox → Contiene i messaggi

- Message queue → ALL outgoing messages

**+ SMTP**

protocollo che gestisce la posta in uscita. (es. app email si collega al server tramite smtp e invia messaggi)

- TCP, port 25

- Comando/risposta come HTTP

- Connessione persistente

- I messaggi devono essere in ASCII 7 bit

- Client tells server A to send mail to server B, ogni passaggio in 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: . (End = CRLF.CRLF)

S: 250 Message accepted for delivery

C: QUIT

S: 221 hamburger.edu closing connection

```

- Transfer → Handshaking, transfer of message, close

- Formato mail

```

Header:

To

From

Subject

Blank line

Body:

Caratteri ASCII

```

- Protocolli di accesso

\* **POP** (Post office protocol) → Autorizzazione, download

+ Download-and-delete → Appena scarichi il server non ha piu

+ Download-and-keep → Scarichi e il server mantiene

\* **IMAP** (Internet mail access protocol) → Manipolo anche le mail sul server

+ Tutto storato sul server, tu ci lavori sopra

+ Cartelle

\* HTTP → Protonmail, Yahoo..

**- DNS** (Domain name system)

sistema utilizzato per tradurre gli indirizzi ip in nomi

\* Database distribuito implementato gerarchicamente di tanti name servers

\* Servizi

+ Hostname to IP address translation

+ Host aliasing

+ Mail server aliasing

+ Load distribution → Tanti IP corrispondono ad un hostname

\* Tipologie

+ Root name server

- Contattato dai local name server

- Se non ha l'indirizzo lo richiede agli authoritative name server

+ Authoritative DNS server

- Gestito da un'azienda o un ISP (tipo DNS di Amazon)

+ Top level domain server (TLD)

- com, org, net, edu, aero, jobs, museums, it, uk, nl..

+ Local DNS

- Telecom, Uni..

- Quello che riceve la richiesta dal client

\* Funzionamento

```

Iterativo:

1) Host contatta local DNS

2) Local DNS contatta in ordine:

° Root DNS

° TLD DNS

° Authoritative DNS

Ricorsivo:

1) Host contatta local DNS

2) Local DNS contatta root DNS

3) Root DNS contatta TLD DNS

4) TLD DNS contatta authoritative DNS

5) Ritorno il risultato riseguendo il percorso

```

\* DNS caching

+ Se un server acquisisce una risorsa la mantiene per un tot di tempo (TTL)

+ TLD DNS sono cachati nei local DNS, così la root non è sempre querata

\* DNS resource record (RR)

+ Formato → (name, value, type, ttl)

+ Tipi

- A

\* Name = Hostname

\* Value = IP

- NS

\* Name = Domain (Es: foo.com)

\* Value = Hostname di authoritative DNS

- CNAME

\* Name = Alias (Es: www.ibm.com)

\* Value = Canonical name (Es: servereast.backup2.ibm.com)

- MX

\* Value = Name del mailserver associato con "Name"

\* Formato messaggi

+ Richieste e risposte hanno lo stesso formato

+ Header

```

2 Bytes 2 Bytes

Identification Flags

Identification → ID richiesta/risposta

Flags → Richiesta/risposta

→ Ricorsione desiderata/possibile

→ Risposta da authoritative

```

+ "Body"

```

4 Bytes each

1) Nome, tipo di campi per una query

2) Record raws (RRs) in risposta

3) Record per server authoritative

4) Additional info

```

\* Inserting records in DNS

```

Es: Insert two RRs into .com TLD server:

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

(dns1.networkutopia.com, 212.212.212.1, A)

```

\* Attacking DNS

+ DDoS

- Bombarda root → Molto difficile e in ogni caso ci sarebbero i TLD di ripiego

- Bombarda TLD servers

+ Redirect

- MITM → Intercetta le query al DNS

- DNS poisoning → Manda richieste del cazzo al DNS che le cacha riempiendolo

+ Use DNS for DDoS

- Manda query con IP spoofato, così chi hai spoofato viene dossato dal DNS

**- Multimedia**

\* Video

+ Coding (decrease bits used to encode image)

- Spatial coding → Manda il colore e quante volte è ripetuto

- Temporal coding → Manda solo le differenze fra frame i e i+1

+ Bit rate

- CBR → Constant bit rate

- VBR → Variable bit rate → Usando spatial e temporal coding

+ Streaming

- **DASH** (Dynamic Adaptive Streaming over HTTP)

tecnica che consente lo streaming di alta qualità di contenuti multimediali su Internet erogati da server Web HTTP

\* Server

+ Divide il file in vari chunk

+ Ogni chunk encodato con rate diverso

+ Manifest → Da il link ai vari chunk

\* Client

+ Periodicamente misura la banda

+ Consultando il manifest sceglie in base alla banda il chunk da scaricare

+ Intelligente, determina

- Quando richiedere un chunk in modo che non ci sia overflow/starvation

- Che encoding richiedere (in base alla banda)

- Dove richiedere il chunk in base al carico dei vari server

**- CDN** (Content Distribution Networks)

un'infrastruttura di rete estesa a livello globale, composta da vari server dislocati su diverse reti, che si occupano di distribuire dei particolari contenuti tramite Internet

\* Fornisci le copie dei contenuti da vari server sparsi per il mondo

\* Scegli il nodo del CDN migliore (più vicino, meno congestionato)

\* Funzionamento

1) Client A si connette a NetCinema e trova url video (http://netcinema.com/6Y7B23V)

2) Risolvo l'URL tramite il DNS locale

3) Il DNS di NetCinema torna l'url che punta a KingCDN (http://KingCDN.com/NetC6y&B239

4) Risolvo l'URL tramite il DNS di KingCDN e mi torna l'url del video sui server di KingCDN

*Parte 3*

**- Transport Layer**

\* Send-side → Divide il messaggio dell'app layer in segmenti passati al network layer

\* Receive-side → Riassembla i segmenti ricevuti e li passa all'app layer

\* Transport vs Network layer

+ Network → Comunicazione logica fra host

+ Transport → Comunicazione logica fra processi

**\* Multiplexing/Demultiplexing**

+ Multiplexing → Raccogli i dati da socket multiple e aggiungi header di trasporto

più canali trasmissivi in ingresso combinando più segnali analogici in un solo segnale £

+ Demultiplexing → Usa gli header per distribuire alle socket i corrispondenti pacchetti

- Funzionamento

\* Host riceve IP datagram

+ Ogni datagram ha l'IP sorgente e destinazione

+ Ogni datagram trasporta un segmento del transport layer

+ Ogni segmento ha la porta sorgente e destinazione

+ Host usa gli IP e le porte per condurre il segmento alla socket corrispondente

\* Connectionless (UDP)

+ Invio → Creo socket bindata a porta e ci faccio passare datagram in uscita (con dest IP e port)

+ Ricezione → Leggo porta da datagram e lo spedisco a socket corretta (IP diversi, same socket)

\* Connection oriented (TCP)

+ Socket TCP definite da quaterne

- Source IP

- Source port

- Dest IP

- Dest port

+ Invio → Più o meno uguale a UDP

+ Ricezione

- Il ricevente usa tutte le 4 info per dirigere il segmento verso la sua socket

- Il ricevente può supportare varie socket simultanee (ognuna definita da quaterna)

- Web server hanno socket diverse per ogni client (HTTP non pers → socket per ogni richiesta)

\* **UDP** (User Datagram Protocol)

+ Header

```

32 bit

16 bit 16 bit

Source port Dest port

Length Checksum

Payload

```

+ **Checksum**

sequenza di bit che, associata al pacchetto trasmesso, viene utilizzata per verificare l'integrità di un dato o di un messaggio che può subire alterazioni durante la trasmissione sul canale di comunicazione

- Sender

\* Considera il segmento come una sequenza di interi a 16 bit

\* Checksum = Addizione a complemento a 1 del contenuto del segmento

- Receiver

\* Calcola il checksum e lo compara al checksum del segmento, se non combacia errore

\* NB → Se riporto va tolto e +1

\* **Principi di trasferimento reliable**

+ Stop and wait (Sender manda pacchetto e aspetta risposta receiver)

- Checksum

- ACK → ACK for last packet received correct, duplicate ACK to retransmit current packet (No NAK)

- Sequence numbers

\* To solve ACK/NAK loss/corruption

\* 0, 1

- Timeout

\* Il sender aspetta tempo x prima di ritrasmettere

- Performance → Molto male poiché mando solo un pacchetto alla volta

\* Link = R = 1 Gbps, Delay propag = 15 ms, L = 8000 bit

\* Tempo di trasporto = D = L / R = 8 microsec

\* Utilizzo del sender (U) = L / R / ((RTT = 2 \* Delay propag) + R / L) = 0.00027

\* In pratica ho 27 Kb/s di throughput

+ Pipelining (Il sender concede multipli pacchetti in volo to be ACKed)

- Maggior range sequence number + buffering

- Con N pacchetti mandati contemporaneamente l'utilizzo (U) viene moltiplicato per N

- Go Back N

\* Il sender può avere al massimo N pacchetti non ACKati (window di dimensione N)

\* Receiver manda ACK cumulativi (ACK(n) = ACK tutti i pacchetti fino ad n, NB: expectedseqnum)

\* Il sender ha timer per più vecchio pacchetto unACKato, quando scade ritrasmette tutti gli unACKed (timeout(n) = ritrasmetti n e tutti i pacchetti della window con sequence number maggiore di n)

\* Out-of-order-packets → Scartati e reACK dell'ultimo pacchetto integro in ordine

- Selective Repeat

\* Il sender può avere al massimo N pacchetti non ACKati

\* Il receiver manda un ACK per ogni pacchetto, quindi buffer contiene pacchetti anche out of order

\* Il sender ha timer per ogni pacchetto non ACKato, quando scade lo ritrasmette

\* Problema → Se io ho una window di N = 3, mando 3 pacchetti (0,1,2), arrivano, e l'ACK non torna indietro io rimanderò gli stessi 3 pacchetti (0,1,2), ma per il ricevitore saranno i 3 successivi (0,1,2,0,1,2)

\* **TCP** (Transmission Control Protocol)

+ Reliable protocol over unreliable channel

+ Point to point → Un sender, un receiver

+ Pipelined → In base alla congestione e al flow control viene definita la window size

+ Full duplex → Data flow bidirezionale sulla stessa connessione

+ Connection oriented

+ Flow controlled → Il sender non dossa il receiver

+ Struttura pacchetti

```

32 Bit

Source\_port Destination\_Port

Sequence\_number → Numero del primo byte nel segmento data

ACK\_number → Sequence number atteso del prossimo pacchetto, cumulativi

Head\_len Unused U A P R S F Rec\_window → U = urgent, A = ACK n. valid, P = PSH = push data now

R = RST S = SYN F = FIN (connection commands)

Checksum Urg\_data\_poiner

Options

Payload

```

+ Sequence number & ACKs

```

TELNET

Host A Host B

User scrive "c" ----------------------------------->

Seq = 42, ACK = 79, Data = c

<----------------------------------- Server ACKa la "c" e la ritorna

Seq = 79, ACK = 43, Data = c

User ACKa "c" ----------------------------------->

Seq = 43, ACK = 80

```

+ Round trip time e timeout

- La stima non deve essere troppo bassa rispetto originale (altrimenti ritrasmissioni continue inutili)

- La stima non deve essere troppo alta rispetto originale (altrimenti reazione lenta alle perdite)

- SampleRTT → Tempo da una trasmissione alla ricevuta del suo ACK

- EstimatedRTT = (1 - a) \* EstimatedRTT + a \* SampleRTT (a = 0.125)

- SafetyMargin → DevRTT = (1 - b) \* DevRTT + b \* |SampleRTT - EstimatedRTT| (b = 0.25)

- Timeout totale = EstimatedRTT + 4 \* SafetyMargin

+ Eventi del sender

- Dati ricevuti da app layer

\* Crea un segmento con un sequence number

\* Sequence number = Numero del primo byte nel segmento data

\* Start timer se non sta gia andando (il più vecchio pacchetto non ACKato)

- Timeout

\* Ritrasmetti il segmento andato in timeout e restarta timer

- ACK ricevuto

\* Se l'ACK corrisponde al precedente pacchetto non ACKato

+ Aggiorna pacchetti ACKati, anche quelli precedenti saranno ACKati

+ Restarta timer se ci sono pacchetti non ACKati

+ Scenario di ritrasmissione

- Seq + Data = Ret ACK

- Cumulative ACK = Sendo 2 pacchetti, ricevuti, ACK del primo lost, secondo ACK arriva, si continua

+ Fast retransmit

- Se il sender riceve 3 ACK stesso pacchetto → Resenda pacchetto non ACKato con sequence number minore

- Si fa questo perché il pacchetto non ACKato è sicuramente stato perso

+ Flow control

- Il sender non intasa il receiver

- Questo avviene perche il receiver controlla la quantità di dati mandata dal sender

\* Il receiver nell'header TCP attacha la "rwnd" value (lo spazio libero del buffer di ricezione) con cui il sender si autoregola

+ Connection management

- Handshake

\* 2-way → A call B che risponde OK → Connessione stabilita

+ Non efficace

- Half-open connection

- Apro connessione e finisco ma al server riarriva richiesta di apertura e i dati trasmessi

\* 3-way

```

Host A Host B

----------------------------------->

SYNBit = 1, Seq = x

<-----------------------------------

SYNbit = 1, Seq = y, ACKbit = 1, ACKnum = x + 1

----------------------------------->

ACKbit = 1, ACKnum = y + 1

```

- Closing connection

```

Host A Host B

----------------------------------->

FINBit = 1, Seq = x

<-----------------------------------

ACKbit = 1, ACKnum = x + 1

<-----------------------------------

FINbit = 1, Seq = y

----------------------------------->

ACKbit = 1, ACKnum = y + 1

```

+ Principi di controllo di congestione

- Congestione → La rete non riesce a gestire troppe sorgenti che mandano troppi dati troppo velocemente

\* Cause

+ Delay grandi se il rate di arrivo si avvicina al rate della capacità della rete

- I pacchetti possono essere droppati a causa delle code dei router piene

- Il sender può andare in timeout prematuramente mandando dei duplicati

\* Costi

+ Più lavoro poiché i pacchetti anche se "buoni" vengono ritrasmessi

+ Questo fa sì che la banda dedicata ai pacchetti "buoni" diminuisca

+ Quando un pacchetto viene droppato le risorse usate per mandarlo sono sprecate

**+ Congestion control**

- Il sender aumenta la transmission rate (window size) per testare la banda finché non avviene una perdita

\* Additive increase → Ingrandisci cwnd (finestra di congestione) di 1 MSS (maximum segment size) ad ogni RTT

\* Multiplicative decrease → Dividi per due la cwnd quando rilevo una perdita

\* (Grafico a dente di sega)

- LastByteSent - LastByteACKed >= cwnd → Dinamica in base alla congestione della rete percepita

- Sending rate → Manda cwnd bytes, aspetta RTT per gli ACK, senda altri byte

\* Rate ~= cwnd / RTT bytes/s

- Slow start

\* Inizialmente la cwnd = 1 MSS

\* Raddoppia la cwnd ad ogni RTT (ogni ACK ricevuto)

\* All'inizio il rate è basso ma sale molto velocemente

- Detecting e reacting to loss

\* Loss indicata dal timeout

+ Cwnd = 1

+ Slow start fino a soglia e poi linearmente

\* Loss indicata da 3 ACK ripetuti (TCP RENO)

+ L'arrivo degli ACK vuol dire che la rete non è congestionata

+ Cwnd / 2 poi crescita lineare

\* TCP Tahoe in tutti e due i casi imposta la cwnd a 1

- Passaggio da slow start a crescita lineare

\* Questo passaggio viene effettuato quando si raggiunge la ssthresh

\* Sshthresh → Quando avviene una perdita questa variabile viene settata a metà della cwnd

+ Throughput

- Media → 3/4 \* cwnd/RTT bytes/s

- In funzione di segment loss probability (L) → 1.22 \* MSS / RTT \* sqrt(L)

+ Fairness → Se N host condividono banda R ognuno dovrebbe avere un rate medio di R/N

- La fairness è data anche dal throughput, gli host si "alternano" in base ai loro momenti di picco

- UDP → L'uso dell'UDP non rispetta la fairness, l'UDP ha sempre la precedenza

+ ECN (Explicit congestion notification)

- Due bit nell'header IP (ToS field) vengono segnati dai router della rete se viene rilevata congestione

- L'host ricevente nell'ACK di risposta comunica al sender la congestione tramite l'ECE bit

*Parte 4*

**- Network Layer**

\* Trasporta i segmenti dal sender al receiver

\* Dalla parte del sender incapsula i segmenti in datagrams

\* Dalla parte del ricevitore deploya i segmenti al trasnport layer

\* Il network layer è presente in ogni nodo in pratica e gli header di IP vengono esaminati ad ogni nodo

\* Funzioni

+ Forwarding → Trasporta i pacchetti dall'input del router all'appropriato output

- Data plane → Definisce come il datagram in arrivo sulla porta di input viene mandato a quella di output

+ Routing → Definisce la strada presa dal pacchetto dalla sorgente alla destinazione (routing algorithms)

- Control plane → Determina come il datagram è routato dall'host sorgente a quello destinazione

\* Traditional routing algorithms → Implementato in ogni router

\* Software defined networking → Implementato lato server

\* **Router**

dispositivo, che in una rete informatica si occupa di incanalare il traffico tra due o più dispositivi connessi alla stessa rete o nelle sottoreti.

+ Percorso del pacchetto: Layer fisico → Link layer → Network layer

- Network layer → Decentralized switching

\* Usando header troviamo porta output cercando in forwarding table

+ Destination-based → Tradizionale, forwarding basato solo sull'IP destinazione

- In forwarding table abbiamo dei range di IP (in) che corrispondono a interfacce di output

- Longest prefix matching → IP (in) corrisponda a più range, allora corrisponde al più lungo

+ Generalized → Forwarding basato su ogni set di campi dell'header

\* Queuing → Se i datagram arrivano più velocemente della velocità di forwarding avviene il queuing

+ Input

- Switch non regge se porte in input combinate → Delay and loss per input buffer overflow

- Head of the line blocking (HOL) → Datagram primi in fila bloccano successivi

\* Cause

+ Contendimento della output port fra due datagram in input

+ Buffer out già occupato

+ Output

- Switch di pacchetti più veloce dell'output rate → Delay and loss per output buffer overflow

+ In generale causa perdita pacchetti anche la "mancanza" o "scarsità" (?) di buffer

+ Buffering

- Average → Typical\_RTT (250 ms) \* Link\_capacity (C)

- Con N flussi → RTT \* C / sqrt(N)

+ Scheduling → Selezione del prossimo pacchetto da mandare

- Network neutrality → Non ci sono priorità "economiche"

- FIFO

\* Politica di drop

+ Tail → Droppa l'ultimo che arriva

+ Priority → Droppa in base alla priorità

+ Random

- Politiche

\* Priority → Manda il pacchetto con priorità maggiore

\* Round Robin → Controlla le code e manda pacchetto completo per ognuna, se possibile

\* Weighted Fair Queuing → Ogni coda riceve una quantità di risorse pesata ad ogni ciclo

\* Switching

+ Trasferisci i pacchetti dai buffer di input agli appropriati buffer di output

+ Switching rate → Velocità a cui questo viene fatto

- Misurato in multipli della line rate

- N input → Switching rate = N \* Line rate

+ Tipi

- Memory → Computer che utilizzavano memoria come buffer e CPU per calcolare "output port"

- Bus → Datagram da input buf a output via bus condiviso (velocità limitata da banda del bus)

- Interconnection network → Divido il datagram in celle di dimensione definita che switcho

\* **IP**

+ Datagram format

```

32 Bit

16 Bit 16 bit

Ver Head\_len Data\_type Length

Identifier Flags Fragment\_offset

Time\_to\_live Upper\_layer Header\_checksum

Source IP address

Destination IP address

Options

Payload

```

+ Fragmentation/Reassembly

- I vari router di Internet hanno una MTU (dimensione più grande frame trasferibile) differente

\* Per questo avvengono varie frammentazioni durante la route di un pacchetto

\* Il riassemblamento viene fatto solo all'host di arrivo

\* Nell'header IP ci sono informazioni dedicate a questo

- Esempio

```

4000 byte datagram → Length = 4000, ID = x, fragflag = 0, offset = 0

MTU = 1500 bytes:

Parte 1 → Length = 1500 (1480 di dati), ID = x, fragflag = 1, offset = 0

Parte 2 → Length = 1500, ID = x, fragflag = 1, offset = 185 (1480/8 → 2 Word)

Parte 3 → Length = 1040, ID = x, fragflag = 0, offset = 370

```

+ **IPv4 Addressing**

- IP address → ID 32-bit di interfaccia host/router (Interfaccia = Conn fra host/router e link fisico)

- Subnets → Dispositivi con stessa parte "Subnet" in IP, possono parlare senza uso router

\* IP address

+ Subnet → Bit di ordine alto

+ Host → Bit di ordine basso

- Classes interdomain routing (CIDR)

\* La porzione dell'IP address dedicata alla subnet ha lunghezza arbitraria

\* a.b.c.d/x → x = Numero di bit della parte subnet dell'indirizzo

- Dynamic host configuration protocol (**DHCP**) → Va in UDP

\* Ricevi l'indirizzo dal server dinamicamente → Plug and play

\* Funzionamento

```

Host → Boadcast: "DHCP Discover"

DHCP Server → Risposta: "DHCP offer"

Host → Richiedo IP: "DHCP Request" + Messaggio

DHCP Server → Mando IP: "DHCP ACK" + Messaggio

```

\* Altre funzioni

+ Indirizzo del primo router di "hop"

+ Nome e indirizzo del server DNS

+ Network mask (network vs host portion of IP)

\* Incapsulamento: DHCP → UDP → IP → 802.1 Ethernet e al contrario

- L'addressing è gerarchico

\* ICANN (Internet corp for assigned names and numbers)

+ Alloca gli indirizzi (ecco come gli ISP ottengono la loro parte di indirizzi)

+ Gestisce i DNS

+ Assegna nomi di dominio e risolve dispute

\* L'ISP assegna alle organizzazioni iscritte una parte dei suoi indirizzi

\* L'ISP farà sempre da "intermediario" per la connessione fra l'esterno e una sua organizzazione

- NAT (Network address translation)

\* Funzionamento

+ Tutti i pacchetti in arrivo da internet hanno stessa destinazione, IP del NAT, ma porte diverse

+ NAT "smista" pacchetti fra host connessi alla subnet del NAT (IP "virtuali") in base alla porta

\* Motivazioni

+ Ogni network locale usa solo un indirizzo IP

- ISP non deve assegnare range di IP ma IP unico

- Indirizzi della rete locale modificabili senza notificare l'esterno

- ISP intercambiabile senza cambiare IP locali

- Dispositivi locali non direttamente indirizzabili da esterno (sicurezza)

\* Implementazione

+ NAT deve

- Outgoing datagrams

\* Source IP e port → NAT IP e port

\* I server remoti risponderanno usando l'IP e la porta del NAT

- NAT translation table → ogni tupla di traduzione (Source IP-port/NAT IP-port)

- Incoming datagrams

\* NAT IP e port → Source IP e port

\* In questo modo i pacchetti sono riconducibili ad un destinatario locale

+ Porta → 16-bit number → 60000 connessioni con un singolo indirizzo

\* Il NAT rompe il sistema a livelli, dovrebbe processare solo fino al livello 3 ma arriva al 4 (TCP)

+ **IPv6**

indirizzo a 128 bit

- Motivazioni

\* 32-bit address (IPv4) presto finiranno

\* Il formato dell'header aumenta la velocità di forwarding e la qualità del servizio

+ Header → 40 bytes fissati

+ Frammentazione non consentita

- Formato

```

32 bit

Ver Priority Flow\_label

Payload\_len Next\_hdr Hop\_limit

Source\_addr (128 bit)

Dest\_addr (128 bit)

Data

Priority → Definisci una priorità dei datagram che scorrono

Flow label → Identifica i datagram in uno stesso "flow" (non definito)

Next header → Identifica il protocollo subito sopra per il segmento data

```

- Cambiamenti da IPv4

\* Checksum → Rimosso per diminuire il tempo di processing ad ogni hop

\* Opzioni → Ammesse ma fuori dall'header, indicate da "Next header" field

\* ICMPv6

+ Aggiunti tipi di messaggi, es "Packet too big"

+ Aggiunte le funzioni di controllo gruppi del multicast

- IPv4 → IPv6

\* Tunneling → IPv6 datagram trasportati come payload nei datagram IPv4 attraverso i router IPv4

```

A → B → C → D → E → F

IPv6 IPv6 IPv4 IPv4 IPv6 IPv6

A → B (IPv6): Flow = x, Src = A, Dest = F, Data

B → C (IPv6 in IPv4): Src = B, Dest = E, Data = (Flow = x, Src = A, Dest = F, Data)

C → D (IPv4)

D → E (IPv6 in IPv4): B → C

E → F (IPv6): A → B

```

\* **Generalized forwarding e SDN**

+ Ogni router contiene una flow table che è distribuita da un routing controller logicamente centralizzato

+ OpenFlow → communications protocol that gives access to the forwarding plane of a network switch or router

- Packet handling

\* Pattern → Matcha i campi dell'header con campi tabella

\* Azione per pacchetti matchati → Drop, forward, modify, send to controller

\* Priorità → Disambigua pacchetti "simili" in base alla priorità della flow table

\* Counter → Numero bytes e numero pacchetti

```

Switch\_port MAC\_src MAC\_dst Eth\_type VLAN\_ID IP\_src IP\_dst IP\_prot TCP\_sport TCP\_dport Action

\* \* \* \* \* \* 51.6.0.8 \* \* \* port6

I datagram IP destinati a 51.6.0.8 sono forwardati alla porta di output del router n.6

```

- Comportamenti

\* Router

+ Match → Il prefisso IP di destinazione più lungo

+ Action → Forwarda out

\* Switch

+ Match → Il MAC di destinazione

+ Action → Forwarda o flooda

\* Firewall

+ Match → IP e TCP/UDP ports

+ Action → Concedi o no

+ NAT

+ Match → IP e porta

+ Action → Riscrivi indirizzo e porta

*Libro*

*Capitolo 1 - Reti di calcolatori e Internet*

**Che cos’è Internet?**

**Gli “ingranaggi” di Internet**

Il termine rete di calcolatori comincia a essere datato, visto il grande numero di dispositivi non tradizionali collegati a Internet. In gergo, tutti questi dispositivi sono detti host (ospiti) o sistemi periferici (end system). I sistemi periferici sono connessi tra loro tramite una rete di collegamenti (communication link) e commutatori di pacchetti (packet switch). Collegamenti diversi possono trasmettere dati a velocità differenti, e tale velocità di trasmissione (transmission rate) viene misurata in bit/secondo (bps). Quando un sistema periferico vuole inviare dati a un altro sistema periferico, suddivide i dati in sottoparti e aggiunge un’intestazione a ciascuna di esse: l’insieme delle informazioni risultanti, nel gergo delle reti, viene chiamato pacchetto. I pacchetti sono inviati attraverso la rete alla destinazione, dove vengono riassemblati per ottenere i dati originari. Un commutatore di pacchetto prende un pacchetto che arriva da uno dei collegamenti in ingresso e lo ritrasmette su uno di quelli in uscita. Esistono commutatori di pacchetto di varia forma e natura, ma i due principali nell’odierna Internet sono i router e i commutatori a livello di collegamento (link-layer switch). I sistemi periferici accedono a Internet tramite i cosiddetti Internet service provider (ISP) che comprendono ISP residenziali quali le compagnie telefoniche. Due dei principali protocolli Internet sono il transmission control protocol (TCP), e l’Internet protocol (IP). I principali protocolli Internet sono noti con il nome collettivo di TCP/IP.

**Descrizione dei servizi**

Possiamo descrivere Internet anche come un’infrastruttura che fornisce servizi alle applicazioni, esse sono dette applicazioni distribuite, in quanto coinvolgono più sistemi periferici che si scambiano reciprocamente dati. I sistemi periferici collegati a Internet forniscono una interfaccia socket (socket interface), che specifica come un programma eseguito su un sistema periferico possa chiedere a Internet di recapitare dati a un programma eseguito su un altro sistema periferico

**Protocolli di rete**

Qualsiasi attività in Internet che coinvolga due o più entità remote in comunicazione viene governata da un protocollo.

Un protocollo definisce il formato e l’ordine dei messaggi scambiati tra due o più entità in comunicazione, così come le azioni intraprese in fase di trasmissione e/o di ricezione di un messaggio o di un altro evento. La maggior parte dei server da cui riceviamo i risultati delle ricerche, l’e-mail, le pagine Web e i video è collocata in grandi data center.

**Ai confini della rete**

I sistemi periferici vengono anche detti host in quanto ospitano ed eseguono programmi applicativi (browser, web server, gestione della posta elettronica…). Talvolta gli host vengono ulteriormente suddivisi in due categorie: client ( host che richiedono servizi) e server (erogano servizi).

**Le reti di accesso**

Reti di accesso (access network): rete che connette fisicamente un sistema al suo edge router (router di bordo), che è il primo router sul percorso dal sistema d’origine a un qualsiasi altro sistema di destinazione collocato al di fuori della stessa rete di accesso.

Accesso residenziale: DSL, via cavo, FTTH, dial-up e satellite

Oggigiorno i due accessi residenziali a larga banda più diffusi sono il digital subscriber line (DSL) e quello via cavo. Il modem DSL residenziale converte i dati digitali in toni ad alta frequenza per poterli trasmettere alla centrale locale sul cavo telefonico.

Mentre la DSL usa le infrastrutture già esistenti della compagnia telefonica locale, l’accesso a Internet via cavo utilizza le infrastrutture esistenti della televisione. Una tecnologia promettente che vanta velocità ancora maggiori è detta fiber to the home (FTTH). La distribuzione su fibra ottica dalla centrale locale all’abitazione può avvenire in modo diretto o condiviso con altre abitazioni e successivamente diviso in più fibre per ogni utente. Vi sono due architetture che eseguono questa suddivisione: le reti ottiche attive (AON, active optical networks; Ethernet commutate) e quelle passive (PON, passive optical networks; ogni abitazione ha un optical network terminator che si connette tramite un ad uno splitter comune tramite fibra).

Accesso aziendale (e residenziale): Ethernet e WiFi

Nelle aziende e nelle università, e sempre di più nelle abitazioni, per collegare i sistemi periferici al router di bordo si utilizza una rete locale (LAN, local area network). Esistono molti tipi di LAN (es. wireless), ma la tecnologia Ethernet è attualmente la più utilizzata. In una LAN wireless gli utenti trasmettono e ricevono pacchetti entro un raggio di poche decine di metri da e

verso un access point wireless.

Accesso wireless su scala geografica: 3G e LTE

A differenza del WiFi, l’utente può trovarsi a poche decine di chilometri dalla stazione base (3G, 4G, LTE)

**Mezzi trasmissivi**

I mezzi fisici ricadono in due categorie: i mezzi vincolati (guided media) e quelli non vincolati (unguided media). Nei primi, le onde vengono contenute in un mezzo fisico, quale un cavo in fibra ottica, un filo di rame o un cavo coassiale. Nei secondi, le onde si propagano nell’atmosfera e nello spazio esterno, come avviene nelle LAN wireless o nei canali digitali satellitari.

Doppino di rame intrecciato

I fili vengono intrecciati assieme per ridurre l’interferenza elettrica generata da altre coppie presenti nelle vicinanze.

Il doppino intrecciato non schermato (UTP, unshielded twisted pair) viene comunemente utilizzato per le reti all’interno di un edificio, cioè per le LAN.

Cavo coassiale

Il cavo coassiale può essere utilizzato come mezzo condiviso vincolato. Più nello specifico, più sistemi periferici possono essere connessi direttamente al cavo e tutti ricevono quanto inviato da altri sistemi periferici.

Fibra ottica

La fibra ottica è un mezzo sottile e flessibile che conduce impulsi di luce, ciascuno dei quali rappresenta un bit.

Canali radio terrestri

I canali radio trasportano segnali all’interno dello spettro elettromagnetico. Non richiede l’installazione fisica di cavi ed è in grado di attraversare le pareti. L’ambiente determina infatti la perdita di segnale lungo il percorso.

Canali radio satellitari

Un satellite per le comunicazioni collega due o più trasmettitori terrestri a microonde, riceve le trasmissioni su una banda di frequenza, rigenera il segnale utilizzando un ripetitore e trasmette segnali su un’altra frequenza. Nelle comunicazioni si usano due tipi di satellite: quelli geostazionari (GEO, geostationary earth orbit) e quelli a bassa quota (LEO, low-earth orbiting).

**Il nucleo della rete**

**Commutazione di pacchetto**

Le applicazioni distribuite scambiano messaggi che possono contenere qualsiasi cosa, la sorgente suddivide i messaggi lunghi in parti più piccole note come pacchetti. Tra la sorgente e la destinazione, questi pacchetti viaggiano attraverso collegamenti e commutatori di pacchetto (tipi principali: router e commutatori a livello di collegamento).

Se un sistema periferico o un commutatore inviano un pacchetto di L bit su un canale con velocità di R bps, il tempo di trasmissione risulta pari a L/R secondi

Trasmissione store-and-forward

La maggior parte dei commutatori di pacchetto utilizza la trasmissione store-andforward. Ciò significa che il commutatore deve ricevere l’intero pacchetto prima di poterne cominciare a trasmettere sul collegamento in uscita il primo bit.

Ritardi di accodamento e perdita di pacchetti

Ogni commutatore di pacchetto connette più collegamenti. Per ciascuno di questi, il commutatore mantiene un buffer di output (o coda di output) per conservare i pacchetti che sta per inviare su quel collegamento. I buffer di output rivestono un ruolo chiave nella commutazione di pacchetto. Oltre ai ritardi store-and forward, i pacchetti subiscono anche ritardi di accodamento. Un pacchetto in arrivo può trovare il buffer completamente riempito e avverrà una perdita di pacchetto (packet loss): verrà eliminato o il pacchetto in arrivo o uno di quelli che si trova già in coda.

Tabelle di inoltro e protocolli di instradamento

Ogni sistema periferico ha un indirizzo IP. Il router determina su quale collegamento inoltrare il pacchetto grazie alla tabella di inoltro (forwarding table), che mette in relazione gli indirizzi di destinazione (o loro parti) con i collegamenti in uscita.

Quando un pacchetto giunge a un router, questo esamina l’indirizzo e consulta la propria tabella per determinare il collegamento uscente appropriato. Il router, quindi, dirige il pacchetto verso quel collegamento di uscita. Internet ha parecchi protocolli di instradamento (routing protocol) che usa per impostare automaticamente le tabelle di inoltro.

**Commutazione di circuito**

Per spostare i dati in una rete di collegamenti e commutatori esistono due approcci fondamentali: la commutazione di circuito e la commutazione di pacchetto. Nelle reti a commutazione di circuito le risorse richieste lungo un percorso per consentire la comunicazione tra sistemi periferici sono riservate per l’intera durata della sessione di comunicazione. Nelle reti a commutazione di pacchetto, tali risorse non sono riservate: i messaggi di una sessione utilizzano le risorse quando necessario, e di conseguenza potrebbero dover attendere (ossia mettersi in coda) per accedere a un collegamento. Quando due host desiderano comunicare la rete stabilisce una connessione end-to-end dedicata a loro.

Multiplexing nelle reti a commutazione di circuito

Un circuito all’interno di un collegamento è implementato tramite multiplexing a divisione di frequenza (FDM, frequency-division multiplexing), dove il collegamento dedica una banda di frequenza a ciascuna connessione per la durata della connessione stessa, o multiplexing a divisione di tempo (TDM, time-division multiplexing, dove il tempo è diviso in frame ognuno dedicato ad una connessione. La larghezza della banda viene detta ampiezza di banda (bandwidth).

La commutazione di circuito è dispendiosa perché i circuiti dedicati sono inattivi durante i periodi di silenzio, di conseguenza vengono sprecate risorse.

Confronto tra commutazione di pacchetto e commutazione di circuito

La commutazione di pacchetto non è adatta ai servizi in tempo reale (come la telefonia e la videoconferenza) a causa dei suoi ritardi end-to-end variabili e non determinabili a priori. D’altro canto, offre una migliore condivisione della larghezza di banda rispetto alla commutazione di circuito, ma è anche più semplice, meno costosa da implementare, fornisce sostanzialmente le stesse prestazioni della commutazione di circuito, ma consente più del triplo degli utenti.

**Una rete di reti**

I sistemi periferici (PC, smartphone, posta elettronica…) si collegano a Internet tramite un ISP (internet service provider) di accesso che può fornire connettività attraverso una rete cablata o senza fili con svariate tecnologie quali DSL, cavo, FTTH, Wi-Fi e cellulare. È da notare che l’ISP non deve necessariamente essere una compagnia di telecomunicazioni o di televisione via cavo, ma potrebbe anche essere un’università o un’azienda. Per connettere i miliardi di utenti che costituiscono Internet bisogna interconnettere gli stessi ISP, ciò avviene creando una rete di reti. Nessun ISP è presente in ogni città del mondo, può esserci un ISP regionale al quale tutti gli ISP di accesso della regione si connettono. Ogni ISP regionale si connette all’ISP di primo livello (tier-1 ISP). In questa gerarchia ogni ISP di accesso paga l’ISP regionale a cui si connette, che a sua volta paga il suo ISP di primo livello. Per costruire una rete che sia più simile all’Internet odierna dobbiamo aggiungere alla struttura di rete 3 i PoP, il multi-homing, il peering e gli IXP. I PoP (point of presence) esistono in tutti i livelli della gerarchia tranne che in quello degli ISP di accesso. Un PoP è semplicemente un gruppo di router vicini tra loro nella rete del provider, tramite il quale gli ISP clienti possono connettersi al fornitore. Qualunque ISP, tranne quelli di primo livello, può scegliere la modalità multi-homing (o multi-home) che consiste nel connettersi a due o più ISP fornitori. Per ridurre tali costi, una coppia di ISP vicini e di

pari livello gerarchico può fare uso di peering,1 cioè connettere direttamente le loro reti in modo che tutto il traffico tra di esse passi attraverso una connessione diretta piuttosto che transitare da un intermediario. Utilizzando queste stesse connessioni, un’azienda terza, usando propri apparati e, di solito, un palazzo dedicato, può creare un IXP (Internet exchange point), un punto d’incontro dove più ISP possono fare peering tra di loro

**Ritardi, perdite e throughput nelle reti a commutazione di pacchetto**

**Panoramica del ritardo nelle reti a commutazione di pacchetto**

Un pacchetto parte da un host, passa attraverso una serie di router e conclude il viaggio in un altro host. A ogni tappa subisce vari tipi di ritardo a ciascun nodo (host o router) del tragitto, i principali ritardi sono il ritardo di elaborazione, il ritardo di accodamento, il ritardo di trasmissione e il ritardo di propagazione, che complessivamente formano il ritardo totale di nodo (nodal delay).

Ritardo di elaborazione

Il tempo richiesto per esaminare l’intestazione del pacchetto e per determinare dove dirigerlo fa parte del ritardo di elaborazione (processing delay) che include anche il tempo richiesto per controllare errori.

Ritardo di accodamento

Una volta in coda, il pacchetto subisce un ritardo di accodamento (queuing delay) mentre attende la trasmissione sul collegamento. La lunghezza di tale ritardo per uno specifico pacchetto dipenderà dal numero di pacchetti precedentemente arrivati.

Ritardo di trasmissione

Sia L la lunghezza del pacchetto, in bit, e R bps la velocità di trasmissione del collegamento dal router A al router B il ritardo di trasmissione (transmission delay) risulta essere L/R. Questo è il tempo richiesto per trasmettere tutti i bit del pacchetto sul collegamento. Solitamente sono microsecondi o millisecondi.

Ritardo di propagazione

Una volta immesso sul collegamento, un bit deve propagarsi fino al router B. Il tempo impiegato è il ritardo di propagazione (propagation delay). Il ritardo di propagazione è dato da d/v, dove d è la distanza tra i due router, mentre v è la velocità di propagazione nel collegamento.

*a*: velocità media di arrivo dei pacchetti nella coda, in pacchetti al secondo

R: la velocità di trasmissione, in bit al secondo

L: lunghezza pacchetto, in bit

d: ritardo (delay)

- ritardo totale di un nodo: dnodo = delab + dacc + dtrasm + dprop

- ritardo di trasmissione: L/R

- velocità media di arrivo dei bit in coda: L*a* bit/s

- intensità di traffico: L*a*/R se >1 male, la coda cresce

**Ritardo end-to-end**

Ritardo dalla sorgente alla destinazione (end-to-end delay) è pari a:

dend-to-end = (N+1) (delaborazione + dtrasmissione + dpropagazione) N: router tra l’host sorgente e quello di destinazione

Traceroute

Per ottenere una misura efficiente dei ritardi in una rete di calcolatori possiamo fare uso del programma diagnostico Traceroute. Si tratta di un semplice programma eseguibile su qualsiasi host di Internet che invia un certo numero di pacchetti speciali verso una destinazione specificata. Quando un router riceve uno di questi pacchetti speciali, invia un breve messaggio che torna all’origine, esso contiene il nome e l’indirizzo del router. Serve per calcolare il tempo intercorso tra l’invio di un pacchetto e la ricezione del corrispondente messaggio di ritorno.

Sistemi periferici, applicazioni e altri ritardi

Il tempo per riempire un pacchetto è detto ritardo di pacchettizzazione, presente nelle applicazioni di telefonia VoIP dove il mittente deve prima di tutto riempire il pacchetto con conversazione digitalizzata e codificata, e poi inviarlo su Internet.

**Throughput nelle reti di calcolatori**

Un’altra misura critica delle prestazioni in una rete di calcolatori è il throughput end-to-end. Considerate il trasferimento di un file voluminoso (es. un grosso videoclip) da A a B, attraverso la rete da un peer a un altro in un sistema di condivisione di file P2P. Il throughput istantaneo in ogni istante di tempo è la velocità (in bps) alla quale B sta ricevendo il file, se il file consiste di F bit e il trasferimento richiede T secondi affinché B riceva tutti gli F bit allora il throughput medio del trasferimento del file è di F/T bps.

**Livelli dei protocolli e loro modelli di servizio**

**Architettura a livelli**

Un’architettura a livelli consente di discutere una parte specifica e ben definita di un sistema articolato e complesso.

Stratificazione dei protocolli

Per dare struttura alla loro progettazione i progettisti organizzano i protocolli, l’hardware e software che implementano in livelli o strati (layer). Ciascun protocollo o funzione appartiene a uno dei livelli, questo è il modello di servizio (service model) di un livello.

Livello di applicazione

Il livello di applicazione (application layer) è la sede delle applicazioni di rete e dei relativi protocolli. Per quanto riguarda Internet, tale livello include molti protocolli, quali http, SMTP, FTP. Un’applicazione in un sistema periferico, tramite il protocollo, scambia pacchetti di informazioni (detti messaggi) con l’applicazione in un altro sistema periferico.

Livello di trasporto

Il livello di trasporto (transport layer) di Internet trasferisce i messaggi del livello di applicazione tra punti periferici gestiti dalle applicazioni. In Internet troviamo due protocolli di trasporto:

TCP (servizio orientato alla connessione che include la consegna garantita dei messaggi a livello di applicazione alla destinazione e il controllo di flusso) e UDP (servizio non orientato alla connessione senza affidabilità, né controllo di flusso e della congestione). Chiameremo segmenti i pacchetti a livello di trasporto.

Livello di rete

Il livello di rete (network layer) di Internet si occupa di trasferire i pacchetti a livello di rete, detti datagrammi, da un host a un altro. Il livello di rete di Internet comprende il protocollo IP.

Livello di collegamento

Il livello di rete di Internet instrada un datagramma attraverso una serie di router tra la sorgente e la destinazione. Chiameremo frame i pacchetti a livello di collegamento. Ad ogni nodo, il livello di rete passa il datagramma al livello sottostante, che lo trasporta al nodo successivo. Esempi di livello di collegamento: Ethernet, Wi-Fi e il protocollo di accesso alla rete DOCSIS.

Livello fisico

Mentre il compito del livello di collegamento è spostare interi frame da un elemento della rete a quello adiacente, il ruolo del livello fisico (physical layer) è trasferire i singoli bit del frame da un nodo a quello successivo. i bit sono trasferiti lungo il collegamento secondo differenti modalità. Ethernet presenta vari protocolli a livello fisico: uno per il doppino intrecciato, uno per il cavo coassiale, uno per la fibra ottica ecc…

**Incapsulamento**

Durante l’invio di un messaggio, l’informazione scende lungo lo stack del mittente. L’incapsulamento è l’operazione di inserimento di un payload di un certo livello N nell’instestazione di livello N-1. Il risultato è un pacchetto di livello N-1, che diventa quindi il carico utile per il protocollo di rete di livello N-2, e così via. Quindi, a ciascun livello, il pacchetto ha due tipi di campi: quello di intestazione e quello di payload.

**Reti sotto attacco**

Malware installati sugli host tramite Internet

I malware possono cancellare i nostri file, installare spyware che raccolgono le nostre informazioni private (come il nostro numero di carta di credito o le password, e poi inviare tutto ciò ai malintenzionati). Il nostro host potrebbe essere reclutato in un botnet (rete di migliaia di dispositivi compromessi usati per spedire e-mail spam o per sferrare attacchi DoS). Molti malware sono auto-replicanti e possono diffondersi sotto forma di virus (che richiedono una qualche forma

di interazione con l’utente per infettarne il dispositivo, es. e-mail) o worm (che possono entrare in un dispositivo senza alcuna interazione esplicita con l’utente, es. eseguire un’applicazione di rete vulnerabile dove può essere mandato un worm).

Attacchi ai server e all’infrastruttura di rete

DoS (denial-of-service) sono attacchi di negazione del servizio che rendono inutilizzabile dagli utenti una rete, un host o un’altra parte di infrastruttura. Si dividono in tre categorie:

- Attacchi alla vulnerabilità dei sistemi: invio di pochi messaggi ben costruiti a un’applicazione vulnerabile.

- Bandwidth flooding (inondazione di banda): L’attaccante invia un “diluvio” di pacchetti all’host bersaglio, così tanti che il suo collegamento di accesso viene ostruito, impedendo ai pacchetti legittimi di raggiungere il server.

- Connection flooding (inondazione di connessioni): L’attaccante stabilisce un gran numero di connessioni TCP completamente o solo parzialmente aperte all’host bersaglio. L’host può così “ingorgarsi” con queste connessioni senza poter accettare le connessioni valide.

Mascheramento

La capacità di immettere pacchetti in Internet con un indirizzo sorgente falso è nota come IP spoofing, ed è uno dei molti modi attraverso i quali un utente può spacciarsi per un altro. Per risolvere tale problema abbiamo bisogno di autenticare il punto terminale della comunicazione (end-point authentication), cioè di un meccanismo che ci permetta di determinare con certezza se il messaggio ha avuto origine da dove supponiamo l’ha avuta.

*Capitolo 2 -Livello di applicazione*

**Princìpi delle applicazioni di rete**

Nelle applicazioni web esistono due programmi diversi che comunicano tra di loro: il browser, che viene eseguito dall’host dell’utente (desktop, laptop, PDA, telefono cellulare e così via) e il web server che si trova nell’host che viene di solito chiamato anch’esso web server.

**Architetture delle applicazioni di rete**

Nell’architettura client-server vi è un host sempre attivo, chiamato server, che risponde alle richieste di servizio di molti altri host, detti client. I data center ospitando molti host creano un potente server virtuale. In un’architettura P2P l’infrastruttura di server in data center è minima o del tutto assente; si sfrutta, invece, la comunicazione diretta tra coppie arbitrarie di host, chiamati peer (ossia pari), collegati in modo intermittente.

**Processi comunicanti**

non si parla di programmi. I processi su due sistemi terminali comunicano scambiandosi messaggi attraverso la rete-

Processi client e server

Nel contesto di una sessione di comunicazione tra una coppia di processi quello che avvia la comunicazione (cioè, contatta l’altro processo all’inizio della sessione) è indicato come client mentre quello che attende di essere contattato per iniziare la sessione è detto server.

L’interfaccia tra il processo e la rete

Un processo invia messaggi nella rete e riceve messaggi dalla rete attraverso un’interfaccia software detta socket. API (application programming interface) ovvero Interfaccia di programmazione delle applicazioni, sono set di definizioni e protocolli con i quali vengono realizzati e integrati software applicativi. Consentono ai tuoi prodotti o servizi di comunicare con altri prodotti o servizi senza sapere come vengono implementati. Sono tra l’applicazione e la rete, il socket rappresenta l’interfaccia di programmazione con cui le applicazioni di rete vengono costruite.

Indirizzamento

In Internet, gli host vengono identificati attraverso i loro indirizzi IP

**Servizi di trasporto disponibili per le applicazioni**

Trasferimento dati affidabili

Se un protocollo fornisce questo tipo di servizio di consegna garantita dei dati, si dice che fornisce un trasferimento dati affidabile (reliable data transfer). Se non affidabile è accettabile per le applicazioni che tollerano le perdite (loss-to lerant).

Throughput

Throughput disponibile è il tasso al quale il processo mittente può inviare i bit al processo ricevente. Con il throughput disponibile garantito l’applicazione può sempre richiedere un throughput garantito di r bps. Le applicazioni che hanno requisiti di throughput vengono dette applicazioni sensibili alla banda (bandwidth-sensible). Applicazioni elastiche possono far uso di tanto o di poco throughput a seconda di quanto ce ne sia disponibile.

Temporizzazione

Un protocollo a livello di trasporto può anche fornire garanzie di temporizzazione (timing) che, come quelle per il throughput, possono assumere varie forme.

Sicurezza

Infine, un protocollo a livello di trasporto può fornire a un’applicazione uno o più servizi di sicurezza.

**Servizi di trasporto offerti da Internet**

Internet (come ogni rete TCP/IP) mette a disposizione delle applicazioni due protocolli di trasporto: UDP e TCP.

Servizi di TCP

TCP prevede un servizio orientato alla connessione e il trasporto affidabile dei dati.

• Servizio orientato alla connessione. TCP fa in modo che client e server si scambino informazioni di controllo a livello di trasporto prima che i messaggi a livello di applicazione comincino a fluire. Questa procedura, detta di handshaking, mette in allerta client e server, preparandoli alla partenza dei pacchetti. Dopo la fase di handshaking, si dice che esiste una connessione TCP tra le socket dei due processi. Tale connessione è full-duplex, nel senso che i due processi possono scambiarsi contemporaneamente messaggi sulla connessione. L’applicazione deve chiudere la connessione quando termina di inviare messaggi.

• Servizio di trasferimento affidabile. I processi comunicanti possono contare su TCP per trasportare i dati senza errori e nel giusto ordine.

TCP include anche un meccanismo di controllo della congestione

Servizi di UDP

UDP è un protocollo di trasporto leggero e senza fronzoli, dotato di un modello di servizio minimalista. UDP è senza connessione, non necessita quindi di handshaking, e fornisce un servizio di trasferimento dati non affidabile. UDP non include un meccanismo di controllo della congestione.

**Protocolli a livello di applicazione**

Un protocollo a livello di applicazione definisce come i processi di un’applicazione, in esecuzione su sistemi periferici diversi, si scambiano i messaggi. (es HTTP e smtp)

**Web e HTTP**

**Panoramica di HTTP**

HTTP (hypertext transfer protocol), protocollo a livello di applicazione del Web. Il protocollo definisce sia la struttura dei messaggi sia la modalità con cui client e server si scambiano i messaggi. Prima di affrontare in dettaglio HTTP soffermiamoci brevemente sulla terminologia web.

Pagina web (web page) è costituita da file indirizzabili tramite URL linkati nel file HTML principale.

Browser web (come firefox, è un client) e server web.

Dato che i server HTTP non mantengono informazioni sui client, HTTP è classificato come protocollo senza memoria di stato (stateless protocol)

**Connessioni persistenti e non persistenti**

In molte applicazioni per Internet, client e server comunicano per un lungo periodo; quando tale interazione client-server ha luogo su TCP, gli sviluppatori possono utilizzare:

HTTP con connessioni non persistenti

Ogni connessione TCP viene chiusa dopo l’invio dell’oggetto da parte del server (molte connessioni connessioni TCP per una singola pagina web).

Round-trip time (RTT): rappresenta il tempo impiegato da un piccolo pacchetto per viaggiare dal client al server e poi tornare al client. RTT include i ritardi di propagazione,di accodamento nei router e nei commutatori intermedi nonché di elaborazione del pacchetto. Quando un utente fa click con il mouse su un collegamento ipertestuale avviene un handshake a tre vie dove il client invia un piccolo segmento TCP al server, quest’ultimo manda una conferma per mezzo di un piccolo segmento TCP. Infine, il client dà anch’esso una conferma di ritorno al server (1° + 2° fase richiedono 1 RTT). Successivamente client fa richiesta HTTP al server che gli inoltrerà il file sulla connessione TCP (3° fase 1 RTT).

Limiti: per ogni oggetto richiesto occorre stabilire e mantenere una nuova connessione dove si devono allocare buffer e mantenere variabili TCP sia nel client sia nel server (grave onere sul web server). Inoltre, ciascun oggetto subisce un ritardo di consegna di due RTT, uno per stabilire la connessione TCP e uno per richiedere e ricevere un oggetto.

HTTP con connessioni persistenti

Il server lascia la connessione TCP aperta dopo l’invio di una risposta, per cui le richieste e le risposte successive tra gli stessi client e server possono essere trasmesse sulla stessa connessione. Il server può inviare un’intera pagina web su una sola connessione TCP permanente e può anche spedire allo stesso client più pagine web. Queste richieste di oggetti possono essere effettuate una di seguito all’altra senza aspettare le risposte delle richieste pendenti (pipelining). In generale, il server HTTP chiude la connessione quando essa rimane inattiva per un dato lasso di tempo.

**Formato dei messaggi HTTP**

Messaggio di richiesta HTTP

La prima riga è detta riga di richiesta (request line) e quelle successive righe di intestazione (header lines). La riga di richiesta presenta tre campi: il campo metodo, il campo URL e il campo versione di HTTP. Il campo metodo può assumere diversi valori, tra cui GET, POST, HEAD, PUT e DELETE. La maggioranza dei messaggi di richiesta HTTP usa il metodo GET, adottato quando il browser richiede un oggetto identificato dal campo URL.

Messaggio di risposta HTTP

una riga di stato iniziale (contiene la versione del protocollo, un codice di stato e un corrispettivo messaggio di stato),

sei righe di intestazione (Connection: close per comunicare al client che ha intenzione di chiudere la connessione TCP dopo l’invio del messaggio. Date: indica l’ora e la data di creazione e invio, da parte del server, della risposta HTTP. Server: indica che il messaggio è stato generato da un web server Apache. Last-Modified: indica l’istante e la data il cui l’oggettoè stato creato o modificato per l’ultima volta. Content-Length: contiene il numero di byte dell’oggetto inviato. La riga Content-Type: indica che l’oggetto nel corpo è testo HTML)

e il corpo (contiene l’oggetto richiesto).

**Interazione utente-server: i cookie**

I web server possono autenticare gli utenti sia per limitare l’accesso da parte di questi ultimi, sia per fornire contenuti in funzione della loro identità. A questo scopo, HTTP adotta i cookie che consentono ai server di tener traccia degli utenti e delle loro azioni sul sito.

**Web caching**

Una web cache, nota anche come proxy server, è un’entità di rete che soddisfa richieste HTTP al posto del web server effettivo. Ha una propria memoria cache in cui conserva copie di oggetti recentemente richiesti; quando riceve richieste da un browser egli invia risposte agisce da server, quando invia richieste e riceve risposte da un server di origine funziona da client.

Può ridurre in modo sostanziale i tempi di risposta alle richieste dei client ed il traffico sul collegamento di accesso a Internet.

GET condizionale

Il web caching introduce un nuovo problema: la copia di un oggetto che risiede in cache potrebbe essere scaduta. Il GET condizionale serve a verificare se i suoi oggetti sono aggiornati.

**Posta elettronica in Internet**

Mezzo di comunicazione asincrono: le persone inviano e leggono i messaggi nel momento per loro più opportuno, senza doversi coordinare con altri utenti. Tre componenti principali: gli user agent (o agenti utente), i server di posta (o mail server) e il protocollo SMTP (simple mail transfer protocol). I mail server costituiscono la parte centrale dell’infrastruttura del servizio di posta elettronica. Ciascun destinatario ha una casella di posta (mailbox) collocata in un mail server. Se un server non può consegnare la posta la trattiene in una coda di messaggi e cerca di trasferirla in un secondo momento.

**SMTP**

Principale protocollo a livello di applicazione per la posta elettronica su Internet. Fa uso del servizio di trasferimento dati affidabile proprio di TCP per trasferire la mail dal server del mittente a quello del destinatario. Presenta un lato client, in esecuzione sul mail server del mittente e un lato server, in esecuzione sul server del destinatario. Una volta stabilita la connessione il server e il client effettuano una qualche forma di handshaking a livello applicativo. Non usa mail server intermedi per inviare la posta. ASCII a 7 bit e fa uso di connessioni persistenti.

**Confronto con HTTP**

HTTP trasferisce file (spesso chiamati oggetti) da un web server a un web client (solitamente un browser), mentre SMTP trasferisce file da un mail server a un altro. Durante il trasferimento, sia HTTP persistente sia SMTP utilizzano connessioni persistenti e quindi presentano caratteristiche comuni. Esistono però sostanziali differenze. Innanzitutto, HTTP è principalmente un protocollo pull: qualcuno carica informazioni su un web server e gli utenti usano HTTP per attirarle a sé (pull) dal server. In particolare, la connessione TCP viene iniziata dalla macchina che vuole ricevere il file. Al contrario, SMTP è sostanzialmente un protocollo push: il mail server di invio spinge (push) i file al mail server in ricezione. In particolare, la connessione TCP viene iniziata dall’host che vuole spedire il file.

**Formati dei messaggi di posta**

L’RFC 5322 specifica il formato esatto delle righe di intestazione della posta e la loro interpretazione. Ogni intestazione deve avere una riga From:, una riga To: e una riga Subject: opzionale; successivamente troviamo il corpo del messaggio.

**Protocolli di accesso alla posta**

SMTP trasferisce la posta dal PC del mittente al suo server di posta, sempre SMTP poi lo trasferirà al server di posta del destinatario, infine l’accesso viene completato introducendo uno speciale protocollo di accesso alla posta tra POP3 (post office protocol – versione 3) , IMAP (Internet mail access protocol) e HTTP, che trasferisce i messaggi dal mail server del destinatario al suo PC locale.

POP3

Il protocollo entra in azione quando lo user agent (il client) apre una connessione TCP verso il mail server (il server) sulla porta 110. Quando la connessione TCP è stabilita, POP3 procede in tre fasi: autorizzazione (lo user agent invia nome utente e password in chiaro per autenticare l’utente), transazione (lo user agent recupera i messaggi e può marcarli) e aggiornamento (il server di posta rimuove i messaggi che sono stati marcati per la cancellazione). Non si possono associare messaggi a cartelle. Modalità POP3:

scarica e cancella: lo user agent recupera e cancella ogni messaggio dal server. Problemi se si vuole accedere da più dispositivi.

scarica e mantieni: lo user agent lascia i messaggi sul server di posta dopo averli scaricati. Si possono rileggere i messaggi da diverse macchine, in differenti momenti.

IMAP

Maggiori potenzialità rispetto a POP3 ed è quindi assai più complesso. Un server IMAP associa ogni messaggio arrivato al server a una cartella (default INBOX). Server IMAP conservano informazioni di stato sull’utente da una sessione all’altra come i nomi delle cartelle e l’associazione tra i messaggi e le cartelle, inoltre permettono agli user agent di ottenere singole parti dei messaggi.

Posta basata sul Web

È oggi in costante crescita il numero di utenti che inviano posta e accedono alle proprie email tramite un browser web. Lo user agent è un semplice browser web e l’utente comunica con la propria casella remota via HTTP. SMTP viene ancora utilizzato per comunicare tra server di posta, mentre l’invio e la ricezione avvengono con il protocollo HTTP.

**DNS: il servizio di directory di Internet**

Gli host Internet possono essere identificati in vari modi: tramite hostname (appropriati per l’uomo ma non per internet) e tramite indirizzi IP (facilmente elaborabili dai server).

**Servizi forniti da DNS**

Il domain name system (DNS) di Internet traduce i nomi degli host nei loro indirizzi IP ed è un database distribuito implementato in una gerarchia di DNS server e un protocollo a livello di applicazione che consente agli host di interrogare il database. I DNS server sono generalmente macchine UNIX che eseguono un software chiamato BIND (Berkeley Internet name domain). Il protocollo DNS utilizza UDP e la porta 53. Oltre alla traduzione degli hostname in indirizzi IP, DNS mette a disposizione altri importanti servizi:

• Host aliasing: Un host dal nome complicato può avere uno o più sinonimi (alias).

• Mail server aliasing: Un’applicazione di posta può invocare il DNS per ottenere il nome canonico di un sinonimo fornito, così come l’indirizzo IP dell’host.

• Distribuzione del carico di rete (load distribution): Il DNS utilizzato per distribuire il carico tra server replicati, i siti con molto traffico vengono replicati su più server, ognuno eseguito su un host diverso con un indirizzo IP differente. Nel caso di web server replicati, va dunque associato a ogni hostname canonico un insieme di indirizzi IP.

**Panoramica del funzionamento di DNS**

Funzionamento del servizio di traduzione da hostname a indirizzo IP: una certa applicazione in esecuzione sull’host di un utente che ha necessità di tradurre un hostname in un indirizzo IP invocherà il lato client del DNS, specificando l’hostname da tradurre. Il DNS sull’host prende poi il controllo, inviando un messaggio di richiesta (query) sulla rete. Tutte le query DNS e i messaggi di risposta vengono inviati all’interno di datagrammi UDP diretti alla porta 53. Il client DNS sull’host dell’utente riceve un messaggio di risposta contenente la corrispondenza desiderata, che viene poi passata all’applicazione che ne ha fatto richiesta.

Questo servizio è composto da DNS server organizzati in maniera gerarchica e distribuiti per il mondo, e da un protocollo a livello di applicazione, che specifica la comunicazione tra DNS server e host richiedenti. Il DNS non è centralizzato perché comporterebbe problemi tra i quali:

• Un solo punto di fallimento, se il DNS server si guasta, ne soffre l’intera Internet.

• Volume di traffico, un singolo DNS server dovrebbe gestire tutte le richieste (per tutte le richieste HTTP e i messaggi di posta elettronica generati da centinaia di milioni di host).

• Database centralizzato distante, non può essere vicino a tutti i client.

• Manutenzione, a dovrebbe essere aggiornato frequentemente per tener conto di ogni nuovo host

• non è scalabile, cioè non è in grado di adattarsi alla crescita esponenziale della rete

Un database distribuito e gerarchico

Nessun DNS server ha le corrispondenze per tutti gli host in Internet, che sono invece distribuite tra tutti i DNS server. Esistono tre classi di DNS server che compongono la gerarchia di DNS server:

• Root server: Forniscono gli indirizzi IP dei server TLD.

• Top-level domain (TLD) server: si occupano dei domini di primo livello (es. .com, .it, ...) e forniscono gli indirizzi IP dei server autoritativi.

• DNS server autoritativi: ogni organizzazione dotata di host pubblicamente accessibili tramite Internet deve fornire record DNS pubblicamente accessibili che associno i nomi di tali host a indirizzi IP. Questi indirizzi saranno contenuti nel DNS server autoritative dell’organizzazione.

Un altro tipo di DNS è il DNS server locale (o default name server), che non appartiene strettamente alla gerarchia di server, ma che è comunque centrale nell’architettura DNS. Ciascun ISP, come un’università, un dipartimento universitario, un’azienda o un ISP residenziale, ha un DNS server locale. Quando un host si connette a un ISP, quest’ultimo gli fornisce un indirizzo IP tratto da uno o più dei suoi DNS server locali, generalmente tramite DHCP.

DNS Caching

Il DNS sfrutta in modo estensivo il caching per migliorare le prestazioni di ritardo e per ridurre il numero di messaggi DNS che “rimbalzano” su Internet. Il DNS server che riceve una risposta DNS può mettere in cache le informazioni contenute, che vengono invalidate dopo un periodo di tempo fissato (in genere di 2 giorni).

**Record e messaggi DNS**

I server che implementano il database distribuito di DNS memorizzano i cosiddetti record di risorsa (RR, resource record) che contiene i seguenti campi: (Name, Value, Type, TTL)

TTL è il time to live, ossia il tempo residuo di vita di un record e determina quando una risorsa vada rimossa dalla cache. Il significato di Name e Value dipende da Type:

• Se Type=A, allora Name è il nome dell’host e Value è il suo indirizzo IP.

• Se Type=NS, allora Name è un dominio (quale foo.com) e Value è l’hostname del DNS server autoritativo che sa come ottenere gli indirizzi IP degli host nel dominio.

• Se Type=CNAME, allora Value rappresenta il nome canonico dell’host per il sinonimo Name.

• Se Type=MX, allora Value è il nome canonico di un mail server che ha il sinonimo Name.

Messaggi DNS

La sezione di intestazione è contenuta nei primi 12 byte (Identificazione; flag; Numero di domande; Numero di RR di risposta, autoritativi e addizionali). Un ulteriore bit, chiamato di richiesta di ricorsione (recursion-desired flag), viene impostato quando un client (sia esso un host o un DNS server) desidera che il DNS server effettui ricorsione quando non dispone del record.

La sezione delle domande contiene informazioni sulle richieste che stanno per essere effettuate.

La sezione delle risposte In una risposta proveniente da DNS server contiene i record di risorsa relativi al nome originariamente richiesto.

La sezione autoritativa contiene i record di altri server autoritativi.

La sezione aggiuntiva racchiude altri record utili.

Per inviare un messaggio di richiesta DNS a un DNS server, direttamente dalla macchina su cui state lavorando si può usare il programma nslookup.

Inserimento di record nel database DNS

Per inserire i record nel database bisogna innanzitutto registrare il nome di dominio presso un registrar (un’azienda che verifica l’unicità del nome di dominio lo inserisce nel database DNS) e dovete fornirgli anche i nomi e gli indirizzi IP dei vostri DNS server autoritativi primario e secondario.

**Distribuzione di file P2P**

L’architettura peer-to-peer P2P è praticamente indipendente da infrastruttura di server sempre attivi. Ci sono, invece, coppie di host connessi in modo intermittente, chiamati peer, che comunicano direttamente l’uno con l’altro. I peer non appartengono ai fornitori dei servizi, ma sono computer controllati dagli utenti.

Scalabilità dell’architettura P2P

Il tempo di distribuzione è il tempo richiesto perché tutti gli N peer ottengano una copia del file. Nell’architettura P2P il tempo di distribuzione minimo è sempre minore di quello dell’architettura client-server. Le applicazioni con architettura P2P possono essere scalabili e la scalabilità è una diretta conseguenza del fatto che i peer re-distribuiscono i bit oltre che a scaricarli.

BitTorrent

E’ un diffuso protocollo P2P per la distribuzione di file. l’insieme di tutti i peer che partecipano alla distribuzione di un particolare file è chiamato torrent (torrente). I peer in un torrent scaricano chunk (parti) del file di uguale dimensione, con una dimensione tipica di 256 kbyte. Una volta che un peer ha acquisito l’intero file, può lasciare il torrent o rimanere nel torrent e continuare a inviare chunk agli altri peer.

Ciascun torrent ha un nodo di infrastruttura chiamato tracker, quando un peer entra a far parte di un torrent, si registra presso il tracker e periodicamente lo informa che è ancora nel torrent. In questo modo, il tracker tiene traccia dei peer che stanno partecipando al torrent. Chiamiamo i peer con i quali un utente riesce a stabilire una connessione TCP “peer vicini” (neighboring peer). Nel decidere quali chunk richiedere, l’utente adotta la tecnica del rarest first (il più raro per primo). L’idea è determinare quali sono i chunk più rari tra quelli che ancora le mancano, cioè i chunk con il minor numero di copie ripetute tra i suoi vicini, e richiederli per primi. L’utente poi contraccambia inviando chunk. I peer unchoked sono quelli che passano i bit all’utente alla velocità più elevata. L’utente poi contraccambia inviando chunk del file a quegli stessi quattro. Aspetto importante è che ogni 30 secondi sceglie casualmente un vicino in più e gli invia dei chunk. Optimistically unchoked è un peer con cui sta comunicando l’utente, tutti gli altri oltre questi 5 sono detti chocked. Tabelle hash distribuite (DHT), semplici database i cui record sono distribuiti tra i peer di un sistema P2P.

**Streaming video e reti per la distribuzione di contenuti**

**Video su Internet**

Forse la caratteristica più saliente del video è l’elevato tasso con cui è necessario inviare i bit sulla rete (bit rate). La compressione può essere usata anche per creare versioni multiple dello stesso video, a livelli di qualità diversi.

**Reti per la distribuzione di contenuti**

Costruire un unico enorme data center per lo streaming ha 3 grandi problematiche:

- se il client è lontano dal data center, i pacchetti dal server al client devono percorrere un lungo cammino passando per molti ISP. Se un collegamento ha throughput minore del tasso di consumo del video, anche il throughput totale end-to-end lo sarà, causando all’utente continui e fastidiosi fermi immagine.

- un video molto popolare verrà inviato molte volte sullo stesso collegamento, non solo sprecando banda, ma costringendo anche l’azienda a pagare tutte le volte il suo ISP (collegato al data center) per inviare gli stessi dati.

- un singolo data center rappresenta un singolo punto di rottura.

Per superare queste problematiche, quasi tutte le maggiori aziende di video streaming usano le CDN (content distribution networks, reti di distribuzioni di contenuti). Una CDN gestisce server distribuiti in molti posti diversi, memorizza copie dei video e di altri contenuti web nei server e cerca di dirigere le richieste degli utenti al punto della CDN in grado di offrire il servizio migliore. La CDN può essere una CDN privata, cioè proprietaria del fornitore di contenuti, come la CDN di Google che distribuisce contenuti di YouTube. Alternativamente può essere la CDN di terze parti. Politiche di dislocazione dei server che i CDN adottano:

- Enter deep: L’obiettivo è quello di essere vicini agli utenti finali in modo da migliorare il ritardo percepito dall’utente e il throughput, diminuendo il numero di collegamenti e router tra l’utente finale e il cluster CDN da cui riceve i contenuti. Tale approccio, altamente distribuito, pone però grandi sfide nella manutenzione e gestione dei cluster.

- Bring home: Il secondo approccio, seguito da Limelight e molti altri fornitori di CDN, è quello di portarsi in casa l’ISP costruendo grandi cluster in pochi (decine, per esempio) punti chiave e interconnetterli usando una rete privata ad alta velocità.

Una volta posizionati i cluster, la CDN replica in contenuti su di essi. I video raramente richiesti o che sono popolari solo in alcuni paesi non vengono copiati su tutti i cluster.

Come funziona la CDN

Quando un browser nell’host di un utente chiede il recupero di uno specifico video identificato da un URL, la CDN deve intercettare la richiesta in modo da poter determinare il cluster di server più appropriato per quel cliente a quell’istante e dirigere la richiesta del client a uno dei server di quel cluster.

Strategie di selezione dei cluster

E’ il meccanismo per dirigere dinamicamente i client a un cluster di server o a un data center della CDN. Una semplice strategia consiste nell’assegnare a un client il cluster geograficamente più vicino usando un database commerciale di geolocalizzazione, gli indirizzi IP degli LDNS (DNS locali) verranno associati a un luogo geografico.

Un secondo approccio determina il cluster migliore per un client basandosi sulle condizioni di traffico correnti, effettuando misure in tempo reale delle prestazioni di ritardo e perdita tra i loro cluster e i loro client; testando tramite messaggi ping o interrogazioni DNS, tutti gli LDNS sparsi nel mondo.

**Programmazione delle socket: come creare un’applicazione di rete**

**Programmazione delle socket con UDP**

UDPClient e server

**Programmazione delle socket con TCP**

TCPClient e server

salto 2.7 £

*Livello di Trasporto*

**Introduzione e servizi a livello di trasporto**

Un protocollo a livello di trasporto mette a disposizione una comunicazione logica (cioè che tutto proceda come se gli host che eseguono i processi fossero direttamente connessi dal punto di vista dell’applicazione) tra processi applicativi di host differenti.

**Relazione tra i livelli di trasporto e di rete**

I servizi che un protocollo di trasporto può offrire sono spesso vincolati al modello di servizio del protocollo sottostante a livello di rete. Se quest’ultimo non può fornire garanzie sul ritardo o sulla banda per i segmenti a livello di trasporto scambiati tra host, allora il protocollo a livello di trasporto non può certo offrire garanzie sul ritardo o sulla banda per i messaggi applicativi inviati tra processi. Un protocollo di trasporto può offrire il servizio di trasferimento dati affidabile alle applicazioni anche quando il sottostante protocollo di rete non è affidabile. Un protocollo di trasporto può usare la crittografia per garantire che i messaggi delle applicazioni non vengano letti da intrusi, anche quando il livello di rete non può garantire la riservatezza dei segmenti a livello di trasporto.

I protocolli di trasporto di Internet sono TCP e UDP.

**Multiplexing e demultiplexing**

Come il servizio di trasporto da host a host fornito dal livello di rete possa diventare un servizio di trasporto da processo a processo per le applicazioni in esecuzione sugli host.

Un processo (come parte di un’applicazione di rete) può gestire una o più socket (che agisce da intermediario), attraverso le quali i dati fluiscono dalla rete al processo e viceversa. Il compito di trasportare i dati dei segmenti a livello di trasporto verso la giusta socket viene detto demultiplexing. Il compito di radunare frammenti di dati da diverse socket sull’host di origine e incapsulare ognuno con intestazioni a livello di trasporto (che verranno poi utilizzate per il demultiplexing) per creare dei segmenti e passarli al livello di rete, viene detto multiplexing.

Possono essere orientati alla connessione (TCP) o meno (UDP). £

**Trasporto non orientato alla connessione: UDP**

In UDP non esiste handshaking tra le entità di invio e di ricezione a livello di trasporto. Per questo motivo, si dice che UDP è non orientato alla connessione. Fa praticamente il minimo che un protocollo di trasporto debba fare. A parte la funzione di multiplexing/demultiplexing e una forma di controllo degli errori molto semplice, non aggiunge nulla a IP. L’applicazione dialoga quasi in modo diretto con IP.

Si sceglie UDP per i seguenti motivi:

- Controllo più fine a livello di applicazione su quali dati sono inviati e quando.

- Nessuno stato di connessione, non conserva lo stato della connessione e non tiene traccia dei parametri per il controllo della congestione e parametri sul numero di sequenza e di acknowledgment (segnale).

- Minor spazio usato per l’intestazione del pacchetto

Viene utilizzato principalmente per: Server di file remoti (NFS), Gestione di rete(SNMP), Traduzione di nomi(DNS).

SI sceglie tra UDP e TCP per: Dati multimediali in streaming e Telefonia su Internet.

**Struttura dei segmenti UDP**

Numero di porta di origine, Numero di porta di destinazione, lunghezza, checksum, Dati dell’applicazione (messaggio)

**Checksum UDP**

Il checksum UDP serve per il rilevamento degli errori. In altre parole, viene utilizzato per determinare se i bit del segmento UDP sono stati alterati durante il loro trasferimento (per esempio, a causa di disturbi nei collegamenti o quando sono stati memorizzati in un router) da sorgente a destinazione. Lato mittente UDP effettua il complemento a 1 della somma di tutte le parole da 16 bit nel segmento, e l’eventuale riporto finale viene sommato al primo bit.

UDP non fa nulla per risolvere le situazioni di errore, alcune implementazioni di UDP si limitano a scartare il segmento danneggiato, altre lo trasmettono all’applicazione con un avvertimento.

**Princìpi del trasferimento dati affidabile**

L’astrazione del servizio offerta alle entità dei livelli superiori è quella di un canale affidabile tramite il quale si possono trasferire dati. Con un canale affidabile a disposizione nessun bit dei dati trasferiti è corrotto (0 al posto di 1 o viceversa) o va perduto e tutti i bit sono consegnati nell’ordine di invio. Si tratta precisamente del modello di servizio offerto da TCP alle applicazioni per Internet che ne fanno uso. Il compito di un protocollo di trasferimento dati affidabile è l’implementazione di questa astrazione del servizio. Ciò è reso difficile dalla possibile inaffidabilità del livello “al di sotto” del protocollo di trasferimento dati.

**Costruzione di un protocollo di trasferimento dati affidabile**

Trasferimento dati affidabile su un canale perfettamente affidabile: rdt1.0

Trasferimento dati affidabile su un canale con errori sui bit: rdt2.0

Trasferimento dati affidabile su un canale con perdite ed errori sui bit: rdt3.0

**Protocolli per il trasferimento dati affidabile con pipeline**

**Go-Back-N (GBN)**

In un protocollo Go-Back-N (GBN) il mittente può trasmettere più pacchetti senza dover attendere alcun acknowledgment, ma non può avere più di un dato numero massimo consentito N di pacchetti (se disponibili) in attesa di acknowledgment nella pipeline. N viene chiamato ampiezza della finestra (window size) e il protocollo GBN viene detto protocollo a finestra scorrevole (sliding- window protocol). Il mittente GBN deve rispondere a tre tipi di evento.

- Invocazione dall’alto: Quando dall’alto si chiama rdt\_send(), come prima cosa il mittente controlla se la finestra sia piena, ossia se vi siano N pacchetti in sospeso senza acknowledgment. Se la finestra non è piena, crea e invia un pacchetto e le variabili vengono aggiornate di conseguenza. Se la finestra è piena, il mittente restituisce i dati al livello superiore che, presumibilmente, ritenterà più tardi.

- Ricezione di un ACK: acknowledgment del pacchetto con il numero di sequenza n verrà considerato un acknowledgment cumulativo (cumulative acknowledgment), che indica che tutti i pacchetti con un numero di sequenza minore o uguale a n sono stati correttamente ricevuti dal destinatario.

- Evento di timeout: Quando si verifica un timeout, il mittente invia nuovamente tutti i pacchetti spediti che ancora non hanno ricevuto un acknowledgment.

£

In una tale programmazione basata su eventi le diverse procedure vengono invocate da altre procedure nella pila di protocolli o in conseguenza di un interrupt.

**Ripetizione selettiva**

I protocolli a ripetizione selettiva (SR, selective-repeat protocol) evitano le ritrasmissioni non necessarie facendo ritrasmettere al mittente solo quei pacchetti su cui esistono sospetti di errore (ossia, smarrimento o alterazione). Questa forma di ritrasmissione a richiesta e personalizzata costringe il destinatario a mandare acknowledgment specifici per i pacchetti ricevuti in modo corretto.

Il destinatario SR invia un riscontro per i pacchetti correttamente ricevuti sia in ordine sia fuori sequenza. Questi vengono memorizzati in un buffer finché non sono stati ricevuti tutti i pacchetti mancanti, momento in cui un blocco di pacchetti può essere trasportato in ordine al livello superiore.

£

**Trasporto orientato alla connessione: TCP**

**Connessione TCP**

TCP viene detto orientato alla connessione in quanto, prima di effettuare lo scambio dei dati, i processi devono effettuare l’handshake, ossia devono inviarsi reciprocamente alcuni segmenti preliminari per stabilire i parametri del successivo trasferimento dati. Come parte dell’instaurazione della connessione TCP, entrambe le parti inizializzano molte variabili di stato associate alla connessione. Lo stato della connessione risiede completamente nei due sistemi periferici dato che negli elementi di rete intermedi TCP non va in esecuzione.

Una connessione TCP è una connessione TCP è anche punto a punto e offre un servizio full-duplex.

buffer di invio della connessione

La massima quantità di dati prelevabili e posizionabili in un segmento viene limitata dalla dimensione massima di segmento (MSS, maximum segment size), impostato determinando l’unità trasmissiva massima (MTU, maximum transmission unit) e poi scegliendo un MSS tale che il segmento TCP stia all’interno di un singolo frame a livello di collegamento + l’intestazione TCP/IP.

TCP accoppia ogni blocco di dati del client a una intestazione TCP, andando pertanto a formare segmenti TCP. Questi vengono passati al sottostante livello di rete, dove sono incapsulati separatamente nei datagrammi IP a livello di rete che vengono poi immessi nella rete. Quando all’altro capo TCP riceve un segmento, i dati del segmento vengono memorizzati nel buffer di ricezione della connessione TCP.

Una connessione TCP è costituita da buffer, variabili e una connessione socket al processo per ogni host

**Struttura dei segmenti TCP**

Il segmento TCP consiste di campi intestazione e di un campo contenente un blocco di dati proveniente dall’applicazione. L’intestazione include:

- numeri di porta di origine e di destinazione, campo checksum come UDP

- campo numero di sequenza (sequence number) e il campo numero di acknowledgment: implementano il trasferimento dati affidabile

- campo finestra di ricezione: per il controllo di flusso.

- campo lunghezza dell’intestazione

- campo opzioni

- campo flag: bit ACK, RTS, STN, FIN, PSH, URG.

Numeri di sequenza e numeri di acknowledgment

Il numero di sequenza per un segmento è pertanto il numero nel flusso di byte del primo byte del segmento. Dato che TCP effettua l’acknowledgment solo dei byte fino al primo byte mancante nel flusso, si dice che tale protocollo offre acknowledgment cumulativi (cumulative acknowledgment).

Telnet: un caso di studio per i numeri di sequenza e di acknowledgment

E’ un protocollo a livello di applicazione impiegato per il login remoto che utilizza TCP ed è progettato per funzionare tra qualsiasi coppia di host. A differenza delle applicazioni di trasferimento massiccio di dati, Telnet è un’applicazione interattiva.

Il primo segmento inviato dal client avrà numero di sequenza 42 e quello inviato dal server 79.

Il secondo segmento spedito dal server al client ha un duplice scopo: fa un acknowledgment dei dati ricevuti dal server e manda indietro un “eco” della lettera.

Acknowledgment piggybacked: dati dal client al server viene trasportato in un segmento che a sua volta trasporta dati dal server al client. E’ piggybacked su quel segmento.

Il terzo segmento viene inviato dal client al server e ha come unico scopo dare un acknowledgment ai dati inviati dal server.

**Timeout e stima del tempo di andata e ritorno**

Stima del tempo di andata e ritorno

L’RTT misurato di un segmento, denotato come SampleRTT, è la quantità di tempo che intercorre tra l’istante di invio del segmento (ossia quando viene passato a IP) e quello di ricezione dell’acknowledgment del segmento.

EstimatedRTT è una media mobile esponenziale ponderata (EWMA) dei valori di SampleRTT:

EstimatedRTT= (1 – a) × EstimatedRTT + a × SampleRTT Il valore raccomandato per a: 0,125

DevRTT è una stima di quanto SampleRTT generalmente si discosta da EstimatedRTT:

DevRTT = (1 – β) × DevRTT + β × | SampleRTT – EstimatedRTT Il valore suggerito per β: 0,25

Impostazione e gestione del timeout di ritrasmissione

TCP determina l’intervallo di timeout di ritrasmissione con questa formula:

TimeoutInterval = EstimatedRTT + 4 × DevRTT

valore iniziale di TimeoutInterval pari a 1 secondo e quando si verifica un timeout, TimeoutInterval viene raddoppiato. Appena viene ricevuto un segmento ed EstimatedRTT viene aggiornato, TimeoutInterval viene ricalcolato

**Trasferimento dati affidabile**

TCP assicura che il flusso di byte che i processi leggono dal buffer di ricezione TCP non sia alterato, non abbia buchi, non presenti duplicazioni e rispetti la sequenza originaria; quindi sia esattamente quello spedito.

Esistono tre eventi principali relativi alla trasmissione e ritrasmissione dei dati:

- dati provenienti dall’applicazione: TCP incapsula i dati che gli giungono dall’applicazione in un segmento e lo passa a IP.

- timeout: TCP risponde ritrasmettendo il segmento che lo ha causato e quindi riavviando il timer.

- ricezione di un ACK: Quando si verifica il TCP confronta il valore y di ACK con la propria variabile SendBase (è il numero di sequenza del più vecchio byte che non ha ancora ricevuto un acknowledgment).

Alcuni scenari interessanti

£

Raddoppio dell’intervallo di timeout

Varianti utilizzate dalla maggior parte delle implementazioni TCP: lunghezza dell’intervallo di timeout dopo la scadenza di un timer. Con questa modifica, in tutti i casi in cui si verifica un timeout, TCP, come descritto precedentemente, ritrasmette il segmento con il più basso numero di sequenza che non abbia ancora ricevuto acknowledgment.

Ritrasmissione rapida

il mittente può in molti casi rilevare la perdita dei pacchetti ben prima che si verifichi l’evento di timeout grazie agli ACK duplicati relativi a un segmento il cui ACK è già stato ricevuto dal mittente. Nel caso in cui siano stati ricevuti tre ACK duplicati, il mittente TCP effettua una ritrasmissione rapida (fast retransmit) rispedendo il segmento mancante prima che scada il timer.

Go-Back-N o ripetizione selettiva?

TCP è un ibrido tra i protocolli di tipo GBN e SR.

Lati GBN: gli acknowledgment TCP sono cumulativi e che i segmenti ricevuti correttamente, ma in modo disordinato, non vengono notificati singolarmente dal destinatario. Il mittente TCP deve solo memorizzare il numero di sequenza più basso tra i

byte trasmessi che non hanno ancora ricevuto acknowledgment (SendBase) e il numero di sequenza del successivo byte da inviare (NextSeqNum).

Lati SR: riscontro selettivo (selective acknowledgment), consente al destinatario di mandare acknowledgment in modo selettivo per i segmenti non in ordine anziché cumulativamente per l’ultimo segmento ricevuto senza errori e nell’ordine giusto.

Ritrasmissione selettiva: evita la ritrasmissione dei segmenti per cui si è già ricevuto un acknowledgment in modo selettivo dal destinatario.

**Controllo di flusso**

TCP offre un servizio di controllo di flusso (flow-control service) alle proprie applicazioni per evitare che il mittente saturi il buffer del ricevente. Il controllo di flusso è pertanto un servizio di confronto sulla velocità, dato che paragona la frequenza di invio del mittente con quella di lettura dell’applicazione ricevente.

TCP offre controllo di flusso facendo mantenere al mittente una variabile chiamata finestra di ricezione (receive window) che, in sostanza, fornisce al mittente un’indicazione dello spazio libero disponibile nel buffer del destinatario.

£

**Gestione della connessione TCP**

Il TCP nel client quindi procede a stabilire una connessione TCP con il TCP nel server nel modo descritto di seguito:

1 - TCP lato client invia un segmento SYN (con il bit SYN posto a 1) al TCP lato server.

2 - Arriva all’host server, il server estrae il segmento dal datagramma, alloca i buffer e le variabili TCP alla connessione e invia un segmento di connessione approvata al client TCP. Il segmento ha il campo ACK con valore client\_isn+1 e il server sceglie il proprio numero di sequenza iniziale(server\_isn) e lo pone nel campo del numero di sequenza. Il segmento di connessione approvata viene detto segmento SYNACK.

3 - Alla ricezione del segmento SYNACK, anche il client alloca buffer e variabili alla connessione. L’host client invia quindi al server un altro segmento in risposta al segmento di connessione approvata del server. Tale operazione viene svolta dal client ponendo il valore server\_isn+1 nel campo ACK dell’intestazione del segmento TCP. Il bit SYN è posto a zero, dato che la connessione è stata stabilita.

Notiamo che al fine di stabilire la connessione, i due host si scambiano tre pacchetti. Per questo motivo tale procedura viene detta handshake a tre vie.

Per chiudere la connessione TCP il processo client invia un comando di chiusura, che forza il client TCP a inviare un segmento con bit FIN a 1 al processo server. Il server riceve questo segmento, risponde inviando un acknowledgment al client. Il server spedisce quindi il proprio segmento di shutdown, con il bit FIN uguale a 1. Infine, il client manda a sua volta un acknowledgment a quest’ultimo segmento del server. A questo punto, tutte le risorse degli host risultano deallocate.

Stati TCP: CLOSED, SYN\_SENT, ESTABLISHED, FIN\_WAIT\_1, FIN\_WAIT\_2, TIME\_WAIT

£

**Princìpi del controllo di congestione**

**Cause e costi della congestione**

Scenario 1: due mittenti e un router con buffer illimitati

Scenario 2: due mittenti e un router con buffer limitati

Scenario 3: quattro mittenti, router con buffer finiti e percorsi composti da più collegamenti

**Approcci al controllo di congestione**

- Controllo di congestione end-to-end:

- Controllo di congestione assistito dalla rete: I componenti a livello di rete (ossia i router) forniscono un feedback esplicito al mittente sullo stato di congestione della rete. L’informazione di congestione viene fornita dalla rete al mittente tramite chokepacket (“pacchetto di strozzatura”) che riferisce: “Sono congestionato!”, oppure il router imposta un campo in un pacchetto che fluisce dal mittente al destinatario.

È anche possibile fare uso di un feedback di rete più sofisticato tipo il controllo di congestione ATM ABR (available bit rate) consente a un router di informare il mittente in modo esplicito sulla frequenza trasmissiva che il router può supportare su un collegamento uscente.

**Controllo di congestione TCP**

Consiste nell’imporre a ciascun mittente un limite alla velocità di invio sulla propria connessione in funzione della congestione di rete percepita. La finestra di congestione (congestion window - cwnd) che impone un vincolo alla velocità di immissione di traffico sulla rete da parte del mittente. All’inizio di ogni RTT, il vincolo consente al mittente di trasmettere cwnd byte di dati sulla connessione; al termine del RTT il mittente riceve gli acknowledgment relativi ai dati. La velocità di invio del mittente è approssimativamente cwnd/RTT byte/s.

Dato che TCP utilizza gli acknowledgment per scatenare (o temporizzare) gli incrementi dell’ampiezza della finestra di congestione, si dice che TCP è auto-temporizzato (self-clocking).

Principi guida TCP:

- Un segmento perso implica congestione, quindi i tassi di trasmissione del mittente TCP dovrebbero essere decrementati quando un segmento viene perso.

- Un acknowledgment indica che la rete sta consegnando i segmenti del mittente al ricevente e quindi il tasso di trasmissione del mittente può essere aumentato quando arriva un acknowledgment non duplicato

- Rilevamento della larghezza di banda

L’algoritmo di controllo di congestione di TCP presenta tre componenti o fasi principali: (1) slow start, (2) congestion avoidance e (3) fast recovery.

Slow start

Quando si stabilisce una connessione TCP, il valore di cwnd viene in genere inizializzato a 1 MSS e si incrementa di 1 MSS ogni volta che un segmento trasmesso riceve un acknowledgment. Quindi, in TCP, la velocità di trasmissione parte lentamente, ma cresce in modo esponenziale durante la fase di slow start. Se c’è un evento di perdita si reimposta il cwnd a 1 e pone ssthresh a cwnd/2, quando il valore di cwnd è pari a ssthresh, la fase di slow start termina e TCP entra in modalità di congestion avoidance.

La fase di slow start può terminare è quando vengono rilevati tre acknowledgment duplicati, nel qual caso TCP opera una ritrasmissione rapida (si veda il Paragrafo 3.5.4) ed entra nello stato di fast recovery.

Congestion avoidance

Quando TCP entra nello stato di congestion avoidance, il valore di cwnd è circa la metà di quello che aveva l’ultima volta in cui era stata rilevata la congestione. Incrementa cwnd di 1 MSS ogni RTT, l’incremento da parte del mittente TCP della propria cwnd di MSS x byte ogni qualvolta riceva un nuovo acknowledgment.

L’algoritmo di congestion avoidance quando si verifica un timeout si comporta nello stesso modo di slow start.

In caso di acknowledgment duplicati TCP dimezza il valore di cwnd e imposta il valore di ssthresh pari a metà del valore di cwnd al momento del ricevimento dei tre ACK duplicati. Infine, TCP entra nello stato di fast recovery.

Fast recovery

Durante la fase di fast recovery il valore di cwnd è incrementato di 1 MSS per ogni ACK duplicato ricevuto relativamente al segmento perso che ha causato l’entrata di TCP nello stato di fast recovery. Quando arriva un ACK per il segmento perso, TCP entra nello stato di congestion avoidance dopo aver ridotto il valore di cwnd. Se si verifica un timeout vi è invece una transizione dallo stato di fast recovery a quello di slow start dopo avere effettuato le stesse azioni presenti sia in slow start che in congestion avoidance: il valore di cwnd è posto a 1 MSS e il valore di ssthresh è impostato a metà del valore di cwnd nel momento in cui si è riscontrato l’evento di perdita.

TCP Reno, adotta invece fast recovery. TCP Tahoe £

Retrospettiva sul controllo di congestione di TCP

Il controllo di congestione di TCP consiste in un incremento additivo lineare della cwnd pari a 1 MSS per RTT e quindi di un decremento moltiplicativo che dimezza la cwnd in corrispondenza di un evento di triplice ACK duplicato. Quindi decrementa la propria finestra di congestione di un fattore due, ma riprende ancora a crescere linearmente per capire se ci sia ulteriore ampiezza di banda disponibile.

Il controllo di congestione di TCP è spesso indicato come una forma di controllo di congestione incremento additivo, decremento moltiplicativo (AIMD, additive increase multiplicative-decrease)

Descrizione macroscopica del throughput di TCP

throughput medio di una connessione = (0,75 ×W)/RTT

Futuro di TCP sulle connessioni a larga banda

throughput medio di una connessione = (1,22 × MSS)/RTT⎷L

**Fairness**

Un meccanismo di controllo di congestione sia equo (fair) se la velocità trasmissiva media di ciascuna connessione è approssimativamente R/K (R capacità trasmissiva in bps; K connessioni); in altre parole, ciascuna connessione ottiene la stessa porzione di banda del collegamento.

AIMD fari? £

Fairness e UDP

Il meccanismo della finestra di congestione consenta al controllo di congestione TCP di regolare il tasso trasmissivo delle applicazioni. Questo è il motivo per cui molte applicazioni multimediali, quali la fonia e la videoconferenza, non fanno uso di TCP: non vogliono che il loro tasso trasmissivo venga ridotto, anche se la rete è molto congestionata. Piuttosto, queste applicazioni preferiscono utilizzare UDP, che non incorpora il controllo di congestione.

Le applicazioni multimediali che fanno uso di UDP non sono fair: non cooperano con altre né adeguano la loro velocità trasmissiva in modo appropriato.

Fairness e connessioni TCP parallele

A volte i browser web usano più connessioni TCP in parallelo per trasferire il contenuto delle pagine.

**Notifica esplicita di congestione (ECN): controllo di congestione assistito dalla rete**

forma di controllo della congestione del protocollo TCP. A livello di rete vengono utilizzati due bit (quindi quattro possibili valori) nel campo Tipo di Servizio dell’intestazione IP. Se un router è congestionato, imposta tali bit e invia il pacchetto IP contrassegnato al destinatario, che quindi informa il mittente.

Quando il destinatario TCP riceve un’indicazione di congestione, ne informa il mittente TCP impostando il bit ECE (explicit congestion notification echo) all’interno di un segmento ACK. Il mittente TCP reagisce dimezzando la finestra di congestione, esattamente come farebbe in caso di perdita di un segmento usando il meccanismo di ritrasmissione rapida, e imposta il bit CWR (congestion window reduced) nell’intestazione del successivo segmento che invia al ricevente.

*Livello di rete: piano dei dati*

**Panoramica del livello di rete**

**Inoltro e instradamento: piano dei dati e piano di controllo**

- Inoltro(forwarding) : Quando un router riceve un pacchetto, lo deve trasferire sull’appropriato collegamento di uscita

- Instradamento (routing). Il livello di rete deve determinare il percorso che i pacchetti devono seguire tramite algoritmi di instradamento (algoritmi di routing).

Per inoltrare i pacchetti, i router estraggono da uno o più campi dell’intestazione i loro valori che utilizzano come indice nella tabella di inoltro (tabella di forwarding o forwarding table), un elemento chiave di qualsiasi router.

Piano di controllo: l’approccio tradizionale

l’algoritmo di routing è implementato in ogni router, che quindi svolge sia la funzione di inoltro che quella di instradamento internamente

Piano di controllo: l’approccio SDN

Il piano di controllo mostrato nella Figura 4.3 è il cuore del software-defined networking (SDN), nel quale la rete è “software-defined” perché il controller che calcola le tabelle di inoltro e interagisce coi router è implementato in software

**Modelli di servizio**

modello di servizio della rete:

-Consegna garantita: assicura che il pacchetto giunga a destinazione

-Consegna garantita con ritardo limitato: come sopra ma con rispetto dei tempi

- consegna ordinata: garantisce che i pacchetti giungano alla destinazione nell’ordine in cui sono stati inviati

- Banda minima garantita: emula un collegamento trasmissivo con bit rate specificato (per esempio, 1 Mbps)

- Servizi di sicurezza: host sorgente può cifrare tutti i datagrammi inviati, la riservatezza viene fornita a tutti i segmenti

Il livello di rete di Internet mette a disposizione un solo servizio, noto come servizio best-effort, ossia “col massimo impegno possibile”. Con questo servizio, non c’è garanzia che i pacchetti vengano ricevuti nell’ordine in cui sono stati inviati, così come non è garantita la loro eventuale consegna.

**Che cosa si trova all’interno di un router?**

-Porte di ingresso (input port). Svolgono le funzioni a livello fisico di terminazione di un collegamento in ingresso al router

- Struttura di commutazione (switching fabric). La struttura di commutazione, che connette fisicamente le porte di ingresso a quelle di uscita, è interamente contenuta all’interno del router: una vera e propria rete in un router di rete!

- Porte di uscita (output port). Memorizzano i pacchetti che provengono dalla struttura di commutazione e li trasmettono sul collegamento in uscita

- Processore di instradamento (routing processor). Esegue le funzioni del piano di controllo. Nei router tradizionali, esegue i protocolli di instradamento. Nei router SDN, il processore di instradamento è responsabile della comunicazione con il controller remoto

Le funzioni del piano di controllo sono solitamente implementate via software ed eseguite sul processore di instradamento (che di solito è una CPU tradizionale).

l’inoltro dei pacchetti:

- Inoltro basato sulla destinazione:

- Inoltro generalizzato:

**Elaborazione alle porte di ingresso e inoltro basato sull’indirizzo di destinazione**

Consideriamo il caso “più semplice” in cui l’inoltro è basato sull’indirizzo di destinazione.

Intervallo degli indirizzi di destinazione Interfaccia

da 11001000 00010111 00010000 00000000 0

a 11001000 00010111 00010111 11111111

da 11001000 00010111 00011000 00000000 1

a 11001000 00010111 00011000 11111111

da 11001000 00010111 00011001 00000000 2

a 11001000 00010111 00011111 11111111

altrimenti 3

Corrispondenza di prefisso Interfaccia

11001000 00010111 00010 0

11001000 00010111 00011000 1

11001000 00010111 00011 2

altrimenti 3

Con questa struttura il router confronta un prefisso dell’indirizzo di destinazione del pacchetto con una riga della tabella; se c’è corrispondenza, il router inoltra il pacchetto al collegamento associato.

Quando si verificano corrispondenze multiple, il router adotta la regola di corrispondenza a prefisso più lungo: viene determinata la corrispondenza più lunga all’interno della tabella e i pacchetti vengono inoltrati all’interfaccia di collegamento associata. Una volta determinata la porta di output di un pacchetto, esso può essere inviato alla struttura di commutazione. In alcune architetture di router un pacchetto può essere temporaneamente fermato prima di entrare nella struttura di commutazione quando essa è utilizzata da altre porte di input.

**Struttura di commutazione**

La commutazione può essere ottenuta in vari modi :

• Commutazione in memoria: Quando sopraggiungeva un pacchetto, la porta di ingresso ne segnalava l’arrivo tramite interrupt e quindi lo copiava nella memoria del processore di instradamento che procedeva a estrarre dall’intestazione l’indirizzo di destinazione. La ricerca dell’indirizzo di destinazione e la memorizzazione del pacchetto nella locazione di memoria opportuna  
vengono effettuate dai processori direttamente sulle line card di ingresso

• Commutazione tramite bus: le porte di ingresso trasferiscono un pacchetto direttamente alle porte di uscita tramite un bus condiviso e senza intervento da parte del processore di instradamento. Se più pacchetti arrivano contemporaneamente al router, ognuno su una porta di input diversa, tutti tranne uno dovranno aspettare, dato che sul bus si può trasferire soltanto un pacchetto alla volta

• Commutazione attraverso rete di interconnessione. Un modo per superare la limitazione di banda di un singolo bus condiviso. Una matrice di commutazione (crossbar switch) è una rete di interconnessione che consiste di 2n bus che collegano n porte d'ingresso a n porte di uscita

**Dove si verifica l’accodamento?**

si possono formare code di pacchetti sia presso le porte di ingresso sia presso quelle di uscita. Il luogo e la lunghezza della dipendono dalla quantità di traffico di rete, dalle velocità relative della struttura di commutazione e dalla linea. Quando queste code crescono, la memoria del router può esaurirsi e quindi può avvenire una perdita di pacchetti nel caso non vi sia memoria per immagazzinare quelli in arrivo

Accodamento in ingresso

se la struttura di commutazione non è sufficientemente rapida

blocco in testa alla coda (HOL, head-of-the-line blocking): pacchetto nella coda di ingresso deve attendere il trasferimento attraverso la struttura in quanto risulta bloccato da un altro pacchetto che lo precede

Accodamento in uscita

è possibile che si formino code di pacchetti alle porte di uscita anche quando la struttura di commutazione è N volte più rapida delle velocità di linea delle porte. In assenza di sufficiente memoria per inserire nel buffer il nuovo pacchetto in ingresso, occorrerà stabilire se scartarlo (politica nota come drop-tail, eliminazione in coda) o se rimuoverne uno o più, fra quelli già in coda. Sono state proposte e analizzate numerose politiche di eliminazione e marcatura dei pacchetti, tecniche che presentano il nome collettivo di algoritmi di AQM (active queue management, gestione attiva della coda). tra questi c’è l’algoritmo RED (random early detection). Quando vi sono più pacchetti accodati sulle porte di uscita, uno schedulatore di pacchetti (packet scheduler) deve stabilire in quale ordine trasmetterli.

**Schedulazione dei pacchetti**

determinare l’ordine con cui i pacchetti vengono trasmessi sulla porta di uscita

Primo-arrivato-primo-servito (FIFO)

Code con priorità: pacchetti sono classificati in base a classi di priorità

Nella modalità di accodamento a priorità non prelazionabile (non-preemptive priority queuing), la trasmissione dei pacchetti non può essere interrotta una volta iniziata

Round robin e accodamento equo ponderato (WFQ): prima viene inviato un pacchetto della classe 1 e poi uno della classe 2, quindi nuovamente un pacchetto di classe 1, e così via.

Nella modalità conservativa (work-conserving round robin), il collegamento non resta mai inattivo fintanto che ci sono pacchetti, di qualsiasi classe, da trasmettere: se la coda di una classe è vuota viene immediatamente consultata quella successiva

Un’astrazione generalizzata di questa strategia che ha trovato vasto impiego nei router è la cosiddetta modalità di accodamento equo ponderato (WFQ, weighted fair queuing). Diversamente da round robin, con WFQ le varie classi possono ricevere un servizio differenziato

**Il protocollo Internet (IP): IPv4, indirizzamento, IPv6 e altro ancora**

**Formato dei datagrammi IPv4**

Il pacchetto a livello di rete è noto come datagramma. I campi dei datagrammi IPv4:

-Numero di versione: 4 bit

-Lunghezza dell’intestazione: 4 bit, indicano dove iniziano effettivamente i dati del datagramma

-Tipo di servizio (TOS): distinguere diversi tipi di datagrammi (quelli che richiedono basso ritardo, alto throughput o affidabilità)

-Lunghezza del datagramma: 16 bit

-Identificatore, flag, offset di frammentazione: servono per la frammentazione

-Tempo di vita (Time To Live TTL): decrementato di un’unità ogni volta che il datagramma è elaborato da un router; quando raggiunge 0, il datagramma deve essere scartato

-Protocollo: indica lo specifico protocollo a livello di trasporto al quale vanno passati i dati del datagramma (6: TCP, 17: UDP)

-Checksum dell’intestazione: Consente ai router di rilevare gli errori sui bit nei datagrammi ricevuti.

-Indirizzi IP sorgente e destinazione.

-Opzioni: consentono di estendere l’intestazione IP

-Dati (payload).

**Frammentazione dei datagrammi IPv4**

La massima quantità di dati che un frame a livello di collegamento può trasportare è detta unità massima di trasmissione (MTU, maximum transmission unit). Come dividete il datagramma nel campo dati del frame a livello di collegamento? La soluzione consiste nel frammentare i dati del datagramma IP in due o più datagrammi IP più piccoli, detti frammenti, e quindi trasferirli sul collegamento di uscita.

**Indirizzamento IPv4**

Il confine tra host e collegamento fisico viene detto interfaccia.

IP assegna a questa sottorete l’indirizzo 223.1.1.0/24, dove la notazione /24, detta anche maschera di sottorete (subnet mask).

La strategia di assegnazione degli indirizzi Internet è detta classless interdomain routing CIDR generalizza la nozione di indirizzamento di sottorete

Gli x bit più a sinistra di un indirizzo della forma a.b.c.d/x costituiscono la porzione di rete dell’indirizzo IP e sono spesso detti prefisso (di rete) dell’indirizzo

Come ottenere l’indirizzo di un host: DHCP

host possono essere configurati manualmente, ma più spesso questo compito è svolto utilizzando il dynamic host configuration protocol (DHCP). DHCP consente a un host di ottenere un indirizzo IP in modo automatico. Si può configurare DHCP di modo che un dato host riceva un indirizzo IP persistente, in modo che ogni volta che l’host entra in rete gli venga assegnato sempre lo stesso indirizzo IP, oppure in modo da assegnare a ciascun host che si connette un indirizzo IP temporaneo. DHCP viene spesso detto protocollo plug-and-play o zero-conf per la sua capacità di automatizzare la connessione degli host alla rete.

Per i nuovi host, il protocollo DHCP si articola in quattro punti:

-Individuazione del server DHCP: è svolta utilizzando un messaggio DHCPdiscover, che un client invia.

-Offerta del server DHCP: risponde al client con un messaggio DHCPoffer, che viene inviato in broadcast a tutti i nodi della sottorete

-Richiesta DHCP: sceglierà tra le offerte dei server e risponderà con un messaggio DHCP request, che riporta i parametri di configurazione.

- Conferma DHCP: Il server risponde al messaggio di richiesta DHCP con un messaggio DHCP ACK, che conferma i parametri richiesti

**NAT (network address translation)**

Lo spazio di indirizzamento 10.0.0.0/8 è una delle tre parti dello spazio di indirizzi IP riservato alle reti private, o reame (realm)

I router abilitati al NAT si comportano come un unico dispositivo con un unico indirizzo IP e nasconde i dettagli della rete domestica al mondo esterno. Spesso calcolatori e router acquisiscono l’IP dal DHCP. Il router ottiene il proprio indirizzo dal server DHCP dell’ISP e manda in esecuzione un server DHCP per fornire gli indirizzi ai calcolatori all’interno dello spazio di indirizzamento della rete domestica.

come apprende il router a quale host interno dovrebbe essere inoltrato un determinato datagramma? Il trucco consiste nell’utilizzare una tabella di traduzione NAT (NAT translation table) nel router NAT e nell’includere nelle righe di tale tabella i numeri di porta oltre che gli indirizzi IP

**IPv6**

Formato dei datagrammi IPv6

cambiamenti:

-Indirizzamento esteso

-Intestazione ottimizzata di 40 byte

-Etichettatura dei flussi

Campi:

-Versione: 4 bit

-Classe di traffico: 8 bit, simile a TOS IPv4, attribuisce priorità o provenienza dei datagrammi

-Etichetta di flusso: 20 bit, identifica flusso di datagrammi

-Lunghezza del payload: 16 bit

-Intestazione successiva: come campo Protocollo di ipv4

-Limite di hop: decrementato di 1 da ciascun router che inoltra il datagramma. Quando il suo valore raggiunge 0, il datagramma viene eliminato

-Indirizzi sorgente e destinazione

-Dati.

stati eliminati vari campi dell’ Ipv4: Frammentazione/riassemblaggio, Checksum dell’intestazione, Opzioni.

**Inoltro generalizzato e SDN**

l’inoltro basato sulla destinazione è caratterizzato da due passi: ricerca dell’indirizzo IP di destinazione (“match”), quindi invio del pacchetto attraverso la struttura di commutazione a una specifica porta di uscita (“action”). Consideriamo ora un paradigma molto più generale, il paradigma “match action” (corrispondenza e azione), dove “match” può essere effettuato su più campi dell’intestazione associati a differenti protocolli corrispondenti a diversi livelli nello stack dei protocolli.

Ogni occorrenza (riga) in una tabella di inoltro match-action, nota come tabella dei flussi (flow table) in OpenFlow (protocollo di comunicazione che consente l'accesso al piano di inoltro di un dispositivo di rete attraverso la rete), contiene:

-Un insieme di valori dei campi dell’intestazione

-Un insieme di contatori

-Un insieme di azioni

**Match**

Un match può essere effettuato sui campi delle intestazioni di tre livelli di protocollo (Livello di collegamento, Livello di rete, Livello di trasporto).

**Action**

ogni occorrenza della tabella dei flussi ha una lista di azioni che determinano l’elaborazione da effettuare su un pacchetto che le corrisponde. In caso vi siano molteplici azioni, vengono effettuate nell’ordine specificato nella lista. Azioni possibili:

-Inoltro (forwarding)

-Scarto (dropping)

-Modifica dei campi (modify-field)

*Reti multimediali*

**Applicazioni multimediali di rete**

**Proprietà del video**

Forse la caratteristica più saliente del video è l’elevato tasso con cui è necessario inviare i bit sulla rete (bit rate). Nella compressione video vengono sfruttati due tipi di ridondanza. La ridondanza spaziale è quella all’interno di una data immagine. Intuitivamente, un’immagine formata per lo più dallo stesso colore omogeneo ha un alto grado di ridondanza e può essere compressa in modo efficace senza sacrificarne in modo significativo la qualità. La ridondanza temporale è data dalla ripetizione della stessa immagine in due tempi successivi. La compressione può essere usata anche per creare versioni multiple dello stesso video, a livelli di qualità diversi

**Proprietà dell’audio**

il segnale audio analogico viene normalmente convertito in un segnale digitale secondo la tecnica della modulazione a codifica di impulso:

-Per prima cosa si procede al campionamento del segnale analogico a una frequenza fissata

-Si procede poi con l’operazione di quantizzazione

-Tutti i valori di quantizzazione sono rappresentati dallo stesso numero di bit

**Tipi di applicazioni multimediali**

streaming audio/video di contenuti registrati

Tre sono le caratteristiche fondamentali: Streaming Interattività Riproduzione continua

conversazione voce/video su IP

Questa classe di applicazioni viene comunemente definita telefonia Internet o Voice over-IP (VoIP), simile al tradizionale servizio telefonico a commutazione di circuito. L’aspetto della temporizzazione è importante, in quanto queste applicazioni sono altamente sensibili ai ritardi (delay-sensitive). D’altro canto, tali applicazioni sono tolleranti alle perdite (loss-tolerant).

streaming audio/video in tempo reale (live)

Consente agli utenti di ricevere in diretta trasmissioni radio o televisive, stazioni radio e televisive che distribuiscono  
contenuti attraverso Internet. la loro distribuzione attualmente avviene tramite CDN

**Streaming di video registrato**

Una volta che il client ha costruito tale riserva di alcuni secondi di video memorizzati, ma non ancora riprodotti, può iniziare la riproduzione. Vi sono due importanti vantaggi derivati da questa procedura di buffering lato client. Se la banda tra server e client improvvisamente va al di sotto del tasso di consumo dei dati, l’utente non se ne accorge finché il buffer non sarà vuoto.

**Streaming UDP**

il server trasmette il video allo stesso bit rate a cui il client lo consuma. Poiché UDP non ha un meccanismo di controllo di congestione, il server può immettere nella rete i pacchetti al tasso di consumo del video senza le restrizioni di TCP. Il server usa il protocollo di trasporto in tempo reale (RTP, real-time transport protocol) per incapsulare i blocchi video in pacchetti di trasporto.

Oltre al flusso video utilizza il real-time streaming protocol (RTSP) per connessioni di controllo sulla quale il client invia i comandi riguardanti i cambiamenti di stato della sessione, quali la pausa, la ripresa della riproduzione ecc. E’ anche uno svantaggio per lo streaming udp a causa della sua complessità e dei suoi costi

**Streaming HTTP**

Nello streaming HTTP il video viene semplicemente memorizzato in un server HTTP come un file ordinario con un URL specifico. Quando un utente vuole vedere un video, il client stabilisce una connessione TCP con il server e invia una richiesta GET HTTP per il suo URL. Il server invia il file video. Inoltre, l’uso di HTTP e TCP permette ai video di attraversare più facilmente firewall e i NAT che sono spesso configurati per bloccare la maggior parte del traffico UDP.

Prefetching del video

il client tenta di scaricare il video a un tasso più alto di quello di consumo, memorizzato nel buffer dell’applicazione. Il prefetching avviene in modo naturale con lo streaming TCP, in quanto il meccanismo di controllo della congestione di TCP tenterà di usare tutta la banda disponibile tra il server e il client

Analisi dello streaming video

quando il tasso disponibile nella rete è minore di quello del video, la riproduzione subisce periodi alternati di riproduzione continua e di fermi immagine. Quando è maggiore l’utente, dopo un ritardo di buffering iniziale, potrà ottenere una riproduzione continua fino alla fine del video

Interruzione anticipata e riposizionamento del video

I sistemi di streaming HTTP spesso usano l’intestazione HTTP byte-range nel messaggio di richiesta GET di HTTP, che specifica l’intervallo di byte del video desiderato che il client vuole ricevere. Questa procedura è particolarmente utile quando l’utente vuole saltare a un punto successivo del video; il client invia una nuova richiesta HTTP, indicando da quale byte del file il server dovrebbe inviare i dati. Il server, quando riceve la nuova richiesta HTTP, si dimentica di quelle precedenti e invia i byte cominciando da quello indicato nella nuova richiesta.

streaming dinamico adattativo su HTTP (DASH, dynamic adaptive streaming over HTTP)

Lo streaming HTTP presenta un grande svantaggio: i client ricevono la stessa versione codificata del video, nonostante abbiano disponibile una larghezza di banda che può variare sensibilmente da un caso all’altro e nel tempo. Per superare il problema è stato sviluppato un nuovo tipo di streaming basato su HTTP, chiamato streaming dinamico adattivo su HTTP (DASH, dynamic adaptive streaming over HTTP). In DASH, i video vengono codificati in diverse versioni, ognuna avente un bit rate differente e quindi un differente livello di qualità, inoltre permette a client con accessi a Internet diversi di fare streaming dei video con codifiche diverse: client con connessioni lente 3G possono ricevere versioni a basso bit rate e quindi a bassa qualità, mentre client con connessioni in fibra possono ricevere una versione ad alta qualità. Il server HTTP ha anche un file di descrizione (detto anche manifest, o manifesto), che per ogni versione fornisce il rispettivo URL insieme al bitrate

**Voice-over-IP**

Il servizio di conversazione in tempo reale su Internet viene comunemente chiamato telefonia Internet o Voice-over-IP (VoIP)

**Limiti del servizio best-effort di IP**

Come detto in precedenza, il protocollo a livello di rete di Internet, IP, fornisce un servizio best-effort, ossia fa del suo meglio per recapitare i datagrammi il più velocemente possibile, senza però fornire alcuna assicurazione in merito al ritardo, alla perdita o all’entità del jitter dei pacchetti.

Perdita di pacchetti

Il segmento è incapsulato in un datagramma IP che, mentre viaggia nella rete, passa attraverso i buffer dei router che possono essere pieni e non possano ricevere il datagramma che viene scartato. TCP ritrasmette i pacchetti persi ma è inaccettabile per trasmissioni in tempo reale.

Per queste ragioni, quasi tutte le applicazioni di VoIP fanno uso per default di UDP, senza curarsi della ritrasmissione dei pacchetti persi. UDP viene usato da Skype a meno che l’utente sia dietro un NAT o un firewall che blocca i segmenti UDP, nel qual caso usa TCP. a seconda di come la voce è codificata e trasmessa, e di come la perdita è mascherata in ricezione, per esempio utilizzando la correzione anticipata degli errori (FEC, forward error correction) che trasmette informazioni ridondanti insieme a quelle originali.

Ritardo end-to-end

La locuzione end-to-end delay indica la somma dei ritardi di trasmissione, di elaborazione e di accodamento nei router, più quelli di propagazione lungo i collegamenti e di elaborazione sui terminali. VoIP trascura solitamente i pacchetti con ritardi superiori a una certa soglia, generalmente di 400 ms. Quindi, questi pacchetti sono effettivamente persi.

Jitter di un pacchetto

Variazione del ritardo del pacchetto. Variabilità nelle code dei router, che può generare sostanziali differenze nel tempo impiegato da ciascun pacchetto tra origine e destinazione, da quando viene creato a quando viene ricevuto. l’intervallo con cui i due pacchetti giungono al ricevente può essere superiore a quello di trasmissione. Se il ricevente ignora il jitter, e riproduce i blocchi man mano che sopraggiungono, la qualità audio può risultare non intelligibile. Fortunatamente, spesso il jitter può essere rimosso tramite numeri di sequenza (sequence number), marcature temporali (timestamp) e ritardo di riproduzione (playout delay), come discusso qui di seguito.

**Rimozione del jitter al ricevente**

- Facendo precedere i blocchi da una marcatura temporale

- Inserendo un ritardo nella riproduzione del blocco al suo ricevimento, che deve essere abbastanza lungo da consentire la ricezione della maggior parte dei pacchetti prima di iniziare la loro riproduzione. Questo ritardo può essere fissato per tutta la durata della conferenza, o variato durante il suo svolgimento.

strategie di riproduzione:

Ritardo di riproduzione fisso

Con questa strategia il ricevente tenta di riprodurre ciascun blocco esattamente q millisecondi dopo che è stato generato. Occorre stabilire un importante compromesso fra ritardo e perdita dei pacchetti.

Ritardo di riproduzione adattativo

Scegliendo un ampio ritardo di riproduzione iniziale, molti pacchetti arriveranno in tempo utile e le perdite saranno trascurabili. Vorremmo un ritardo di riproduzione minimo con il vincolo che la perdita sia al di sotto dei pochi punti percentuale. Questa regolazione adattativa del ritardo di riproduzione all’inizio dei periodi di attività vocale farà in modo che le pause dei trasmittenti siano compresse o prolungate, secondo la necessità. Questa soluzione, se contenuta in limiti ragionevoli, non è rilevabile dall’ascoltatore.

**Recupero dei pacchetti persi**

schemi di recupero delle perdite (loss recovery scheme):

Correzione anticipata degli errori (FEC)

Aggiunge informazioni ridondanti al flusso originale dei pacchetti. In cambio di un aumento marginale nel tasso trasmissivo dell’audio, la ridondanza può essere utilizzata per fornire un’approssimazione dell’esatta versione di alcuni pacchetti persi.

Due meccanismi FEC:

Il primo meccanismo invia, dopo ogni n blocchi, un blocco ridondante ottenuto da un’operazione di OR esclusivo degli n blocchi originali. Ma se in un gruppo si perdono due o più pacchetti, il ricevente non può ricostruirli.

Il secondo meccanismo FEC consiste nell’inviare uno stream audio a bassa risoluzione come informazione ridondante. Ogni volta che si perdono pacchetti non consecutivi, il ricevente può mascherare la perdita riproducendo il blocco codificato a bassa velocità che arriva con il pacchetto successivo.

Interfogliazione

(interleaving) può ridurre gli effetti della perdita di pacchetti. Il trasmittente pone in sequenza le unità dati audio, in modo che quelle adiacenti siano separate da una data distanza nel flusso trasmesso.

**Protocolli per applicazioni in tempo reale**

**RTP (Real-time Transport Protocol)**

Utilizzato per servizi di comunicazione in tempo reale. Un’applicazione multimediale aggiunge campi di intestazione ai blocchi audio prima di passarli al livello di trasporto. RTP può essere utilizzato per trasportare formati comuni come PCM, ACC e MP3 per l’audio e MPEG e H.263 per il video, ma anche per formati proprietari.

RTP utilizza UDP: il lato trasmittente incapsula un blocco di dati audio o video in un pacchetto RTP, incapsula poi quest’ultimo in un segmento UDP e lo affida a IP. Il lato ricevente estrae il pacchetto RTP dal segmento UDP, recupera il contenuto multimediale dal pacchetto e lo passa al media player per la decodifica e la riproduzione. Utilizzo di RTP per il trasporto della voce. Il lato trasmittente fa precedere ciascun blocco da una intestazione RTP, Il blocco e l’intestazione formano il pacchetto RTP che viene inviato, tramite una socket UDP, al lato ricevente. L’applicazione estrae il blocco audio dal pacchetto RTP e utilizza l’intestazione per decodificarlo e riprodurlo correttamente.

RTP non fornisce alcun meccanismo per assicurare la spedizione tempestiva dei dati o altre forme di qualità del servizio e non garantisce nemmeno la consegna dei pacchetti né il loro ordine.

Flussi audio e video emessi da più trasmittenti in una videoconferenza costituiscono una sessione RTP.

Intestazione dei pacchetti RTP

-Numero di sequenza (16 bit): Il numero di sequenza è incrementato di un’unità per ogni pacchetto RTP inviato e può essere utilizzato dal ricevente per rilevare le perdite e ricostruire la sequenza dei pacchetti. (es. da 86 a 89 allora 87 e 88 persi)

-Marcatura temporale (32 bit): Riporta l’istante del campionamento del primo byte nel pacchetto dati RTP.

-Identificatore della sorgente di sincronizzazione (32 bit): Identifica la sorgente del flusso RTP.

**SIP (session initiation protocol)**

Il protocollo SIP è un protocollo “leggero” ed aperto che offre i seguenti servizi:

-Fornisce i meccanismi che consentono al chiamante di connettersi al chiamato su una rete IP e notificargli che vuole iniziare una chiamata

- Fornisce al chiamante i meccanismi necessari per determinare l’attuale indirizzo IP del chiamato

- Fornisce le procedure per la gestione della chiamata e aggiungere nuovi flussi multimediali

**Supporto di Internet alle applicazioni multimediali**

Meccanismi a livello di applicazione possono essere usati dalle applicazioni multimediali per aumentare le loro prestazioni; abbiamo anche visto come si possono usare le CDN e le reti P2P per fornire un approccioa livello di sistema alla consegna dei contenuti multimediali. Tutte queste tecniche e approcci sono stati progettati per essere usati nell’attuale Internet che fa uso di best-effort (servizio che non dà alcuna garanzia dell'effettiva consegna dei dati né tantomeno livelli di qualità o priorità garantiti); in realtà vengono usati proprio perché attualmente Internet fornisce solo una classe di servizio: quella best-effort.

tre grandi approcci per il supporto a livello rete delle applicazioni multimediali:

• Utilizzare al meglio il servizio best-effort. I meccanismi a livello di applicazione e le infrastrutture appena viste sono adeguate quando la rete è ben dimensionata.

•Servizi differenziati. Sin dall’inizio di Internet si pensava di assegnare a tipi di traffico diversi differenti classi di servizio, invece della sola best-effort (priorità).

• Garanzia di Qualità del Servizio (QoS, Quality of Service) per connessione. Si intende che ogni istanza di un’applicazione prenota esplicitamente banda end-to-end e ha fissate garanzie di prestazione. Una garanzia stringente (hard guarantee) significa che un’applicazione riceverà sicuramente la qualità di servizio che ha richiesto, mentre una garanzia lasca (soft guarantee) significa che riceverà la qualità di servizio che ha richiesto con elevata probabilità.

**FORMULE**

Tempo necessario per trasmettere L-bit(L dimensione pacchetto) nel collegamento

L/R L = bits R = bits/sec

Ritardo di trasmissione

L/R L = bits R = bits/sec

Ritardo di propagazione

d/v d = distanza tra i due router v = velocità di propagazione nel collegamento

-----------------------------------------------------------------------------------

Intensità di traffico

La / R L = bits R = bits/sec a= tasso medio di arrivo dei pacchetti

Se:

= 0 : ritardo di coda medio piccolo

<= 1 : va bene

> 1 : ritardo di coda medio alto, perdita di pacchetti

-----------------------------------------------------------------------------------

Throughput(in bps):

- Istantaneo: te lo da Pau spero per te

- Medio: F/T F = bit T = secondi

-----------------------------------------------------------------------------------

Tempo di distribuzione

Dcs = max {NF/us, F/dmin} NF = N peer x F bit us = banda di upload del server (massimo)

F = bit dmin = peer con la banda di download più bassa

Tempo di distribuzione minimo per l’architettura P2P

Dp2p = max{F/us, F/dmin, NF/us + (u1 + u2...uN)}

-----------------------------------------------------------------------------------

Performarce rdt3.0(Stop and wait)

dt = L/R = 8000 bit per pacchetto / 10E9 bit al secondo = 8 microsecondi

Umit = (L/R) / (RTT + (L/R)) = .0008 / 30.008 = 0.00027

\*Le prestazioni di questo protocollo fanne ride\*

Con pipelining invece, in questo caso 3-packet pipelining:

Umit = (3L/R) / (RTT + (L/R)) = .0024 / 30.008 = 0.00081

\*81% rispetto al 27% di prima, buono\*

-----------------------------------------------------------------------------------

EstimatedRTT = (1- a)\*EstimatedRTT + a\*SampleRTT

TimeoutInterval = EstimatedRTT + 4\*DevRTT

DevRTT = (1-0.25)\*DevRTT + 0.25\*|SampleRTT-EstimatedRTT| Ci sarebbe la Beta al posto di 0.25 ma solitamente è 0.25

La velocità trasmissiva TCP varia tra:

W/(2 × RTT) e W/RTT

Se l’ampiezza della finestra vale w byte la frequenza trasmissiva è approssimativamente

w/RTT. TCP va alla ricerca di banda aggiuntiva aumentando w di 1

MSS a ogni RTT, fino al verificarsi di un evento di perdita. Detto W il valore di w

quando si verifica tale evento e assumendo che RTT e W siano approssimativamente

costanti per la durata della connessione.

*ESAMI PASSATI*

1. Gli indirizzi IPv4 sono a lunghezza variabile. FALSO

2. Il protocollo TCP durante la fase di Slow Start incrementa la finestra di un segmento ogni round trip time. FALSO

3. L’architettura SDN prevede che non ci sia alcun controller privilegiando un sistema completamente distribuito. FALSO?

SDN suggerisce di centralizzare l'intelligenza di rete in un componente separato, disassociando il processo di forwarding dei pacchetti (Data Plane) da quello di routing (Control Plane). Il Control Plane è costituito da uno o più controller che sono considerati il cervello della rete.

4. La rete internet ha un'architettura non gerarchica in termini di ISP. FALSO

5. HTTP persistente apre una connessione TCP per ogni oggetto nella pagina HTML e per ottimizzare non fa pipelining. FALSO

tiene una connessione tcp condivisa, la non persistente ne apre una per ogni oggetto

6. Il protocollo SMTP è utilizzato solo per leggere la posta da smartphone. FALSO

7. Il protocollo POP è utilizzato per spedire e leggere la posta solo da desktop. FALSO più dispositivi

8. Il sistema DNS è un database centralizzato usato dalle reti Content delivery networks per ottimizzare la distribuzione di

contenuti. FALSO

9. Gli algoritmi di congestion control in TCP sono chiamati AIMD perché usano AI nell definire la finestra di congestione. VERO

Durante la fase AIMD si ha un incremento additivo lineare del valore di cwnd di 1 MSS ogni RTT. Ciò si può ottenere attraverso l'incremento da parte del mittente della propria cwnd di una quantità di byte pari a: MSS\*(MSS/cwnd)

ogni volta che giunge un nuovo riscontro. Tale comportamento viene detto Additive Increase. Il termine Multiplicative Decrease si riferisce al comportamento messo in atto dal mittente alla ricezione di tre ACK duplicati consecutivi (con lo stesso numero di riscontro): la variante TCP Reno, in questa circostanza, imposta il valore di SSTRESH a cwnd/2 e assegna a cwnd tale valore incrementato di 3 MSS. Se la congestione diventa eccessiva, uno o più buffer dei router lungo il percorso vanno in overflow, causando l'eliminazione di un datagramma IP che contiene un segmento TCP. Rilevato questo evento allo scadere di un timeout di ritrasmissione, TCP reagisce dimezzando il valore di ssthresh e reimpostando cwnd alla dimensione di un segmento, tornando quindi nella fase di slow start.

10. Il protocollo IPv6 consente la frammentazione dei pacchetti ma effettua il recupero hop-by-hop in contrasto con IPv4. FALSO

non consente la frammentazione

11. In IPv6 la perdita di un frammento richiede la trasmissione del solo frammento perso. FALSO

richiede la ritrasmissione dell’intero datagramma

12. In TCP il raggiungimento dello “slow start threshold” determina lo stato di “congestion-avoidance” per la connessione. La crescita della “congestion window” passa da esponenziale a logaritmica. FALSO

b1. Gli indirizzi IPv4 sono a lunghezza fissa di 128 bit FALSO sono a 32 bit

b2. Il protocollo UDP durante la fase di. Slow Start raddoppia la congestion window ad ogni round trip time. FALSO

b3. L’architettura DNS prevede che ci sia un controller logico per il control plane. FALSO

b4. La rete Internet usa solo canali sicuri SSL e niente altro. FALSO

b5. HTTP persistente Apre una connessione TCP per ogni oggetto della pagina html FALSO

b6. Il protocollo SMTP è utilizzato spedire la posta tra due mail client. VERO

b7. Il protocollo FTP è utilizzato per spedire e leggere la posta solo da smartphone. FALSO

b8. Il sistema DHCP è un database distribuito usato solo dalle reti di smartphones per ottimizzare la distribuzione degli IP. FALSO

b9. Gli algoritmi di Congestion control in UDP sono chiamati AIMD perché performano meglio nel definire il rate ottimo. FALSO

gli udp non implementano algoritmi di controllo della congestione

b10. Il protocollo IPv4 manda solo pacchetti criptati per sicurezza. FALSO?

b11. In IPv4 la perdita di un pacchetto richiede la ritrasmissione di tutti i K frammenti del pacchetto perso. VERO

b12. In UDP-NewReno la perdita di un pacchetto determina la fase di slow-start. FALSO? esiste?

**Selezionare la risposta corretta.**

1) Il Protocollo UDP Reno Implementa:

(\_\_) Un Algoritmo di Congestion Control Additive Increase Multiplicative decrease

(\_\_) Un algoritmo di controllo della congestione sempre esponenziale

(\_\_) Un algoritmo di slow-start lineare nello spazio della finestra di congestione

(\_\_) Un protocollo di Forwarding Link State Gerarchico

(\_\_) nessuna delle precedenti risposte è corretta

2) L’algoritmo di Slow Start:

(\_\_) trova la distanza minima tra due nodi negli algoritmi di routing Link State con una topologia nota a priori.

(\_\_) limita il flip-flop delle performance in TCP

(\_\_) definisce il costo della congestione in condizioni Ottimali

(\_\_) Esplora la banda disponibile per una connessione TCP in fase iniziale

(\_X\_) nessuna delle precedenti risposte è corretta

b2) L’algoritmo di Slow Start:

(\_\_) trova la distanza minima tra due nodi negli algoritmi di routing Link State con una topologia nota a priori.

(\_\_) limita il flip-flop delle performance in TCP

(\_\_) definisce il costo della congestione in condizioni Ottimali

(\_\_) Esplora la banda disponibile all’inizio di una connessione TCP attraverso la crescita esponenziale della finestra.

(\_X\_) nessuna delle precedenti risposte è corretta

3) Si consideri Explicit Congestion Notification (ECN) :

(\_\_) Definisce la finestra media della connessione in funzione del RTT

(\_\_) Definisce la stima fatta dal trasmittente del round time-out time

(\_X\_) Fornisce assistenza da parte dei Routers per la gestione della congestione

(\_\_) Nessuna delle precedenti risposte è corretta

b1) Il Protocollo TCP Implementa:

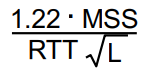
(\_\_) Un Algoritmo di Forwarding Distance Vector

(\_X?\_) Un algoritmo di controllo della congestione sempre esponenziale

(\_X?lineare\_) Un algoritmo di slow-start lineare nello spazio della finestra di congestione

(\_\_) Un protocollo di Forwarding Link State Gerarchico

(\_\_) nessuna delle precedenti risposte è corretta

b3) Data una connessione TCP la formula:  dove: L= Loss Probability; MSS=Maximum Segment Size.

(\_\_) Definisce la finestra media della connessione in funzione del RTT

(\_\_) Definisce la stima fatta dal trasmittente del round time-out time

(\_\_) Definisce la distanza di pipeline tra un pacchetto e il suo successivo necessaria nella fase di fast-retransmit-fast-recovery

(\_\_) Nessuna delle precedenti risposte è corretta