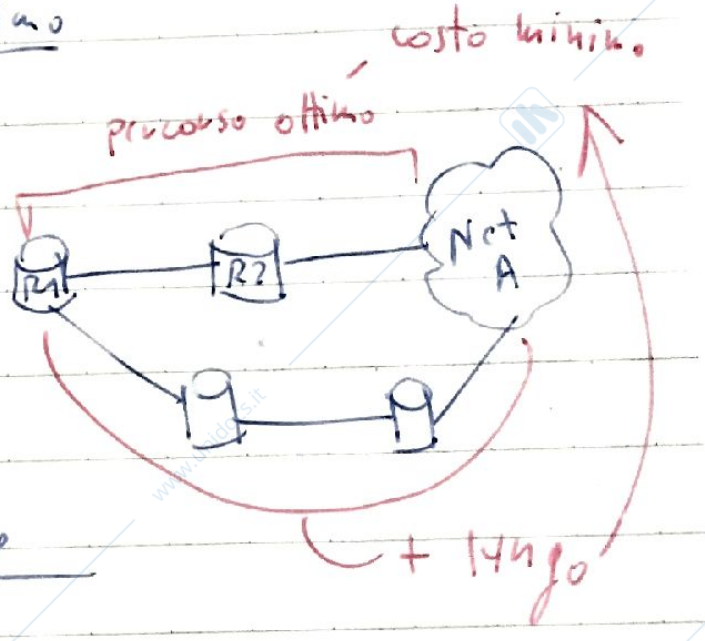
Algoritmi con tabella

Algoritmi diversi da Random, Flooding
Source Routing.

↓

Avere una tabella instradamento che questi algoritmi tengono conto del costo minimo

Consideriamo:



OS

TAB ROUTING R1:

RETE / h	NEXT HOP
Net A	IP R2

Si dovrà quindi definire una metrica di costo

↳ OS minor numero di hops (salti)

Ne esistono anche altre

↳ OS metrica basata sul ritardo ....

Mo Tu We Th Fr Sa Su

Date 10 · 12 · 19

le metriche possono essere combinazioni tra varie metriche

Routing statico

↳ compilate offline, (es. manualmente) dall'amministratore di rete

Routing dinamico

Tiene conto degli eventi che accadono ogni momento.

↳ es - guasti
- cambiamento della topologia di rete
- Carico di rete e congestione

Varie famiglie di algoritmi:

↳ Link state

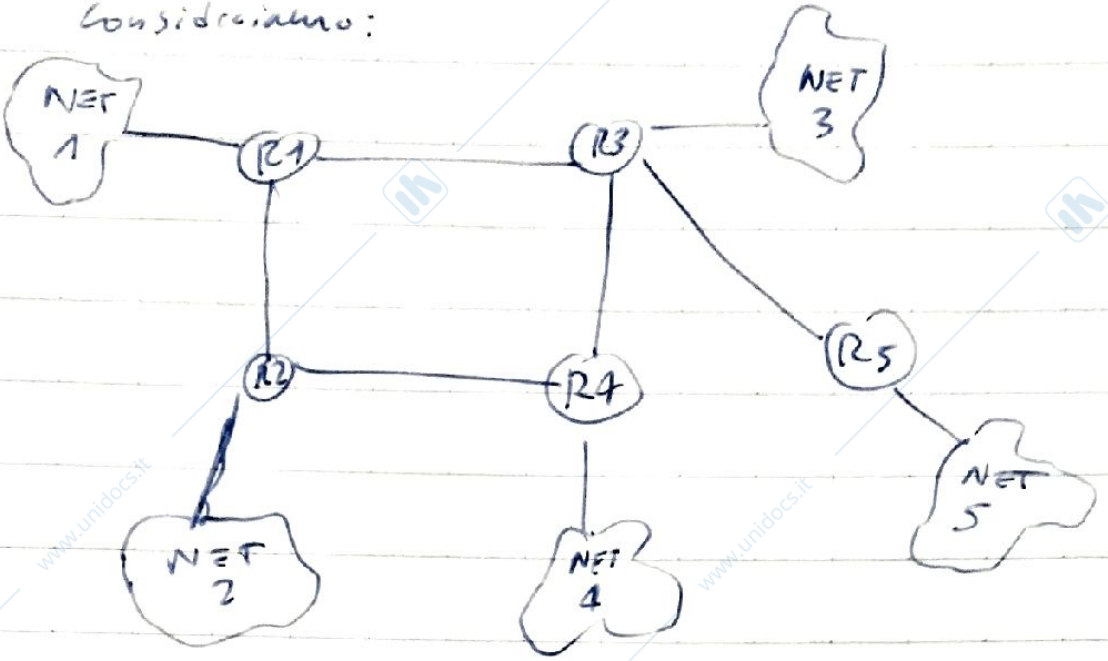
Mo Tu We Th Fr Sa Su

No. RETI DI TELE

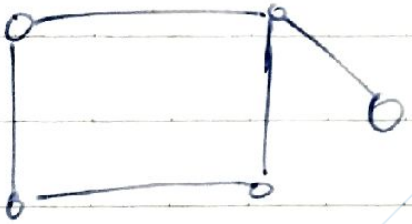
Date 10 · 12 · 19

→ Link state

Consideriamo:



Dal punto di vista dell'algoritmo consideriamo il grafo dove i nodi sono i router



LSA = Link State Advertisement

↳ messaggio che viene mandato in flooding (a tutti tranne che nel ramo di ingresso) da un router per dire come è collegato agli altri

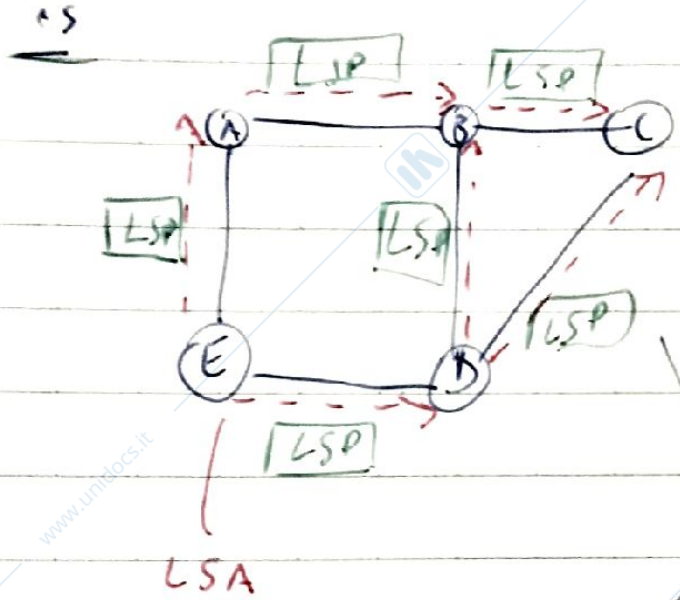
No. RETI DI TELE

Date 10 · 12 · 19

Arrivano i LSP (link state packet) infine a tutti

informazioni di link

aggiornando queste informazioni convergono all'ultimo nodo



passo tra il topology discovery

C scopre come è fatta la rete

ogni volta che so la topologia e i costi dovuti ai link → posso applicare l'algoritmo di Dijkstra

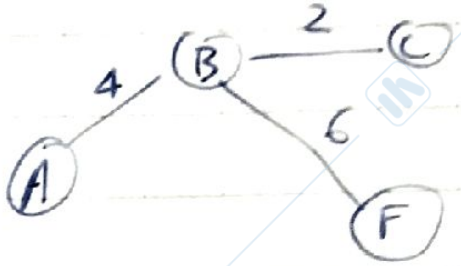
cammino minimo

es.

Scopriamo link state

Siamo al centro

RA



Router (nodi) : Costo

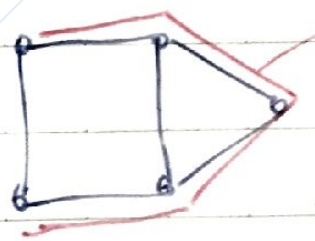
Link state funziona solo con costi con segno positivo

Come funziona l'algoritmo di Dijkstra?

- N = nodi della rete
- M = insieme dei nodi del MST corrente

Minimum Spanning Tree

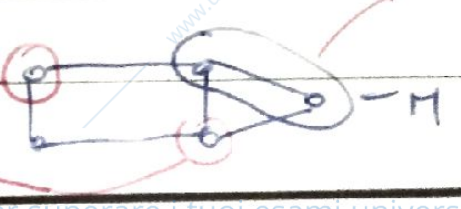
Percorso minimo verso + vicino destinazione



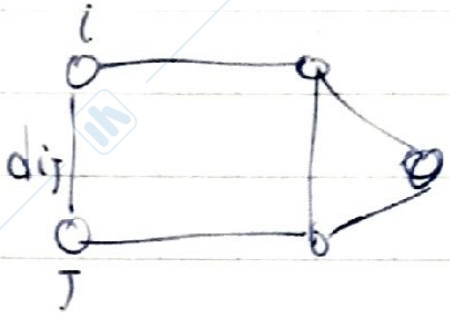
- V(M) = nodi vicini all'insieme M

1 es.

a ogni step aggiungo un nodo



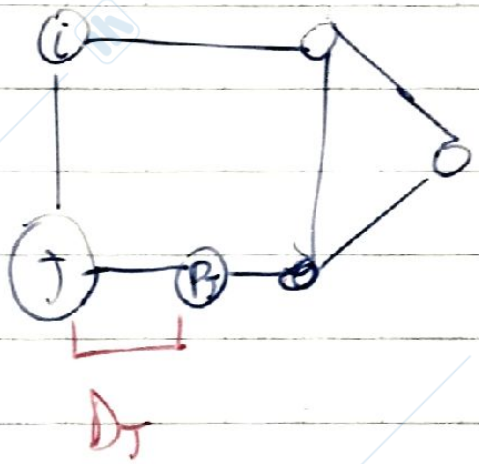
- d_{iT} = costo della via diretta tra i e T



se non è diretto $d_{iT} = \infty$

- D_T = costo della via a minimo costo da s a T

- P_j = nodo predecessore del nodo j



① Inizializzazione:

$M = \{s\}$ ← nodo radice

$D_j = d_{sj} \quad \forall j \in N - \{s\}, \; D_s = 0$

$P_j = s \quad \forall j \in N - \{s\}$

7

No. RETI DI TELE.

Date 20 · 12 · 19

(2) Scelgo $k \in V(M) \mid D_k = \min_{i \in V(M)} D_i$

$M = M \cup \{k\}$

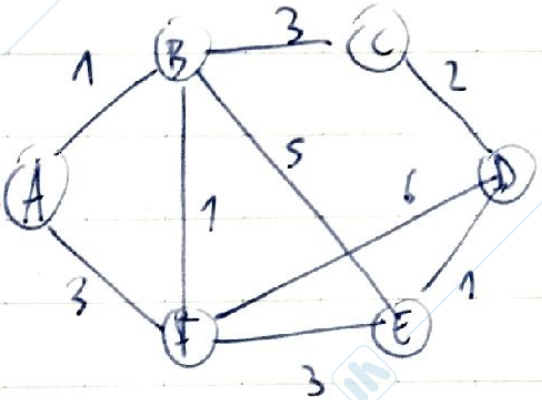
Se $D_j > D_k + d_{kj}$ allora $P_j = k \quad \forall j \in V(M)$

$D_j = \min \{D_j, D_k + d_{kj}\} \quad \forall j \in V(M)$

(3) Se $M = N$ STOP

Sennò vai a (2)

es.



Applico Dijkstra ad A

(1) $M = \{A\}$

Nodi	M{A}	
	Costi	Route
A	—	
B	1	B
C	∞	
D	∞	
E	∞	
F	3	F

(2) → $M\{A,B\}$

Mo X We Th Fr Sa Su 8

No RETI DI TELE.

Date 10 12 19

M{A,B}

	Costi	Route
A	—	--
B	1	B
C	4	B
D	∞	
E		
F		

② ..... siti

Tutto questo si applica in internet con LSA e il calcolo con gli algoritmi Dijkstra

con protocolli OSPF

Questo algoritmo non si può naturalmente applicare su tutta la rete

Routing Domain → gerarchizzazione sugli algoritmi

si fa Dijkstra su reti piccole e poi attraverso altri algoritmi li collego e collego e trovo effettivamente il cammino ottimale

9

Mo Tu We Th Fr Sa Su

No. LETI DI TELE

Date 10 · 12 · 19

IGP = Interior Gateway Protocol

↓

All'interno

del Routing Domain

EGP = Exterior Gateway Protocol

↓

All'esterno

del Routing Domain

OSPF = Open Shortest Path First

↳ non ci fa sapere

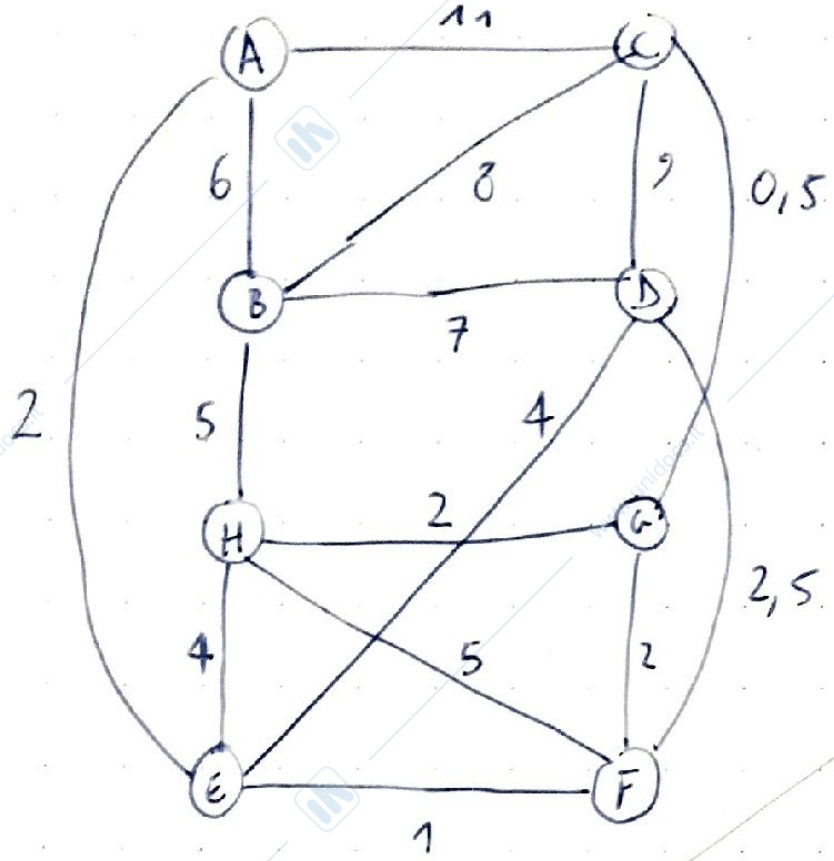
come è fatto il pacchetto OSPF

Mo Tu We Th Fr Sa Su

No. RETI DI TELE.

Date 10 12 19

ex. (6.11)



radice = B — M = {B}

costo ITERAZIONI : Predecessore

NODO	1	2	3	4	5	6	7	8
A	6, B	6, B →	—	—	—	—	—	—
C	8, B	8, B	8, B	8, B	7,5, G →	—	—	—
D	7, B	7, B	7, B →	—	—	—	—	—
E	∞, B	9, H	8, A	8, A	8, A	8, A →	—	—
F	∞, B	10, H	10, H	9,5, D	9, G	9, G	9, G →	—
G	∞, B	7, H	7, H	7, H →	—	—	—	—
H	5, B →	—	—	—	—	—	—	—

Costo minore

Predecessore inizializzato al nodo sorgente

$D_J = D_C = 8$

$D_K = D_A = 6$

$d_{KJ} = d_{AC} = 11$

$8 < 17$

→ tengo 8

ITERAZIONI:

① M = {B}

② M = {B, H}

Valutare D_J secondo: $D_J \geq D_K + d_{KJ}$ (K = H)

③ M = {B, H, A}

④ M = {B, H, A, D}

⑤ M = {B, H, A, D, G}

⑥ M = {B, H, A, D, G, C}

⑦ M = {B, H, A, D, G, C, E}

⑧ M = {B, H, A, D, G, C, E, F}

Livello 4

→ Strato di trasporto

Abbiamo bisogno di una porta che indirizzi ogni processo che si trova nel terminale

Il livello 4 si occupa di risolvere i problemi lasciati dal problema 3:

- perdita di pacchetti
- arrivo in ordine
- affidabilità della trasmissione

→ tutto ciò è il protocollo TCP

Ho bisogno di un'altra connessione diversa rispetto a quella che si fa a livello 2 poiché l'errore può avvenire dentro al router e non solo sul link.

→ Protocollo UDP

Voglio instaurare una comunicazione tra processi su macchine diverse.

No. APPUNTI DI IELE

Mo Tu We Th Fr Sa Su

Date 11 12 19

Non è sufficiente però l'indirizzo IP per indirizzare un processo.

↓

l'IP indirizza la macchina
la porta indirizza il processo

Funzione di buffering ⟵ TCP

Quando un processo viene associato ad una porta viene anche associato a un buffer

↙ 1 in ingresso
↗ 1 in uscita

tipo
doppia
finestra
a livello 2

→ le porte sono divise in 3 insiemi: (indirizzi di 16 bit)

- well-known
- numeri registrati
- numeri dinamici

Mo Tu We Th Fr Sa Su

No. RETI DI TELE.

Date 11 12 19

In UDP ho un tipo di servizio connectionless (non chiedo il permesso per inviare) e non affidabile (non aspetto il riscontro)

↳ Per servizi con bassa latenza e bisogno di trasferire velocemente i dati

→ Protocollo TCP

↳ affidabile (aspetto il riscontro)
connection oriented (chiedo il permesso)

↳ lo uso quando voglio la garanzia che i pacchetti arrivano e non ci siano errori

UDP vs TCP

- UDP non fa frammentazione

↳ tanto lo si può fare a livello 3

↳ header + piccolo di quello di TCP

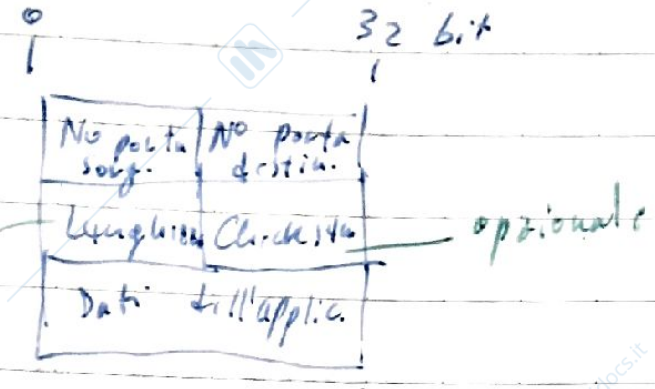
4

Mo Tu We Th Fr Sa Su

No. RET. DI TELE.

Date 12. 11. 19

Header UDP



dell'header + dei dati (in byte)

$2^{16} = 65536$

8 byte → L4

20 byte → L3

18 byte → L2

L1

d.vo togliere la lunghezza dagli header del livello 4 e dei precedenti

↳ Pseudoheader → serve per calcolare il checksum

(campi copiati dall'intestazione IP

→ verifica la correttezza dell'intestazione IP a destinazione

5

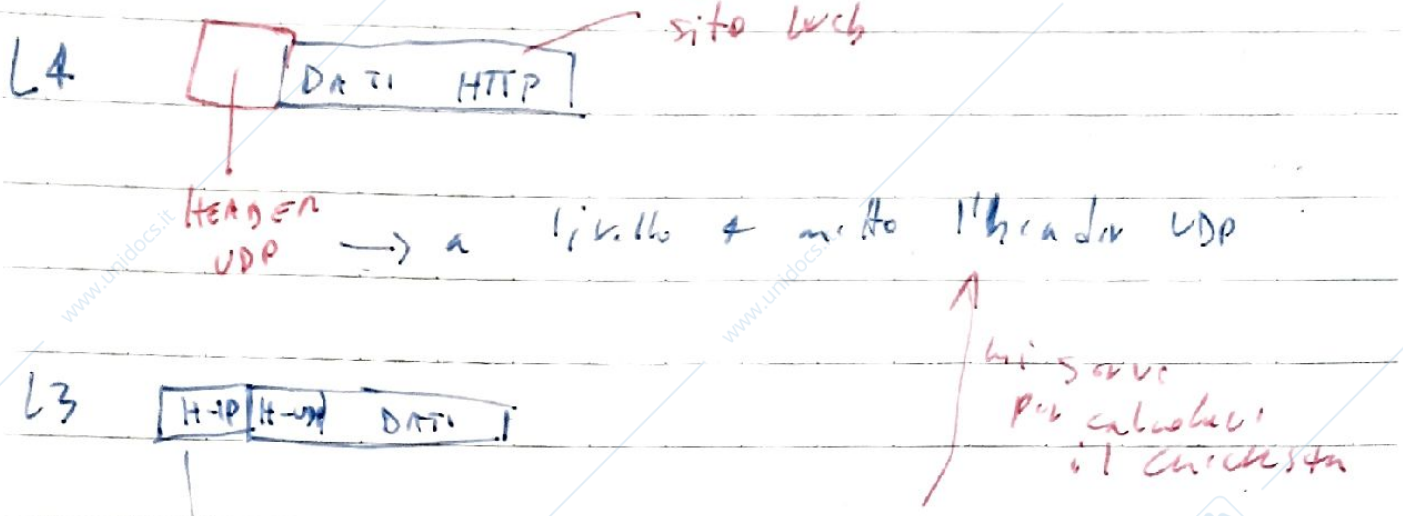
Mo Tu We Th Fr Sa Su

No. RETI DI TELE.

Date 12 11 19

Checksum = codice a rivelazione d'errore

Supponiamo di ricevere dal livello 5

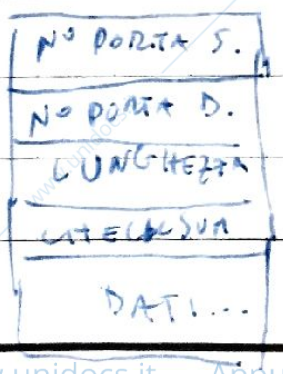


→ Calcolo del checksum

"Checksum" = "controlla somma"

mette in colonna tutte le parti dell'header

UDP per 16 bit



viene sommato allo pseudoheader

e poi nel campo checksum viene messo il compl. a 1 (NOT) della somma

6

No. RETI DI TELE.

Mo Tu We Th Fr Sa Su

Date 12 · 11 · 19

- Inoltre all'inizio del checksum abbiamo tutti 0
- Poi facciamo somma segmenti UDP a 16 bit + padding
- Poi facciamo il complemento a 1 (NOT) di questa somma
- Inoltre poi arrivando alla destinazione ci facciamo la somma e poi facciamo la somma totale (quindi sommando il checksum)

↓

se non ci sono stati errori otteniamo

1111111111111111

(

poiché il checksum è il compl. a 1 della somma parziale

→ Livello: basta che ci sia un errore, non dice quale è

Nel caso in cui ho il riporto della somma (+ bit in +) sommo l'1 in + al risultato (l'1 uscita dall'ingresso)

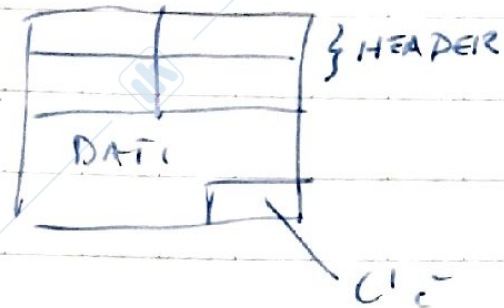
7

Mo Tu We Th Fr Sa Su

No. Reti di Tele

Date 12 · 11 · 19

Oss. Allineamento dei dati



C'è un padding

in modo che il calcolo del checksum avvenga su multipli di 16

ossia quanti divisi in dati da 16 bit per un HEXA in colonna

VS TCP

↓

fa molto di più

↓

header + complesso → 20 byte di header + (fino a 40 byte opzionali)

Meccanismo di buffer

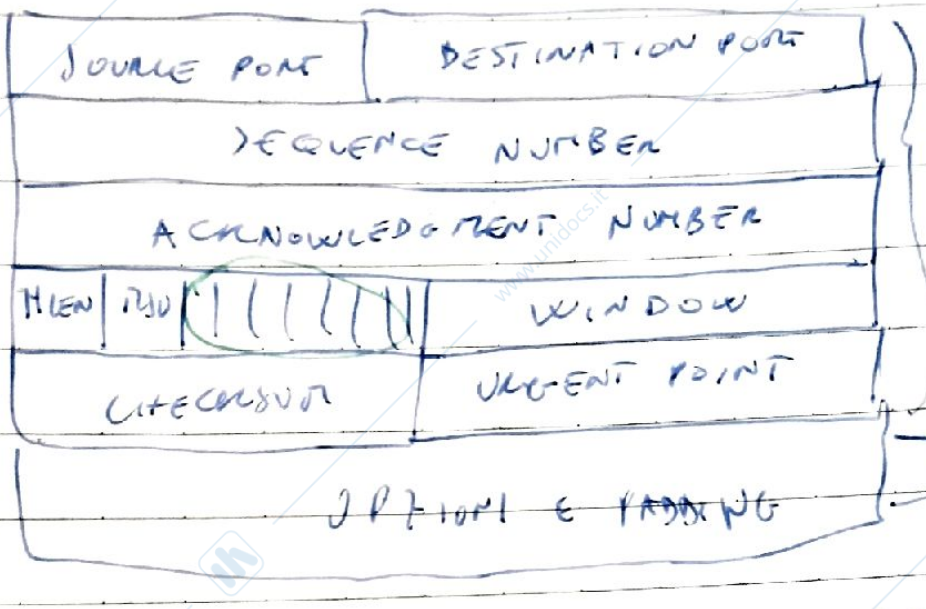
Mo Tu We Th Fr Sa Su

No. RETI DI TELE

Date 12. 11. 1

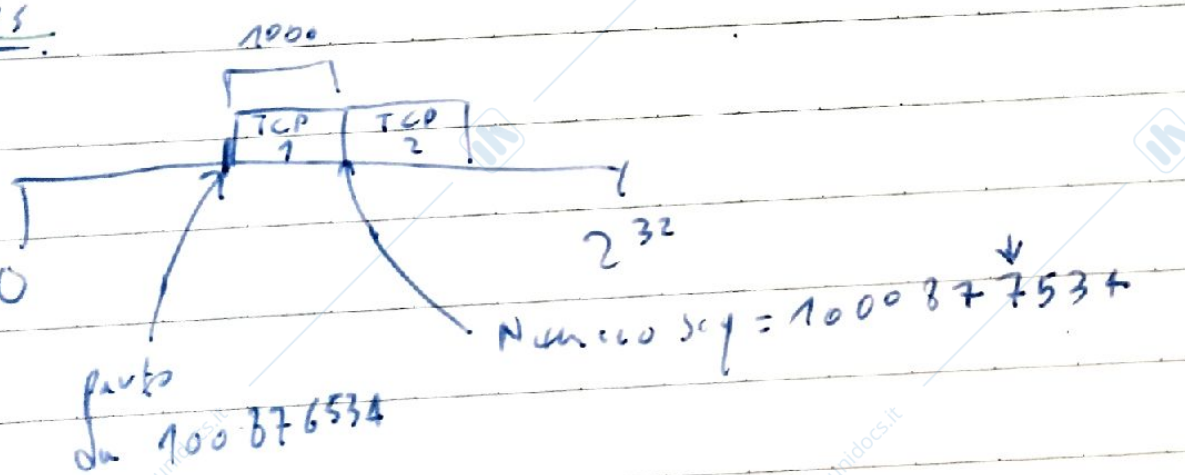
Livello di uscita funziona come a livello 2

Header



Sequence number: numero seq. del 1° byte del pacchetto a partire da un numero scelto in fase di instaurazione

es.



No. RETI DI TELE.

Mo Tu We Th Fr Sa Su

Date 12 · 11 · 19

Acknowledgment number: quello che nella trama TCP nel campo Control chiamavamo ACK(R)

numero che mi aspetto di ricevere ←

HLEN: lunghezza header in multipli da 32 bit

Window: campo che la destinazione utilizza per dire quanto spazio di mio sto buffer mi dispone

URG ACK PSH RST SYN FIN → sono dei flag

↳ i importanti: -RST
-SYN
-FIN

Opzioni: es. MSS
↳ Maximum Segment Size

↓

di default è 536 byte

Perché = L2-MTU = 576 byte a cui tolgo 20 byte di L3-IP e 20 byte di L4-TCP = 536 byte