

Appunti di Fondamenti di telecomunicazioni T

Edoardo Carra'

May 13, 2022

0.1 Ethernet- IEE 802.03

Ethernet (and other old IEEE standard) layers

- Ethernet includes the physical and Data Link layer
 - The Data Link layer is in turn divided into the MAC and the LLC sub-layers

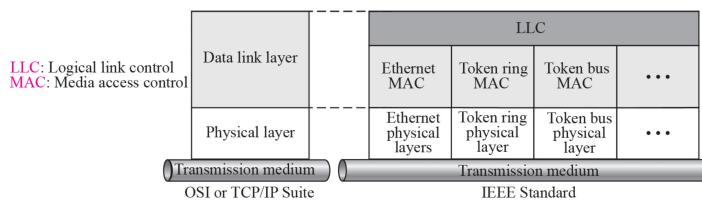


Figure from Forouzan, "TCP/IP Protocol Suite", McGraw Hill

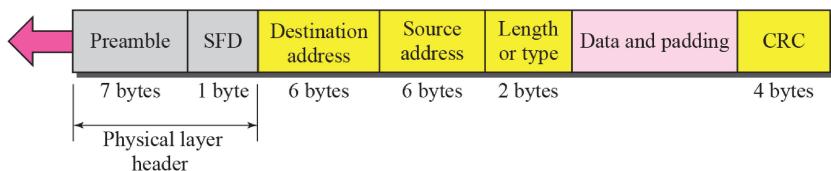
2

Ethernet è una famiglia di standard stabilito dalla IEEE come **802.3**, che comprendono due livelli dello stack: **Link layer** e **Physical layer**. Il Link Layer si occupa di gestire la comunicazione tra dispositivi, punto a punto ed è a sua volta è suddiviso in due: **LLC** e **MAC** layers. Il MAC layer si occupa invece di gestire l'accesso al mezzo fisico condiviso. Ethernet ha un accesso a contesa, cioè chi deve comunicare si contendere il canale di comunicazione. Questa modalità dal punto di vista storico risultò la scelta vincente contro il token ring di IBM, il quale accesso avveniva attraverso "token".

Ethernet Frame

Preamble: 56 bits of alternating 1s and 0s.

SFD: Start frame delimiter, flag (10101011)



TCP/IP Protocol Suite

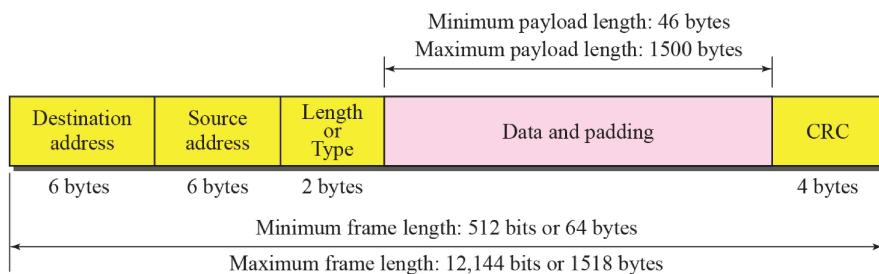
3

Il frame di livello 2, contiene il pacchetto che il livello 3 gli ha fornito e lo incapsula in un **frame**

header e in un **frame trailer**. Il pacchetto di livello superiore così encapsulato viene chiamato **payload**. Il frame è composto da:

1. I primi 8 byte servono al livello fisico per la sincronizzazione. Sono necessari per settarsi esattamente sulla stessa frequenza. I primi 7 bytes sono una successione di uno e zeri alternati, mentre l'ultimo bit dell'ottavo byte è un doppio 1 per segnalare l'inizio del pacchetto.
2. **dest and source address**: rappresentano il **MAC address** del destinatario e del mittente. L'indirizzo MAC è composto da 6 bytes, cioè 12 cifre esadecimali.
3. **Length or type**: Indica il tipo di protocollo di livello superiore encapsulato nel frame (es IPv4 o IPv6).
4. **Data and padding**: I dati encapsulati nel frame, con l'aggiunta di un padding nel caso i dati non raggiungessero la dimensione minima.
5. **CRC**: codice a correzione d'errore.

Maximum and minimum lengths



TCP/IP Protocol Suite

4

Esiste una grandezza massima per il frame. Perché negli anni 80 la memoria della scheda di rete costava molto. Mentre è necessario impostare un limite minimo a causa dell'implementazione del protocollo CSMA-CD per la gestione dell'interferenza.

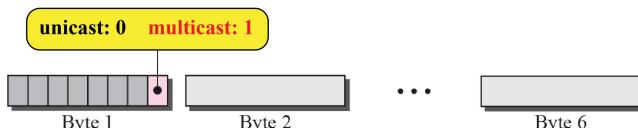
Address format and types

d: Hexadecimal digit

d₁d₂ : d₃d₄ : d₅d₆ : d₇d₈ : d₉d₁₀ : d₁₁d₁₂

6 bytes = 12 hexadecimal digits = 48 bits

- There are 3 address types:
 - Unicast (d₂ even)
 - Multicast (d₂ odd)
 - Broadcast (all "1", or FF:FF:FF:FF:FF:FF)



Figures from TCP/IP Protocol Suite

5

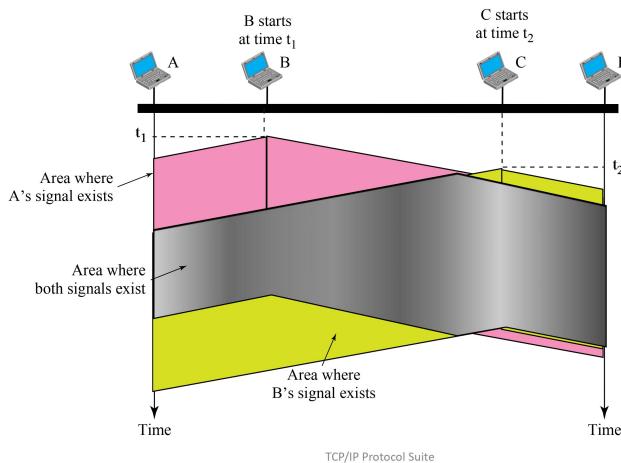
Per quanto riguarda la struttura dell'indirizzo MAC, i primi tre bytes indicano il produttore. Esistono tre tipi di indirizzi:

1. **Unicast**: seconda cifra esadecimale è pari.
2. **Multicast**: seconda cifra esadecimale è dispari. As with the unicast and broadcast addresses, the multicast IP address requires a corresponding multicast MAC address to actually deliver frames on a local network
3. **Broadcast**: **FF:FF:FF:FF:FF:FF**.

N.B. Un pacchetto viene sempre inviato sul canale condiviso, sia unicast che multicast. Quello che cambia è come questo viene usato: broadcast non viene mai scartato, unicast viene scartato in caso non ci sia un match tra mac address del destinatario e il mac del dispositivo.

L'indirizzo viene serializzato byte per byte, da destra verso sinistra, in modo tale che il primo bit che arriva faccia capire se si è in multicast oppure in unicast.

Space time model of a collision in CSMA



9

(C'è un cavo comune orizzontale e il tempo è in verticale)

Il segnale rosa parte al tempo t_1 e grazie al ritardo introdotto dal mezzo compie un triangolo sul grafico. Questo ritardo non permette ad un dispositivo qualunque della rete di capire se il canale sia effettivamente in uso. C in questo caso per esempio non si accorge che il canale è occupato dal segnale rosa e genera il suo segnale giallo, creando una collisione di dominio grigio. Il primo dispositivo che sta trasmettendo che rileva una collisione, cioè banalmente saranno presenti per un certo tempo più di 5V, invia un segnale di **jamming** sul canale per disturbare volontariamente il segnale, comunicando di fatto all'altro dispositivo che c'è stata interferenza. The jam signal or jamming signal is a signal that carries a 32-bit binary pattern sent by a data station to inform the other stations of the collision and that they must not transmit.

Minimum frame size

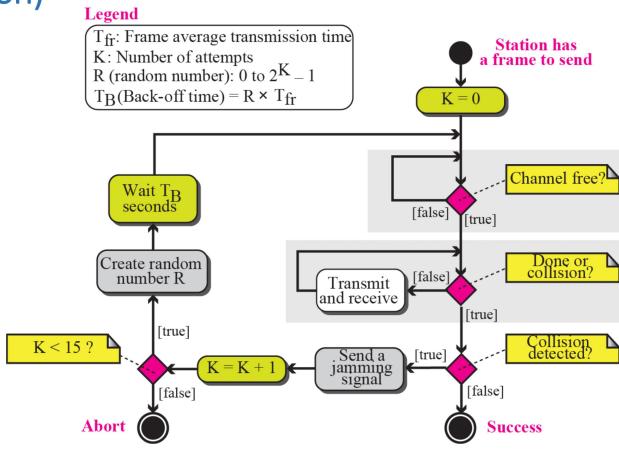
- In the “classic” Ethernet the maximum propagation time T_p is 25.6 μs
- The design of CSMA-CD requires that a collision be detected by the transmitter before the ending of transmission
- This implies that the frame transmission time must be twice T_p
 - $T_{fr} = 2 \times T_p = 51.2 \mu s$.
- The minimum size of the frame is thus $10 \text{ Mbps} \times 51.2 \mu s = 512 \text{ bits}$ or 64 bytes

11

L’idea è che il caso peggiore in cui si verifica interferenza sia dato da un tempo di propagazione massimo (che nel caso di ethernet è stabilito a priori) moltiplicato per due - **total trip time**. Questo perché il segnale deve propagarsi fino al destinatario che nel caso peggiore si trova ad un ritardo massimo dal mittente, e se quest’ultimo comincia a spedire esattamente nel momento in cui gli arriva il segnale, ci vorrà un altro tempo massimo prima che il segnale di jamming raggiunga di nuovo il mittente iniziale. Per questo la lunghezza minima deve essere tale che io stia ancora inviando prima che mi arrivi il pacchetto, altrimenti non rilevo l’interferenza.

Quindi il pacchetto deve essere tale da contenere un $N_{bit} = \text{frequenzabit} * 2 * t_{max} = 512 \text{ bit} = 64 \text{ bytes}$, di cui 46 bytes di payload (nel caso del 10Mbps).

CSMA-CD (Carrier Sensing Multiple Access-Collision Detection)



TCP/IP Protocol Suite

12

CSMA-CD(Carrier Sensing Multiple Access-Collision Detection) K è il contatore degli insuccessi. Per riprovare non ha senso farlo immediatamente, altrimenti ci sarebbe un’alta probabilità di finire in un’altra collisione. Si aspetta un tempo casuale R compreso tra 0 e $2^K - 1$, che poi va moltiplicato per il tempo di frame ottenendo così il **back-off time**. Più è alto il numero degli insuccessi più si aspetterà in media (tieni conto che due dispositivi, secondo questa politica, è probabile che vadano incontro ad una collisione: il 50% la prima volta, il 25 % la seconda ecc..). Come faccio a sapere se la comunicazione è andata male? Il destinatario può usare il **CRC**. Con questo metodo, però, solo il destinatario è a conoscenza dell’errore. Il protocollo ethernet non ha alcuna protezione contro questo genere di errori, infatti è compito dei protocolli di livello più alto gestire questo genere di errori.

Introdurre un ack può essere un ottimizzazione ma non un requisito del livello.

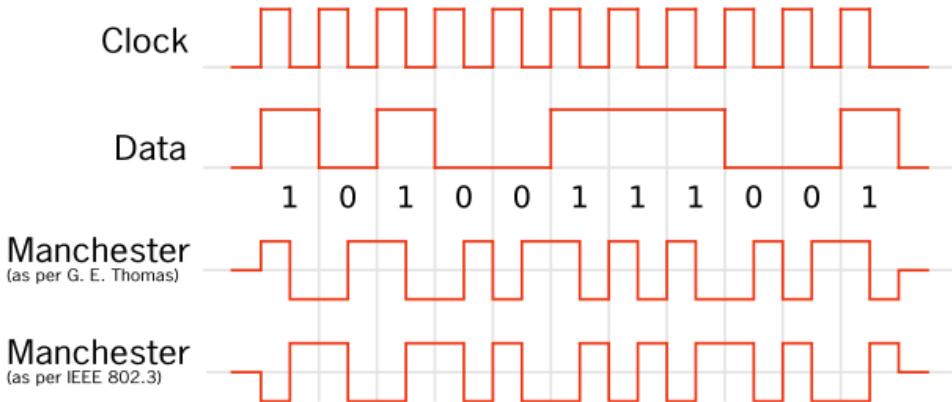
0.1.1 Hub e switch

L’hub è un dispositivo che lavora semplicemente sul livello 1, per questo si preferisce utilizzare lo switch. Questo, lavorando a livello più alto, gestisce anche l’interferenza. Questo perché lo switch

non si limita a ripetere il segnale, ma smista il segnale solamente al destinatario utilizzando un buffer di supporto. Si elimina il mezzo condiviso, i cavi terminano negli switch. Il **dominio di collisione** è limitato al ramo connesso allo switch, i domini di collisione sono N tanti quanti sono i dispositivi connessi alla rete. Quindi quello che succede è che un dispositivo invia un messaggio, lo switch capisce chi è il destinatario, e poi lo inserisce nel buffer del destinatario. Si elimina la necessità di utilizzare CSMA-CD(in realtà non è proprio così), abbiamo cambiato il protocollo di accesso al mezzo condiviso. È possibile anche attuare una gestione del controllo di flusso più efficiente. Si regola la velocità di trasmissione in base alla capacità di ricezione del ricevitore.

Come fa lo switch a sapere dove si trova il destinatario del pacchetto? Si parte da una tabella vuota, che si riempie automaticamente man mano che arriva a conoscere gli indirizzi delle varie porte a cui invia/riceve frame.

0.2 Codifica Manchester



Manchester coding is a special case of binary phase-shift keying (**BPSK**), where the data controls the phase of a square wave carrier whose frequency is the data rate. Manchester code ensures frequent line voltage transitions, directly proportional to the clock rate; this helps clock recovery. Manchester code always has a transition at the middle of each bit period and may (depending on the information to be transmitted) have a transition at the start of the period also. The direction of the mid-bit transition indicates the data. Transitions at the period boundaries do not carry information. They exist only to place the signal in the correct state to allow the mid-bit transition. The existence of guaranteed transitions allows the signal to be self-clocking, and also allows the receiver to align correctly; the receiver can identify if it is misaligned by half a bit period, as there will no longer always be a transition during each bit period. The price of these benefits is a doubling of the bandwidth requirement compared to simpler NRZ coding schemes.

Original data		Clock	Manchester value
0	XOR	0	0
		1	1
	\oplus	=	
		0	1
1		1	0

0.3 Altri standard

10 Mbit/s sono pochi, quindi esistono altri standard come ethernet a 100Mbit/s, 1Gbit/s ecc. Con l'aumentare dello standard l'unico dispositivo di collegamento permesso è lo switch. L'obiettivo però è mantenere la banda costante anche se aumenta di un fattore 10^n la frequenza di simbolo. I trade-off tra cui poter scegliere sono:

1. Aumentare lo standard dei cavi (cat-5 ecc...), di fatto aumenta la "frequenza di taglio" del cavo;
2. Dimezzare la banda utilizzando una NRZ al posto della Manchester. Onde evitare problemi di sincronizzazione però, è necessario utilizzare una codifica che garantisca un certo numero di transizioni per ogni simbolo, in modo da poter permettere al ricevente di sincronizzarsi. Per questo si utilizza la codifica 4b5b, la quale però aumenta del 25% la banda utilizzata rispetto alla NRZ (simboli da 5 bit per rappresentare 4 bit).

Table 2.2.: 4B/5B encoding.

4-bit Data Symbol	5-bit Code
0000	11110
0001	01001
0010	10100
0011	10101
0100	01010
0101	01011
0110	01110
0111	01111
1000	10010
1001	10011
1010	10110
1011	10111
1100	11010
1101	11011
1110	11100
1111	11101

3. Introduzione del **flow control**

4. Possibile utilizzare la modalità **full-duplex** grazie al secondo doppino presente nel cavo oppure utilizzare fino a 2 canali (sono presenti in totale 4 doppini nel caso permettendo quindi l'utilizzo di due canali in ingresso e 2 in uscita). il 10 Mbit/s usa un doppino, il 100Mbit/s ne usa due e la 1 Gbit/s ne usa 4. Nota che l'introduzione del doppio canale permette di eliminare la necessità di utilizzare il CSMA-CD.
5. Viene introdotta anche **l'auto negoziazione** per permettere a due dispositivi di trovare una tecnologia comune con la quale comunicare (tutte le schede di rete devono poter andare almeno a 10 Mbit/s).
6. Per assicurare che l'algoritmo CSMA/CD continui a funzionare, la relazione tra la dimensione minima del frame e la lunghezza massima del cavo deve essere mantenuta in proporzione alla velocità della rete che passa da 10 Mbps a 100 Mbps. Quindi o va aumentata la dimensione minima di frame di 64 byte o dovrà diminuire proporzionalmente la lunghezza massima di cavo di 2500 m. La scelta più facile è stata di abbassare la distanza tra ogni coppia di stazioni di un fattore 10, poiché un hub con un cavo da 100 metri rientra già in questo nuovo limite massimo.

Ethernet 10 Mbit/s (or Classic)

- Several variants depending on the physical medium
 - Coaxial cable («thick» or «thin») or
 - 2 pairs of UTP Cat. 3 (used for telephone lines)
 - Fiber
- 10Base-T
 - Signal
 - 10 Msymbol/s, Manchester encoding (about 20 MHz BW)

	10Base5	10Base2	10Base-T	10Base-F
Medium	Thick coax	Thin coax	2 UTP	2 fibers
Max hop length	500m	185m	100m	2000m

The "100" in the media type designation refers to the transmission speed of 100 Mbit/s, while the "BASE" refers to baseband signaling. The letter following the dash ("T" or "F") refers to the physical medium that carries the signal (twisted pair or fiber, respectively), while the last character ("X", "4", etc.) refers to the line code method used. Fast Ethernet is sometimes referred to as **100BASE-X**, where "X" is a placeholder for the FX and TX variants.

In telecommunications and signal processing, **baseband** is the range of frequencies occupied by a signal that has not been modulated to higher frequencies

Con la **1000 BASE-T** vengono introdotti dei nuovi requisiti:

1. Auto-negoziazione.
2. Each 1000BASE-T network segment is recommended to be a maximum length of 100 meters (330 feet), and must use Category 5 cable or better (including Cat 5e and Cat 6).
3. Bit Error Rate of less than or equal to 10^{-10}

I cavi smettono di essere unidirezionali. Poiché non si verifica contesa, non si utilizza il protocollo CSMA/CD, perciò la lunghezza massima del cavo dipende dall'energia del segnale e non più dal tempo imposto dal CSMA/CD. Le NIC capiscono automaticamente quale verso usare per ogni doppino presente nel cavo. Quando si inizia ad andare a certe velocità, si bufferizzano i pacchetti affinché non si sovraccarichi la CPU con troppe richieste.

The three-bit symbols are then mapped to voltage levels which vary continuously during transmission.(1000BASE-T) An example mapping is as follows:

Symbol	000	001	010	011	100	101	110	111
Line signal level	0	+1	+2	-1	0	+1	-2	-1

Una tabella utile per quanto riguarda l'interpretazione degli standard:

Ethernet 1 Gbit/s variants

Name	Medium	Specified distance
1000BASE-CX	Twinaxial cabling	25 meters
1000BASE-SX	Multi-mode fiber	220 to 550 meters dependent on fiber diameter and bandwidth
1000BASE-LX	Multi-mode fiber	550 meters
1000BASE-LX	Single-mode fiber	5 km[4]
1000BASE-LX10	Single-mode fiber using 1,310 nm wavelength	10 km
1000BASE-ZX	Single-mode fiber at 1,550 nm wavelength	~ 70 km
1000BASE-BX10	Single-mode fiber, over single-strand fiber: 1,490 nm downstream 1,310 nm upstream	10 km
1000BASE-T	Twisted-pair cabling (Cat-5, Cat-5e, Cat-6, or Cat-7)	100 meters
1000BASE-TX	Twisted-pair cabling (Cat-6, Cat-7)	100 meters

http://en.wikipedia.org/wiki/Gigabit_Ethernet

1 WIFI - IEEE 802.11

Il principale insieme di standard per le LAN Wireless è IEEE 802.11, meglio conosciuto come WiFi (Wireless Fidelity). Tutti i protocolli 802, inclusi 802.11 e Ethernet, presentano una certa somiglianza nella propria struttura. Il livello fisico corrisponde circa al livello fisico OSI, ma il livello data link nei protocolli 802 è diviso in due o più sotto livelli.

The 802.11 Protocol Stack

- The LLC sublayer and the format of MAC addresses are the same as other 802 standards (e.g. 802.3 Ethernet)
- MAC sublayer is different

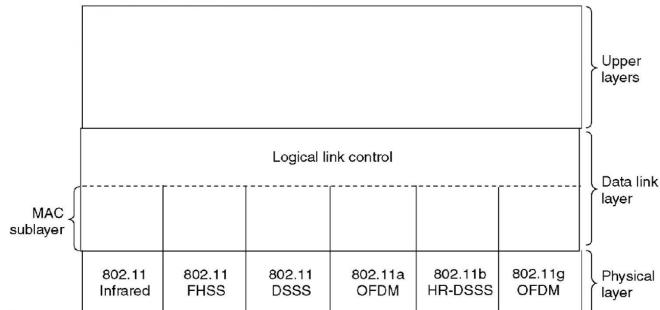


Figure from Tanenbaum, Computer Networks, McGraw Hill

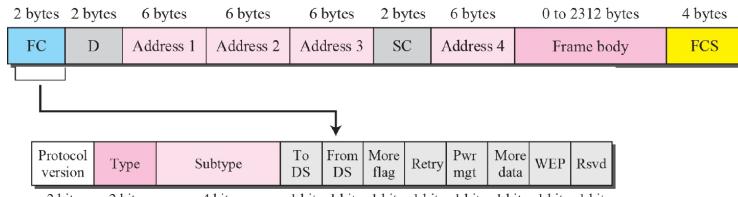
3

In 802.11 il sottolivello MAC determina com'è allocato il canale, cioè a chi tocca trasmettere. Sopra c'è il sottolivello LLC (logical link control) il cui compito è nascondere le differenze tra i differenti 802 e renderli indistinguibili al livello di rete; oggi LLC è un livello di raccordo che identifica il protocollo (ad esempio IP) trasportato all'interno di un frame 802.11. (riguardare)

Le reti 802.11 possono essere utilizzate in due modalità: nella modalità con **infrastruttura** (infrastructure BSS) (**BSS = Basic Service Set**) ogni client è associato a un **AP** (access point) connesso a sua volta all'altra rete. Il client spedisce e riceve i propri pacchetti attraverso l'AP. Alcuni access point possono essere connessi insieme, in genere attraverso una rete cablata che prende il nome di sistema di distribuzione (distribution system). L'altra modalità prende il nome di rete **ad hoc**. Questa modalità consiste in una collezione di computer associati tra loro che possono spedirsi direttamente i frame. Non esiste alcun access point.

N.B. l'access point è anch'esso parte della BSS.

Frame format (early versions)



- D (Duration) contains the NAV (channel booking duration)
- Types: data (as in figure), control (CTS, RTS, ACK, etc.), management (beacon, association, authentication, etc.)
- There are at most four MAC addresses (same format as 802.3).
 - In "ad hoc", or before association, only two (for the 2 STAs)
 - In "Infrastructure", usually three (2 for STAs, 1 for the intermediate AP).
 - Four only in special cases (not well specified by the standard), e.g. when a data packet goes through one AP and one repeater.
To DS=1 & From DS=1
 - Note that even one AP isolated is considered as a Distribution System
- The format has been augmented in recent versions

Figure from Forouzan, "TCP/IP Protocol Suite, McGraw Hill"

7

Lo standard 802.11 definisce tre diverse classi di frame: dati, controllo e gestione. Ognuna ha

un'intestazione composta da una varietà di campi utilizzati all'interno del sottolivello MAC.

1. **Frame control:** lungo 2 byte è costituito da 11 sotto campi. Il primo di questi è il **protocol version**, impostato a 00. È lì per permettere future versioni di 802.11. Quindi ci sono i campi **type** che può valere dati, controllo e gestione e **subtype**, ad esempio RTS o CTS; per un frame di dati regolare sono fissati a 100000_2 . ToDS and FromDS: Each is one bit in size. They indicate whether a data frame is headed for a distribution system. Control and management frames set these values to zero. Il bit **more fragments** significa che seguiranno più frammenti. Il bit **retry** indica la ritrasmissione di un frame spedito in precedenza. Il bit **power management** indica che il mittente sta andando in modalità power-save. Il bit more data indica che il mittente ha altri dati per il ricevente. Il bit **protected frame** indica che il corpo del frame è stato criptato per sicurezza (per esempio WPA2). Infine, il bit **order** indica al ricevente che il livello superiore si aspetta che la sequenza di frame arrivi rigorosamente in ordine.

FIGURE 3.20 802.11 MAC addressing

Protocol Version	Type	Subtype	To DS	From DS	More Frag	Retry	Power Mgmt	More Data	Prot. Frame	Order
Frame Control field										
To DS	From DS		Address 1	Address 2	Address 3	Address 4				
0	0		RA = DA	TA = SA	BSSID	N/A				
0	1		RA = DA	TA = BSSID	SA	N/A				
1	0		RA = BSSID	TA = SA	DA	N/A				
1	1		RA	TA	DA	SA				

- SA = MAC address of the original sender (wired or wireless)
- DA = MAC address of the final destination (wired or wireless)
- TA = MAC address of the transmitting 802.11 radio
- RA = MAC address of the receiving 802.11 radio
- BSSID = L2 identifier of the basic service set (BSS)

2 Livello di trasporto

I principali protocolli di trasporto sono **UDP** e **TCP**. UDP si caratterizza per essere connection less (ogni livello può essere connection less oppure connection oriented ricorda). TCP invece è orientato alla connessione invece. In generale vale che ogni livello offre al livello superiore dei servizi, questi servizi sono la composizione dei servizi offerti dal livello di sotto con l'aggiunta di qualcosa. (il modello ricorda l'estensione delle classi) In questo caso UDP non aggiunge particolari estensioni al servizio offerto da IP.

Trasporto

- UDP
 - Non orientato alla connessione (Connectionless)
 - datagram
 - Non affidabile (Unreliable)
- TCP (Transmission Control Protocol)
 - Orientato alla connessione
 - Apertura connessione (Establishment)
 - Trasferimento dati
 - Chiusura connessione (Closing)
 - Affidabile (Reliable)

2.1 UDP- USER DATAGRAM PROTOCOL

Un datagram (oppure segmento) UDP viene incapsulato in un pacchetto IP aggiungendo poche funzionalità ai servizi offerti, la lunghezza del payload può raggiungere un massimo di 65.515 byte:

- multiplexing grazie alle porte UDP: attraverso gli indirizzi di livello 4 posso discriminare quale applicazione è il destinatario su un determinato dispositivo.
- error detection con checksum. il checksum viene controllato una sola volta a differenza del checksum dell'IP. Il checksum può essere disabilitato.
- Unicast, multicast e broadcast. Questo viene fatto utilizzando i servizi offerti dal livello sottostante.

RFC - request for comments: A Request for Comments is a publication in a series, from the principal technical development and standards-setting bodies for the Internet, most prominently the Internet Engineering Task Force. RFC 768 è quello dell'UDP. <https://datatracker.ietf.org/doc/html/rfc768>

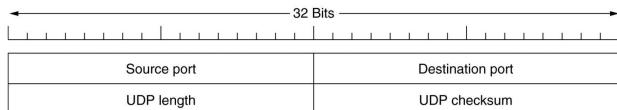
UDP

- **Pro**
 - Semplicissimo, consegna "veloce" dei dati
- **Contro**
 - Non "affidabile"
- **Uso**
 - Molti protocolli Internet di segnalazione usano UDP, fra i quali:
 - [Domain Name System](#) (DNS)
 - [Dynamic Host Configuration Protocol](#) (DHCP)
 - [Simple Network Management Protocol](#) (SNMP)
 - [Routing Information Protocol](#) (RIP)
 - Non usato dai protocolli di routing [OSPF](#) (IP) e [BGP](#) (TCP)
 - Spesso usato per Voice over IP o streaming da protocolli superiori, dove le specifiche sul ritardo sono stringenti e sono invece tollerabili perdite di segmenti o consegna non ordinata
 - [RTP \(Real time Transport Protocol\)](#)

I principali utilizzi di questo protocollo sono per tutti quei protocolli di livello più alto che richiedono di instaurare una connessione del tipo query-response. (e quindi l'overhead dell'instaurarsi della connessione non è necessario).

Introduction to UDP

The UDP header.



Parentesi sul checksum: a differenza del protocollo IP, il checksum viene calcolato su header, dati(primo protocollo end to end, ha senso) e pseudo header perché conoscendo ip destinatario e mittente all'inizio della comunicazione, posso comprenderli nel calcolo del checksum. Questo genera problemi di inconsistenza perché si basa sulla conoscenza dell'header del livello sottostante, che può ipv4 oppure ipv6. Un altro problema è dato dal NAT: questo cambia gli indirizzi su cui il checksum era stato calcolato. Quindi il nuovo header avrà un checksum errato che va sostituito con un checksum aggiornato con i nuovi valori di IP:porta aggiunti dal NAT.

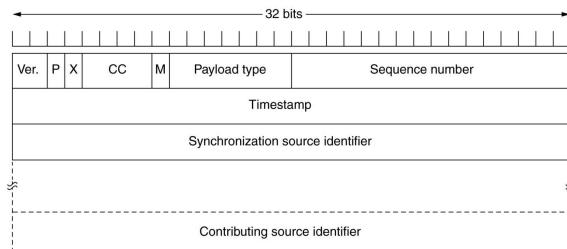
2.2 RTP

È un protocollo di livello trasporto, non affidabile, che si basa sulla trasmissione via UDP che permette la gestione ad hoc per il traffico multimediale. Solitamente i protocolli stanno nel kernel, mentre RTP è presente nella user space. Invece di implementare totalmente da capo, mi baso sull'UDP aggiungendo delle funzionalità in più. Il protocollo è di livello 4+. Di fatto è l'applicazione che scrive questo payload.

L'header è composto da un **CC**, cioè l'identificativo di flusso. Il **payload type** serve per decodificare il contenuto multimediale fornendo tipo di contenuto, bit rate ecc... Il **sequence number** viene incrementato ogni volta che viene inviato un pacchetto. Il **timestamp**: il primo campione

del payload quando è stato inviato? A cosa servono entrambi? Servono per capire quale sia l'error rate.

The Real-Time Transport Protocol (2)



Avere sequence number e timestamp permette di calcolare l'**error rate** e permette di capire quando il dato deve essere riprodotto nel tempo (i pacchetti arrivano a intervalli non regolari). Il timestamp serve inoltre perché il flusso di dati può cambiare in base alla codifica (mp3 ha coding rate variabile, riguarda MP3!!) ecc... Infine il timestamp serve per sincronizzare e per eliminare la ridondanza.(se un'immagine è ferma in un video non ha senso inviare tutti i frame, ma terrò l'immagine costante fino al prossimo timestamp)

Il campo **Synchronization source identifier** specifica a quale flusso appartiene il pacchetto e rappresenta l'informazione utilizzata per il multiplexing e il demultiplexing di più flussi dati in un singolo flusso di pacchetti UDP.

2.2.1 RTCP

I dati sono inviati con il protocollo RTP, però per avere una retroazione della qualità si inviano dei segmenti TCP di controllo per avere una gestione della qualità della connessione. Per esempio viene informato a quanto ammonta il **jitter**, cioè la variazione di ritardo. Se il delay si riesce a compensare con il timestamp, il jitter si compensa solamente con un buffer. Però se il buffer è maggiore, il ritardo nella riproduzione è anch'esso maggiore, con una migliore gestione del jitter. Il feedback è fondamentale per la gestione della congestione.

2.3 TCP

TCP è stato progettato appositamente per fornire un flusso di byte affidabile end-to-end su una rete. È molto più complesso dell'UDP in quanto deve offrire un servizio orientato alla connessione affidabile, a partire dal servizio offerto da IP. I dati sono visti come un flusso di byte, anche se sono organizzati in pacchetti chiamati segmenti. Mittente e destinatario si chiamano **socket**. Le porte come in UDP sono lunghe 16 bit.

Il TCP, a differenza dell'UDP, non supporta unicast e multicast.

Se un pacchetto IP viene perso, è il protocollo TCP che prova a rispedire i pacchetti. I nodi intermedi, lavorando a livello 3, non fanno nulla a riguardo perché non immagazzinano i dati. ARQ è il protocollo per recuperare i pacchetti persi ed è esclusivamente host to host. Si preferisce fare un intero avanti e indietro piuttosto che à complicare la rete intermedia con l'aggiunta della gestione dei pacchetti persi. Questo viene fatto perché il ritardo non è proibitivo (nelle DTN questo non vale). Il recupero delle perdite viene quindi fatto a livello di sorgente nel caso entro un certo limite di tempo non arrivi l'ack di conferma.

La regolazione di flusso serve per la gestione della congestione e la capacità di ricezione del ricevente. Questo viene realizzato in modo decentralizzato, basato su feedback. Si evita quindi l'utilizzo del controllore centrale per spartire la banda tra i nodi di internet (è un bene).

Qual è la relazione tra TCP ports e UDP ports? Nessuna! Sono due protocolli diversi e quindi gli indirizzi sono diversi.

È il TCP che organizza i segmenti, indipendentemente dal livello applicazione e mantiene indipendente la scrittura e la lettura. Una connessione è individuata da $IP_{dest} : Port_{dest} - IP_{source} : Port_{source}$. La differenza principale della gestione dei segmenti tra UDP e TCP è che il livello applicazione con UDP può decidere la dimensione dei dati da inviare per ogni pacchetto, mentre nel TCP è il livello 4 che gestisce la formazione dei segmenti.

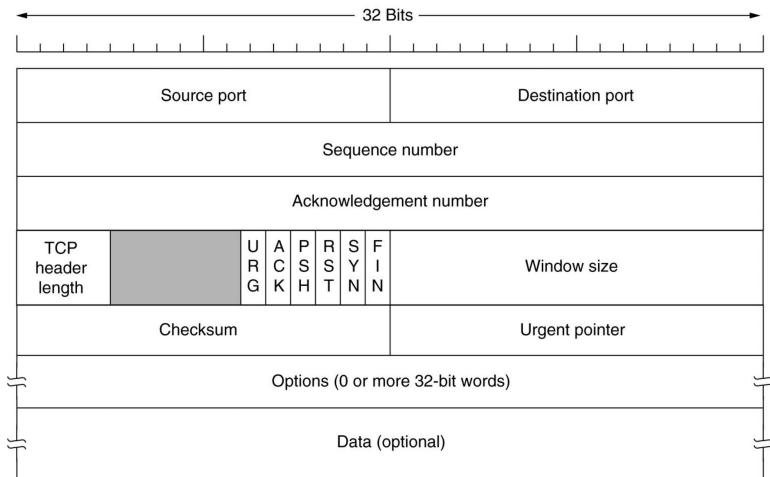
La connessione una volta stabilita, è bidirezionale. Ogni segmento deve essere confermato da un ACK che può essere in **piggybacked** nel caso il ricevente abbia qualcosa da dire. Qual è il vantaggio fondamentale di questa tecnica? La riduzione dell'overhead unendo l'informazione ad un segmento di controllo.

Ogni segmento è caratterizzato da un **sequence number** che indica il byte della sequenza del primo byte del payload. L'ack non segnala "è arrivato tutto fino a N", ma segnala "mi aspetto un SEQ=N+1" (di fatto gli ack sono cumulativi). Siccome TCP richiede l'ordine dei pacchetti in ricezione, se il destinatario non riceve il SEQ previsto, continua ad inviare un ack per ogni pacchetto della sequenza sbagliato, attuando il cosiddetto **ack duplicato**. (simile al polling nel wifi) Il dupack ci fornisce comunque un'informazione: i messaggi continuano ad arrivare. Gli ack hanno una dimensione di 20byte.

Nel caso la comunicazione sia asimmetrica, cioè la velocità in ricezione sia diversa da quella in spedizione, esistono delle implementazioni che prevedono l'invio di un ack ogni due segmenti.

Inoltre per la gestione del flusso, viene inserito il timestamp nel pacchetto TCP che verrà riportato anche nell'ack. Questo permette al mittente di calcolare il round trip time per fornire il feedback. TCP è un algoritmo a **retroazione** e si dice che RTT è l'elemento critico del TCP. (se intercorre molto tempo tra azione e reazione vado più piano)

The TCP Segment Header



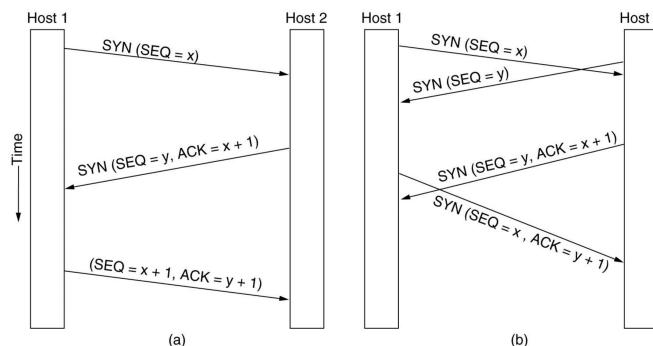
1. sequence number si evince che possono essere inviati fino a un max di 4 GB.
2. ack number di dimensioni omogenee a seq number.
3. TCP header: indica la dimensione in parole da 4 byte dell'header. L'header deve essere quindi multiplo di 32 bit (vale anche per udp e ip).
4. window size (receiver window): indica quanto spazio in byte è ancora presente nel buffer del ricevente.
5. flags:
 - **spazio nero**: sono presenti due flag **ECE** e **CWR**. Siccome in ip è presente il flag del QoS che viene ripetuto fino al destinatario in caso di congestione della rete, si utilizza ECN-Echo per comunicarlo al mittente tramite TCP. ECE indica al mittente

che il destinatario ha ricevuto un pacchetto IP con congestione, mentre CWR indica al destinatario che il mittente ha ricevuto la segnalazione di congestione.

- **ack**: indica che l'ack number ha un significato.
 - **urg**: il campo urgent ha significato.
 - **psh**: asks to push the buffered data to the receiving application.
 - **rst**: reset the connection.
 - **syn**: iniziare la connessione.
 - **fin**: terminare la connessione.
6. **urgent pointer**: indica che è appena stato inviato l'ultimo byte di una sequenza urgente.
 7. **checksum**: anche qui calcolata sullo pseudoheader.

2.3.1 Three way handshake

TCP Connection Establishment

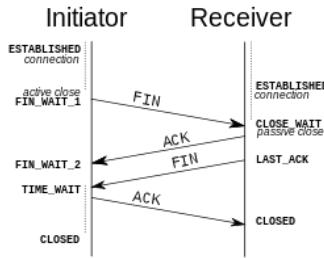


(a) TCP connection establishment in the normal case.
(b) Call collision.

1. host 1 vuole iniziare la comunicazione: $\text{syn}=1$ e $\text{seq}=x=\text{random}(0,2^{32}-1)->$. **pacchetto "syn"**.
2. host 2 rimanda un **pacchetto syn-ack**. ack coerente al canale di comunicazione tra host 1 e host 2. (attenzione non è $+1$ ma è $\text{len(payload)}+1$) Mentre syn è il numero di sequenza di host 2.
3. il terzo e ultimo passo è un **pacchetto ack** da parte di host 1 senza syn.

Il secondo caso rappresenta una possibile collisione.

2.3.2 Four way handshake



Come funziona la chiusura della connessione? Se un terminale fa una close NON chiude la connessione, ma chiude solamente il proprio canale verso il secondo host. La chiusura può avvenire in due modi:

4 way handshake:

1. Host 1 invia un FIN.
2. Host 2 invia ACK. Host1 non può più comunicare.
3. Host 2 invia FIN.
4. Host 1 invia ACK. Host 2 non può più comunicare e si chiude la comunicazione

Oppure la chiusura avviene anche per 3 way handshake:

1. Host 1 invia un FIN.
2. Host 2 invia un FIN-ACK e host1 non può più comunicare.
3. host 1 invia ACK e host 2 non può più comunicare e si chiude la comunicazione.

Il timeout a cosa serve? serve perché l'ack potrebbe non arrivare all'host che vuole chiudere la connessione. Se all'host non arriva l'ack, non può sapere se è realmente arrivato il FIN, quindi lo rimanda. Questo si ripete per 5 o 6 volte prima di terminare definitivamente.

TCP Connection Management Modeling (2)

TCP connection management finite state machine. The heavy solid line is the normal path for a client. The heavy dashed line is the normal path for a server. The light lines are unusual events. Each transition is labeled by the event causing it and the action resulting from it, separated by a slash.

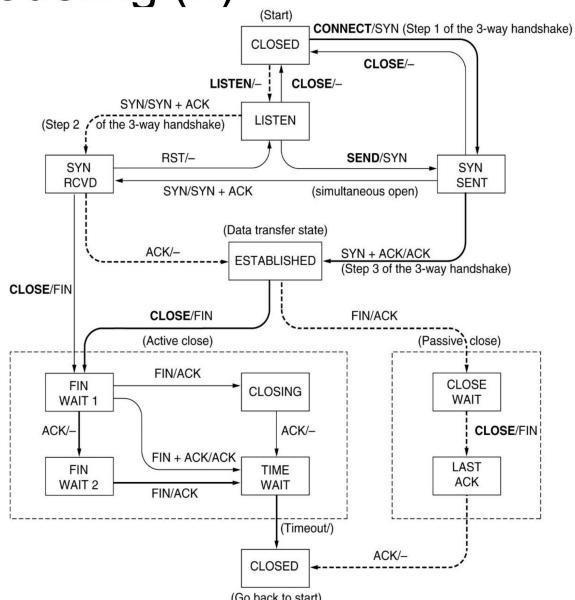
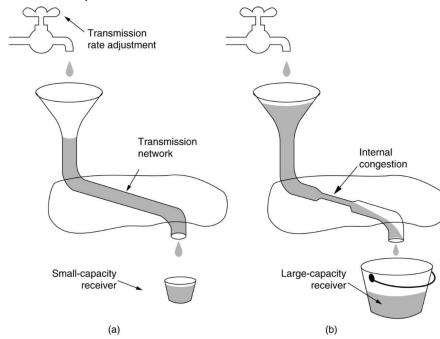


Diagramma degli stati. righe spesse: casi comuni, righe sottili: casi speciali. Lato server trattato. Le transizioni sono contrassegnate da causa/effetto.

2.3.3 Controllo del flusso

TCP Flow & Congestion Control

- (a) A fast network feeding a low capacity receiver. (need of a flow control, rwnd window)
- (b) A slow network feeding a high-capacity receiver. (need of a congestion control, cwnd window)



1. **Flow control:** non spedire più velocemente di quanto che il ricevente può gestire
2. **Congestion control:** nel secondo caso il problema è tra i nodi della rete, la quale non riesce a gestire tutti i pacchetti che gli sono stati forniti. Il link in cui avviene la congestione si chiama bottleneck.

Finestre & velocità di Tx

- W (in segmenti) è il numero massimo di segmenti che possono essere spediti dopo l'ultimo confermato.
- W=1
 - Invio un segmento ed aspetto l'ACK prima di inviarne un altro. La Tx è pari ad 1 segmento per RTT, normalmente troppo bassa.
- W>1
 - Invio W segmenti, quindi alla ricezione del primo ACK ne invio un altro e così via, facendo scorrere la finestra (sliding window).
 - $Tx(\text{bit}/\text{s}) = W(\text{in bit})/\text{RTT}(\text{s})$
- Se voglio $Tx(\text{bit}/\text{s}) = \text{Banda disponibile}$, la finestra dovrà essere uguale al prodotto banda ritardo (BDP)
 - $W = B * \text{RTT}$

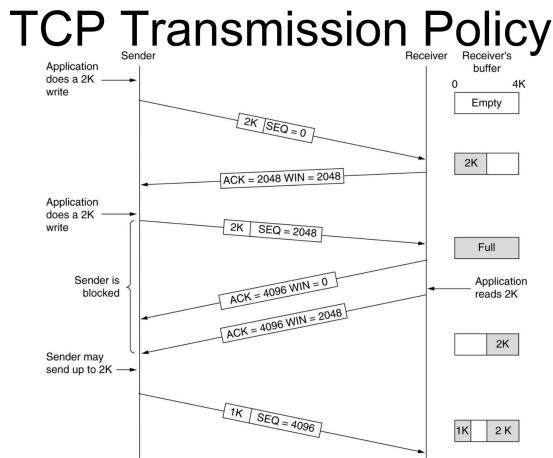
Per capire come funzionano i due algoritmi è necessario capire come funzionano le finestre (per comodità la grandezza della window è fornita in MAXPAYLOADLENGTH): - TCP uses a sliding window flow control protocol. In each TCP segment, the receiver specifies in the receive window field the amount of additionally received data (in bytes) that it is willing to buffer for the connection. The sending host can send only up to that amount of data before it must wait for an acknowledgement and receive window update from the receiving host. - La velocità di trasmissione è quindi pari a $Tx(\text{bit}/\text{s}) = W[\text{bit}]/\text{RTT}[\text{s}]$ importantissimo.

Data a disposizione una certa banda B, qual è la dimensione massima che può avere la finestra? $W=B*\text{RTT}$ (importantissimo)

Finestre & velocità di Tx

- W (in segmenti) è il numero massimo di segmenti che possono essere spediti dopo l'ultimo confermato.
- $W=1$
 - Invio un segmento ed aspetto l'ACK prima di inviarne un altro. La Tx è pari ad 1 segmento per RTT, normalmente troppo bassa.
- $W>1$
 - Invio W segmenti, quindi alla ricezione del primo ACK ne invio un altro e così via, facendo scorrere la finestra (sliding window).
 - $Tx(\text{bit/s}) = W(\text{in bit})/\text{RTT(s)}$
- Se voglio $Tx = \text{Banda disponibile}$, la finestra dovrà essere uguale al prodotto banda ritardo (BDP)
 - $W=B*\text{RTT}$

La finestra disponibile si calcola come $W = \min(congestion_{wnd}, receiver_{wnd})$ cioè come minimo tra la window fornita dalla congestione di rete e quella del buffer del destinatario. (Solitamente $W = congestion_{wnd}$). Se la $receiver_{wnd}$ è pari a 0, la trasmissione viene stoppata.



Advertised window (o rwnd) management in TCP.

Quando si libera la finestra, viene inviato un dup-ack per avvisare che la finestra non è più uguale zero. **importante** Se questo ack viene perso cosa succede? Nel TCP non è detto che si parli di continuo. In quel caso il sender invia un segmento di probe(prova) di lunghezza 1 byte.

La massima velocità di trasmissione dettata dal $receiver_{wnd}$ da **rwnd/RTT** (rwnd può essere al massimo di 2^{16} Byte, veramente poco, con 200ms ho una banda di massimo 512Mb/s). Questo problema è stato superato con un nuovo protocollo TCP che si mette d'accordo sulla window scale factor.

Un segmento è definito **perso** se:

1. Avviene l'**RTO(retransmission time out)**. Non ci sono N timer per ogni pacchetto, ma è presente un unico timer che conta dall'ultimo ack(non dup-ack). L'RTO è dinamico e dipende dall'RTT, il quale deve essere minore dell'RTO. È importante trovare un buon bilanciamento tra reattività e ridondanza. Solitamente vengono usati algoritmi di exponential back-off: se il tentativo di ritrasmissione fallisce, riprovo aspettando un tempo T random che può variare tra 1 e 2^n .

Perdite

- Si ritiene sia andato perso un segmento se:
 - scade l'RTO (Retransmission Time Out)
 - ovvero se non sono arrivati ACK freschi (non duplicati) per un periodo più lungo di RTO (RTO è calcolato dinamicamente sulla base del RTT)
 - Era l'unica condizione nelle prime versioni di TCP
 - Arrivano tre dupACK (Fast retransmit)
 - Sono generati alla ricezione dei segmenti successivi a quello mancante
 - Aggiunta in seguito (da TCP Tahoe)
 - In entrambi i casi viene ritrasmesso il primo segmento non confermato

2. (fast retransmit) Con il passare del tempo si è aggiunta un'altra condizione per velocizzare il riconoscimento delle perdite (aggiunta da TCP "Tahoe"). L'idea è che con molta probabilità se arrivano 3 dupAcks il pacchetto è perso. (anche se con ip non è assicurato l'ordine, nella realtà la maggior parte delle volte i pacchetti arrivano ordinati).

Se uno di questi due eventi accade si entra in modalità recovery. Essendo gli ack cumulativi, il primo non confermato indica il primo segmento perso di quelli inviati.

Congestion Control (& error recovery)

- Congestion Control Algorithms from [RFC 5681](#) (Reno)
 - This section defines the four congestion control algorithms:
 - slow start
 - congestion avoidance
 - fast retransmit and fast recovery
 - In some situations it may be beneficial for a TCP sender to be more conservative than the algorithms allow, however a TCP MUST NOT be more aggressive than the following algorithms.
 - Troppo conservativo per tratte con lunghi RTT o bande larghe!

2.3.4 Controllo della congestione

Esistono principalmente 4 algoritmi di controllo della congestione: /*parte mancante su cosa dice RFC rispetto ai vincoli*/

Slow start Aumento la finestra di uno per ogni ack ricevuto. L'idea è che il mittente non sa

quanto è congestionata una rete. Parte con uno, se va bene allora proviamo con due. Se due vanno bene ne provo 4, se me ne tornano indietro 3 aumento solo di tre e non di 4, quindi ne invierò 7.

Il delayed ack è un arma a doppio taglio: Se me ne arriva uno ogni due rallento la crescita dell'ack. Quindi un'altra tecnica potrebbe essere aumentare in base a quanto confermato nell'ack. C'è ovviamente una soglia.

slide 23 congestion avoidance qua invece aumento ogni volta che mi arrivano un numero di ack pari alla grandezza della finestra. Linux usa questo. (nella pratica c'è un contatore)

QUESTO AVVIENE SOLO NEL CASO IDEALE! LO CHIEDE PER VEDERE SE HAI CAPITO. Solo in condizioni ideali avviene un aumento/raddoppio della window size ogni RTT.

slide 25 grafico slow start all'inizio fino a threshold, poi congestion avoidance IN AI e poi appena c'è un RTO(time-out) riparto da 1. La soglia viene posta alla metà della finestra prima del timeout(non può quindi sempre raddoppiare in slowstart), se arriva un 3 dup-acks riparto dalla soglia, che viene sempre dimezzata in base alla window size prima del recovery.

se time-out: $wnd=1$ e $threshold=wnd_{old}/2$ se 3 dupack: $threshold=wnd_{old}/2$ e $wnd=threshold$ (questa modalità è proprio Reno e newReno) PARTE DA UNO LA WINDOWS size(lo chiede). Chiede spesso questa figura(disegnarla)

Perché reagisco in modo diverso? Perché se mi arriva un dupack significa che comunque stanno arrivando dei segmenti dall'altra parte. Se non mi arrivano nemmeno i dupack significa che c'è stata una mega congestione. Non è detto che la congestione si risolva però: i pacchetti come UDP non rilevano la congestione(al massimo rtcp un pochetto).

la fairness del TCP è che ipoteticamente si arriva a una divisione della rete senza il grande controllore. La divisione avviene tra pari. È la realizzazione del controllo decentralizzato della congestione. La figata è che non è detto che una connessione TCP utilizzi sempre banda. Se una connessione è attiva non è detto che utilizzi banda necessariamente! Un algoritmo di controllo centralizzato distribuirebbe la banda in modo equo, ma non è detto che questo sia il metodo migliore, la banda non è sempre utilizzata da una connessione TCP.

unfairness del TCP. Se RTT è molto diverso tra le connessioni, chi va più lenta aumenterà meno nel tempo rispetto a chi va più veloce. Es. RTT1=25ms RTT2=600ms, RTT1 ogni 25ms potrebbe avere un aumento di banda, mentre RTT2 ogni 600ms. Per questo le connessioni via satellite non vanno molto bene con TCP.

N.B. la dimensione è in max-length-segment

Inoltre il TCP non è in grado di riconoscere l'origine della congestione (bit errati ecc...)

slide 26 SACK: dupack che però dice anche quale è arrivato. Permette di capire quali pacchetti sono andati persi in un unico RTT invece che 3 RTT con il dup-ack. Conferma fino a un massimo di 4 blocchi che sono arrivati.

LINUX usa una variante di nome CUBIC. Quando mi invento una nuova variante devo sempre testare la fairness con gli altri protocolli.

Un'altra variante è Vegas: invece di controllare i pacchetti persi, controllo la variazione di RTT, se aumenta rallento. Ovvio che Vegas confronto a Reno è molto più lenta.

Hybla è per le connessioni via satellite

slide 28 parametri interessanti da aggiungere alle cose vecchie. slide 29 opzioni. maximum segment size: relativo al payload(di solito 1460 è il massimo che potrei avere senza causare frammentazione su ethernet)

RTT viene stimato con il timestamp. Invio nelle opzioni il timestamp del pacchetto, l'ack mi ritorna con il timestamp che mi conferma. È utile per capire se mi sta riconfermando il ritrasmesso oppure no.