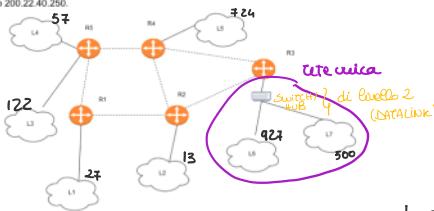


Indirizzamento (Super & Sub Netting) C.I.D.R.

Dato il Sistema Autonomo mostrato in Figura con indirizzo di base pari a 200.22.32.0, il candidato realizza un piano di indirizzamento classico con tecnica VLSM e CIDR. Si dovrà ottimizzare il numero dei blocchi utilizzati di classe C, considerando, inoltre, che nella sottorete L1 deve essere riservato l'indirizzo 200.22.40.250.



Richieste:

L1 : 27 host Vincolo (200.22.40.250)

L2 : 13 host

L3 : 122 host

L4 : 57 host

L5 : 724 host

L6 : 927 host

L7 : 507 host

200.22.40.250/27
bb 200.22.39.255/27
32 vis. \rightarrow 32/27
Subnetmask 255.255.255.224

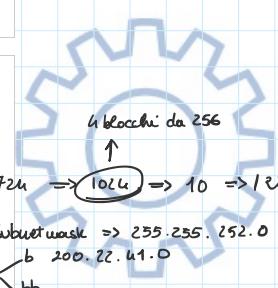
Partiamo dal vincolo su L1. L1 richiede 27 host + base + broadcast + R1 = 30 indirizzi \Rightarrow 32. Il vincolo ci impone che l'indirizzo 200.22.40.250 deve essere assegnato ad un sistema all'interno della sottorete. Per soddisfare entrambi i vincoli possiamo procedere nel seguente modo: Si assegna il broadcast 200.22.40.255/27 e si sceglie l'ultimo blocco da 32 indirizzi della sottorete 200.22.40.0/24. Otteniamo così la base di L1:

L1 Piano di indirizzamento
Base : 200.22.40.224/27
Broadcast : 200.22.40.255/27
R1 : 200.22.40.225/27

*le basi vuole pescare come quelle
che uscirebbe numero di host richiesto*

A questo punto prendiamo la sottorete che richiede il maggior numero di indirizzi. Dall'analisi delle richieste si può calcolare che la rete L6 = (L6|L7) richiede il maggior numero di indirizzi per un totale di 927+500 = base + broadcast + R3 = 1430 \Rightarrow 2048 indirizzi. Per poter assegnare un blocco unico dobbiamo ricorrere al supernetting andando ad assegnare ad L6 8 blocchi di classe C con una maschera /21 = 255.255.248.0. Il primo blocco di indirizzi liberi è quello che parte dalla base 200.22.32.0/21.

2048 \Rightarrow 11 \Rightarrow 121
Subnetmask \Rightarrow 255.255.248.0
8 blocchi



Indirizzamento (Super & Sub Netting) C.I.D.R.

L6 Piano di indirizzamento
Base : 200.22.32.0 / 21
Broadcast : 200.22.39.255 / 21
R3 : 200.22.32.1 / 21

La seconda sottorete più grande è L5 che richiede 724 indirizzi per host raggiungendo una richiesta complessiva di 724 + base + broadcast + R4 = 727 \Rightarrow 1024. Il risultato ci impone di considerare una maschera di /22 = 255.255.252.0. Abbiamo bisogno di 200.22.32.0/21, 200.22.40.0/24 (Abbiamo assegnato in precedenza ad L1), il blocco /24 non può essere assegnato. Non resta quindi che andare sulla 200.22.44.0/22.

L5 Piano di indirizzamento
Base : 200.22.44.0 / 22
Broadcast : 200.22.47.255 / 22
R4 : 200.22.44.1 / 22

La terza sottorete è L3 che richiede 122 + base + broadcast + R5 = 125 indirizzi \Rightarrow 128. Il primo blocco libero /25 = 255.255.255.128 appartiene alla sottorete 200.22.40.0 / 25 che possiamo quindi assegnare tranquillamente ad L3.

L3 Piano di indirizzamento
Base : 200.22.40.0 / 25
Broadcast : 200.22.40.127 / 25
R5 : 200.22.40.1 / 25

L4 è la quarta sottorete per grandezza il che ci porta ad assegnare 57 + base + broadcast + R6 = 60 \Rightarrow 64 indirizzi che corrisponde ad una maschera /26 = 255.255.255.192. Il primo blocco da 64 indirizzi disponibile è a partire dalla base 200.22.40.128 / 26

L4 Piano di indirizzamento
Base : 200.22.40.128 / 26
Broadcast : 200.22.40.191 / 26
R6 : 200.22.40.129 / 26

Infine la sottorete L2 può essere allocata nello spazio rimasto libero a partire dalla 200.22.40.192/27 al 200.22.40.223/27. Ovviamente di questi 32 indirizzi prenderemo solo quelli necessari a soddisfare il vincolo dello spazio richiesto da L2 ovvero 13*base+broadcast+R2 = 16 \Rightarrow 16 corrispondente ad una maschera /28 = 255.255.255.240

L2 Piano di indirizzamento
Base : 200.22.40.192 / 28
Broadcast : 200.22.40.207 / 28
R2 : 200.22.40.193 / 28

CLASSE C : $2^8 + 8 = 32$

L1

$$I = R_1 + b + bb + 2^7 = 30$$

$$\text{qui tutti } 2^5 = 32$$

$$\text{MASK: } 32 - 3 = 29$$

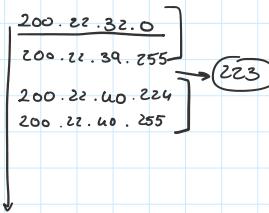
$$8 \cdot 8 \cdot 8 \cdot 3$$

$$\underline{\underline{255.255.255.224}}$$

Possiamo occupare gli ultimi di 200.22.40.250 :

$$b = 200.22.40.224 / 27$$

$$bb = 200.22.40.255 / 27$$



$$b = 200.22.40.224 / 27$$

$$bb = 200.22.40.255 / 27$$

$$32 \text{ vis.} \rightarrow 32 / 27$$

Subnetmask 255.255.255.224

$$200.22.39.255 / 27$$

$$bb = 200.22.39.255 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.224 / 27$$

$$bb = 200.22.40.224 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.128 / 26$$

$$bb = 200.22.40.128 / 26$$

32 vis. $\rightarrow 32 / 26$

Subnetmask 255.255.255.192

$$200.22.40.192 / 28$$

$$bb = 200.22.40.192 / 28$$

16 vis. $\rightarrow 16 / 28$

Subnetmask 255.255.255.240

$$200.22.40.193 / 28$$

$$bb = 200.22.40.193 / 28$$

16 vis. $\rightarrow 16 / 28$

Subnetmask 255.255.255.240

$$200.22.32.0 / 21$$

$$bb = 200.22.32.0 / 21$$

32 vis. $\rightarrow 32 / 21$

Subnetmask 255.255.252.0

$$200.22.39.255 / 21$$

$$bb = 200.22.39.255 / 21$$

32 vis. $\rightarrow 32 / 21$

Subnetmask 255.255.252.0

$$200.22.40.224 / 27$$

$$bb = 200.22.40.224 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.225 / 27$$

$$bb = 200.22.40.225 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.226 / 27$$

$$bb = 200.22.40.226 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.227 / 27$$

$$bb = 200.22.40.227 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.228 / 27$$

$$bb = 200.22.40.228 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.229 / 27$$

$$bb = 200.22.40.229 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.230 / 27$$

$$bb = 200.22.40.230 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.231 / 27$$

$$bb = 200.22.40.231 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.232 / 27$$

$$bb = 200.22.40.232 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.233 / 27$$

$$bb = 200.22.40.233 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.234 / 27$$

$$bb = 200.22.40.234 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.235 / 27$$

$$bb = 200.22.40.235 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.236 / 27$$

$$bb = 200.22.40.236 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.237 / 27$$

$$bb = 200.22.40.237 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.238 / 27$$

$$bb = 200.22.40.238 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.239 / 27$$

$$bb = 200.22.40.239 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.240 / 27$$

$$bb = 200.22.40.240 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.241 / 27$$

$$bb = 200.22.40.241 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.242 / 27$$

$$bb = 200.22.40.242 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.243 / 27$$

$$bb = 200.22.40.243 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.244 / 27$$

$$bb = 200.22.40.244 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.245 / 27$$

$$bb = 200.22.40.245 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.246 / 27$$

$$bb = 200.22.40.246 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.247 / 27$$

$$bb = 200.22.40.247 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.248 / 27$$

$$bb = 200.22.40.248 / 27$$

32 vis. $\rightarrow 32 / 27$

Subnetmask 255.255.255.224

$$200.22.40.249 / 27$$

$$bb = 200.22.40.249 / 27$$

$$R_1 = 200 \cdot 22 \cdot 40 \cdot 225 / 27$$

L6 + L7

$$I = 1627 + b + bb + R_3 = 1430$$

quindi $2^{10} = 2048$ (8 blocchi)

perciò $32 - 11 = 21$

MASK: $8 \cdot 8 \cdot 5 \cdot 0$

$$\underline{255 \cdot 255 \cdot 248 \cdot 0}$$

$$b = 200 \cdot 22 \cdot 32 \cdot 0 / 21$$

$$bb = 200 \cdot 22 \cdot 39 \cdot 255 / 21$$

$$R_3 = 200 \cdot 22 \cdot 32 \cdot 1 / 21$$

Rimane libero 200.22.40.0 fino a 200.22.40.228

L5

$$I = 726 + b + bb + Ru = 727$$

quindi $2^{10} = 1024$

perciò $32 - 10 = 22$

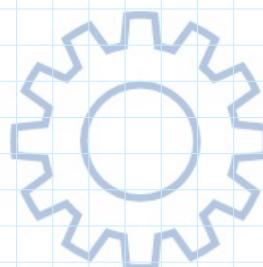
MASK: $8 \cdot 8 \cdot 6 \cdot 0$

$$\underline{255 \cdot 255 \cdot 252 \cdot 0}$$

$$b = 200 \cdot 22 \cdot 40 \cdot 224 / 21$$

$$bb =$$

$$Ru =$$



APPUNTI DI INGEGNERIA INFORMATICA

GAIA BERTOLINO

Esercitazione 7

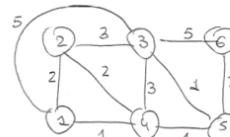
giovedì 10 febbraio 2022 16:59



E07 -
Algoritmo...

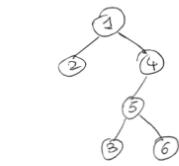
$$\begin{aligned} k=1 \\ T = \{1\} \\ L(2) = \infty \quad w(1,2) = 2 \\ L(3) = w(1,3) = 5 \\ L(4) = w(1,4) = 1 \\ L(5) = w(1,5) = \infty \\ L(6) = w(1,6) = \infty \end{aligned}$$

$$\begin{aligned} k=2 \\ L(x) = \min_{y \in T} L(y) \Rightarrow L(4) \Rightarrow x=4 \\ T = \{1,4\} \\ L(2) = \min [L(2), L(4) + w(4,2)] = 2 \\ L(3) = \min [L(3), L(4) + w(4,3)] = 4 \\ L(5) = \min [L(5), L(4) + w(4,5)] = 2 \\ L(6) = \min [L(6), L(4) + w(4,6)] = \infty \end{aligned}$$



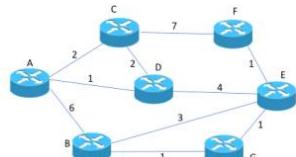
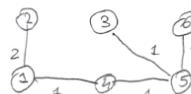
$$\begin{aligned} k=3 \\ L(x) = \min_{y \in T} L(y) = L(2) \Rightarrow x=2 \\ T = \{1,4,2\} \\ L(3) = \min [L(3), L(2) + w(2,3)] = 4 \\ L(5) = \min [L(5), L(2) + w(2,5)] = 2 \\ L(6) = \min [L(6), L(2) + w(2,6)] = \infty \end{aligned}$$

$$\begin{aligned} k=4 \\ L(x) = L(5) \quad x=5 \\ T = \{1,4,2,5\} \\ L(3) = \min [L(3), L(5) + w(5,3)] = 3 \\ L(6) = \min [L(6), L(5) + w(5,6)] = 4 \end{aligned}$$



$$\begin{aligned} k=5 \\ L(x) = L(3) \quad x=3 \\ T = \{1,4,2,5,3\} \\ L(6) = \min [L(6), L(3) + w(3,6)] = 4 \end{aligned}$$

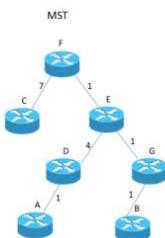
$$\begin{aligned} k=6 \\ L(x) = L(6) \quad x=6 \\ T = \{1,4,2,5,3,6\} \end{aligned}$$



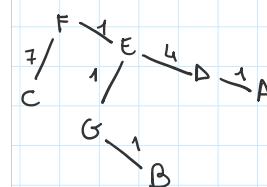
ALGORITMO DI DIJKSTRA

	A	B	C	D	E	F	G
F	∞	∞	7/F	∞	1/F	//	∞
FE	∞	4/E	7/F	5/E	//	//	2/E
FEG	∞	3/G	7/F	5/E	//	//	//
FEGB	9/B	//	7/F	5/E	//	//	//
FEGBD	6/D	//	7/F	//	//	//	//
FEGBDA	//	//	7/F	//	//	//	//
FEGBDAC	//	//	//	//	//	//	//

	A	B	C	D	E	F	G
F	∞	∞	7/F	∞	1/F	∞	∞
FE	∞	3/G	7/F	5/E	//	\times	2/E
FEG	∞	3/G	7/F	5/E	//	\times	∞
FEGB	9/B	\times	7/F	5/E	//	\times	\times
FEGBD	6/D	\times	7/F	//	\times	\times	\times
FEGBDA	\times	\times	7/F	//	\times	\times	\times
FEGBDAC	\times						



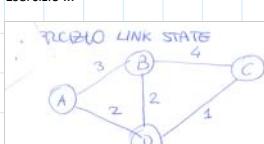
TO	NEXT	COST
C	C	7
E	E	1
D	E	5
G	E	2
A	E	6
B	E	3



TO	NEXT	COST
C	C	7
E	E	1
D	E	5
G	E	2
A	E	6
B	E	3

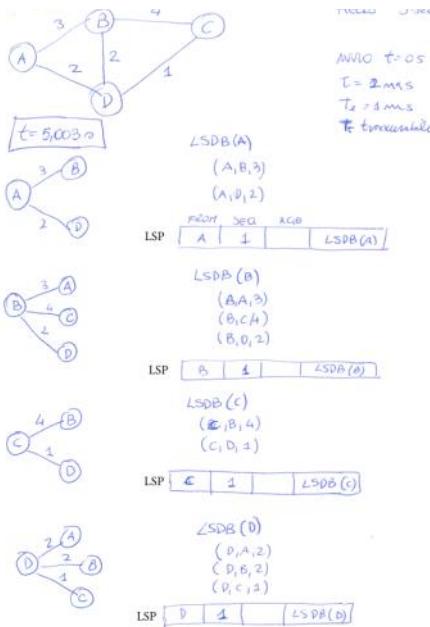
Protocollo di Link State → si basa sullo scambio di messaggi LSP che servono a far conoscere a tutta i nodi le informazioni agli altri nodi.

E07 -
Esercizio ...



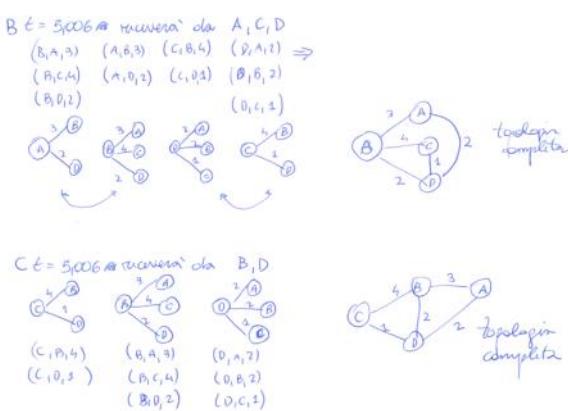
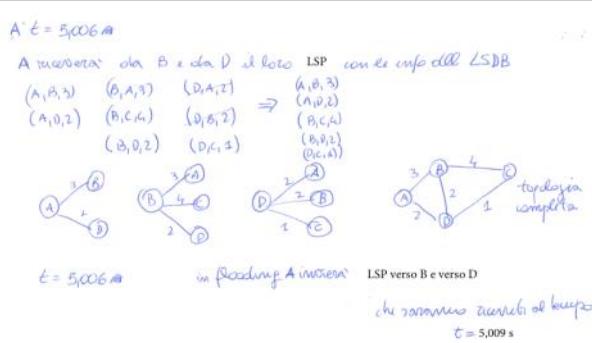
HELLO 5sec
ANVO t=0s
T=2ms
Tmax = 2... = 2.10^-3 s

→ a 5 secondi PARTE l'Hello
Hello = 5 s
anvo = 0 s
Tmax = 2... = 2.10^-3 s



Analizza la rete in figura seguendo un protocollo Link State per ottenere le tabella di Routing dei nodi. Si consideri un Hello packet con timer 5 sec. Si osserva che l'ente non ha tempo per inviare, tuttavia, un messaggio di risposta.

Si evidenzia il funzionamento del protocollo mostrando lo scambio di messaggi LSU tra i nodi della rete. Mostri l'esecuzione dell'algoritmo di invio/trasmissione da parte del protocollo.



HELLO = 5 s

avvio = 0 s

$t_{prop} = 2 \text{ ms} = 2 \cdot 10^{-3} \text{ s}$

$t_{lab} = 1 \text{ ms} = 1 \cdot 10^{-3} \text{ s}$

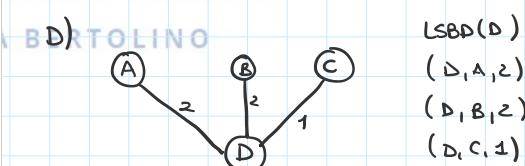
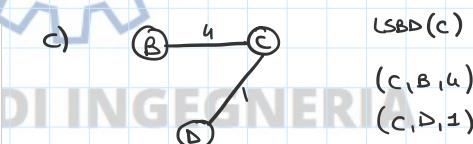
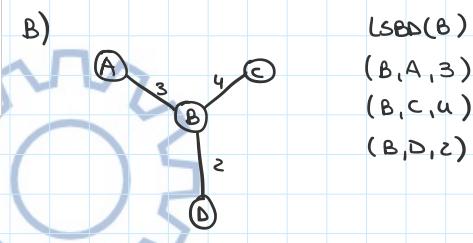
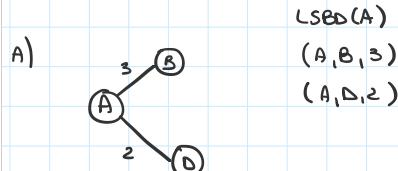
$t_{reas} = \text{trascurabile}$

Dunque $t = \text{Hello} + t_{prop} + t_{lab} = 5,003 \text{ s}$

Si costruiscono le informazioni circostanti

Un generica informazione LSP è del tipo

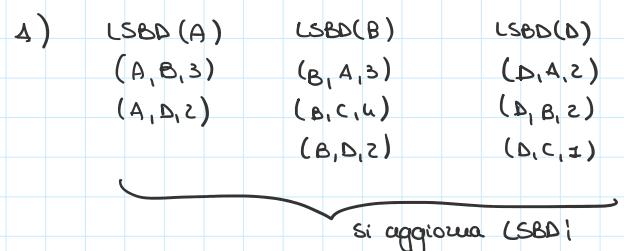
FROM	SEQ	AGE
A	1	(LSDB(A))



A $t = 5,003 \text{ s}$ vengono inviate le info

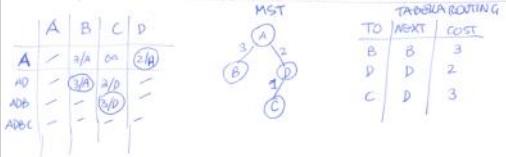
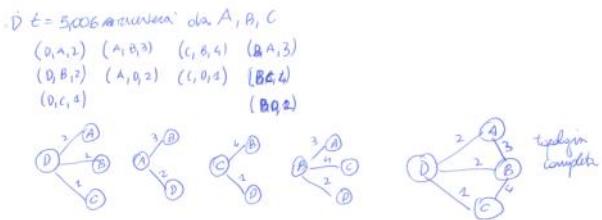
che arriveranno, a causa del ritardo, a $t = 5,006 \text{ s}$

per cui si aggiornano gli LSBD con le info degli LSP!

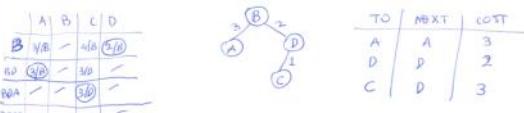


LSBD(A) al tempo $t = 5,006 \text{ s}$

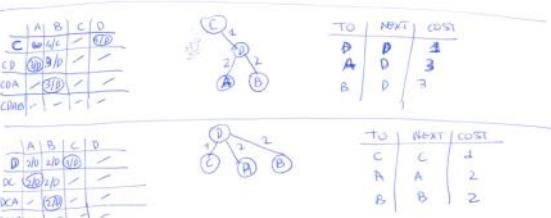
(A, B, 3)
(A, D, 2)
(B, D, 2)
(B, C, 4)



TO	NEXT	COST
B	B	3
D	D	2
C	D	3



TO	NEXT	COST
A	A	3
D	D	2
C	D	3

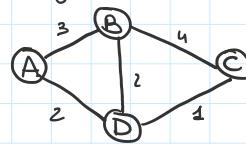


TO	NEXT	COST
P	P	3
A	D	3
B	D	2

...

- (A, D, 2)
- (B, D, 2)
- (B, C, 4)
- (D, C, 1)

La topologia conosciuta da A sarà



A ha la topologia completa

A genererà un nuovo LSP che viene inviato ai suoi nodi vicini che verrà inviata ai vicini al tempo $t=5,009$

B)	LSBD(B)	LSBD(A)	LSBD(D)	LSBD(C)
	(B, A, 3)	(A, B, 3)	(D, A, 2)	(C, B, 4)
	(B, C, 4)	(A, D, 2)	(D, B, 2)	(C, D, 1)
	(B, D, 2)			

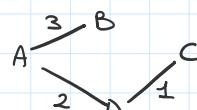
- LSBD(B)
- (B, A, 3)
 - (B, C, 4)
 - (B, D, 2)
 - (A, D, 2)
 - (D, C, 1)

APPUNTI DI INGEGNERIA INFORMATICA

GAIA BERTOLINO
Applica Dijkstra!

TABELLA DI ROUTING DI A

	A	B	C	D
A	X	3/A	00	<u>2/A</u>
AD	X	<u>3/A</u>	<u>3/D</u>	X
ADB	X	X	X	X



TO	NEXT	COST
B	B	3
D	D	2
C	D	3

DIFF. LVC e DVI :

Sono protocollo silenzioso a livello diretto che differiscono per:

Distance-Vector

- ogni router invia le informazioni di routing ai router adiacenti
- l'informazione trasmessa è una stima del costo verso tutte le reti
- l'informazione è trasmessa su base periodica regolare
- un router determina l'informazione sul next-hop usando l'algoritmo di Bellman-Ford distribuito sulle stime dei costi ricevute

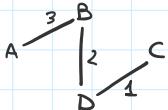
Link-State

- ogni router invia le informazioni di routing a tutti gli altri router
- l'informazione trasmessa è il valore esatto del costo del link verso le reti adiacenti
- l'informazione è trasmessa quando avviene un cambiamento
- un router costruisce prima una descrizione della topologia della rete e poi usa un algoritmo di routing qualsiasi per calcolare le informazioni sul next-hop (tipicamente Dijkstra)

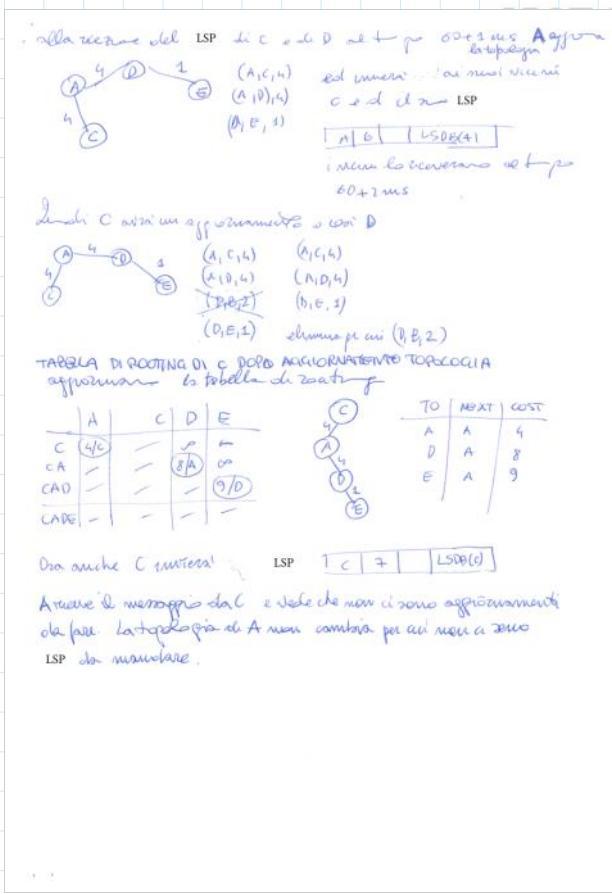
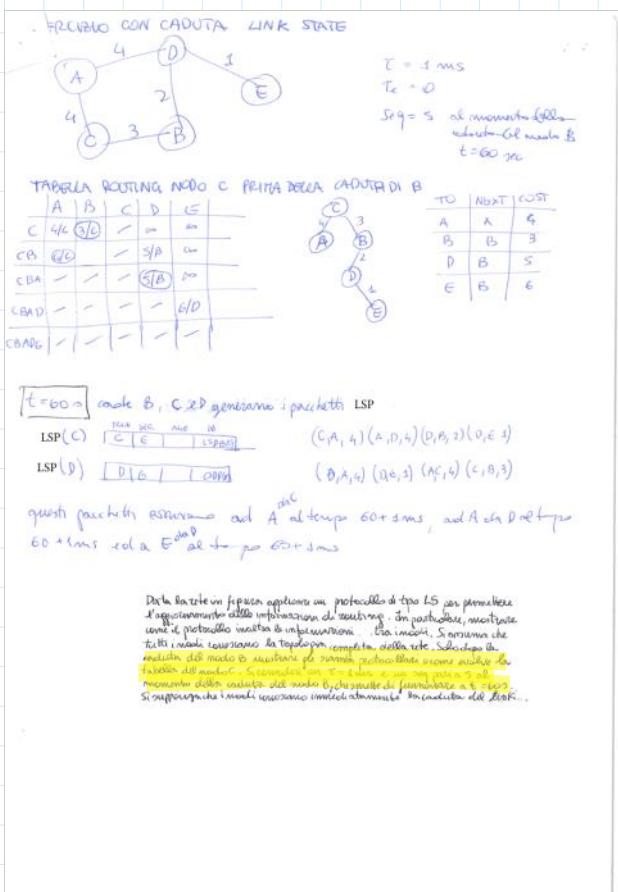
TABELLA DI B

	A	B	C	D
B	3/B	X	4/B	<u>2/B</u>
BD	<u>3/B</u>	X	<u>3/D</u>	X
BDA	X	X	X	X

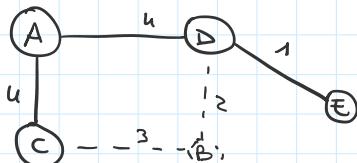
TO	NEXT	COST
A	A	3
D	D	2
C	D	3



E08 -
Esercizio ...



In questo caso si ha che t = 60 ms tutti i nodi conoscono la topologia e che B cade. A t = 60 ms la rete diventa



Vengono generati degli LSP dove la seq sarà pari a 6:

A: LSBD(A)
(A, D, 4)
(A, C, 4)
(D, E, 1)
(D, B, 2)
(C, B, 3)



D: LSBD(D)
(D, A, 4)
(D, E, 1)
(D, B, 2)
(A, C, 4)
(C, B, 3)

C: LSBD(C)

(C, A, 1)
(A, D, 4)
(D, E, 1)
(D, B, 2)
(C, B, 3)

E: LSBD(E)

(E, D, 1)
(D, B, 2)
(C, B, 3)
(A, C, 4)
(A, D, 4)

A t = $60 + 1 \cdot 10^{-3} = 60,001 \Delta$ avrà l'apprendimento!

A: LSBD(A) $\begin{bmatrix} \text{seq 6} & \text{seq 12} \\ A \rightarrow D & \text{LSP}(A) \end{bmatrix}$ LSBD(D) $\begin{bmatrix} \text{seq 7} & \text{seq 12} \\ D \rightarrow A & \text{LSP}(D) \end{bmatrix}$ LSBD(C) $\begin{bmatrix} \text{seq 7} & \text{seq 12} \\ C \rightarrow D & \text{LSP}(C) \end{bmatrix}$

(A, D, 4)
(A, C, 4)
(D, E, 1)
(D, B, 2)
(C, B, 3)

~~(D, B, 2)~~ \rightarrow ~~(D, B, 2)~~ \rightarrow ~~(C, B, 3)~~ \rightarrow ~~(C, B, 3)~~

D: LSBD(D) $\begin{bmatrix} \text{seq 7} & \text{seq 12} \\ D \rightarrow A & \text{LSP}(D) \end{bmatrix}$ LSBD(A) $\begin{bmatrix} \text{seq 6} & \text{seq 12} \\ A \rightarrow D & \text{LSP}(A) \end{bmatrix}$ LSBD(E) $\begin{bmatrix} \text{seq 8} & \text{seq 12} \\ E \rightarrow D & \text{LSP}(E) \end{bmatrix}$

(D, A, 4)
(D, E, 1)
(A, C, 4)
(C, B, 3)
(A, D, 4)

~~(D, B, 2)~~ \rightarrow ~~(D, B, 2)~~ \rightarrow ~~(C, B, 3)~~ \rightarrow ~~(C, B, 3)~~

C: LSBD(C) LSBD(A)

↓ Se Hello ≠ 0 allora C e D si accorgono
che B caduto a t + 3 · Hello

C:	LSBD(c)	LSBD(A)
	(C, A, u)	(A, D, u)
	(A, D, u)	(A, C, u)
	(D, E, z)	(D, E, z)
	(D, A, z)	(D, E, z)
	(C, B, z)	(C, B, z)

E:	LSBD(E)	LSBD(D)
	(E, D, z)	(D, A, u)
	(D, B, z)	(D, E, z)
	(C, B, z)	(D, B, z)
	(A, C, u)	(A, C, u)
	(A, D, u)	(C, B, z)

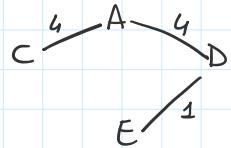
C si aggiornereà definitivamente a 60,002 → poiché
serve un altro aggiornamento di E



Tabella 1:

	TO	NEXT	COST
A	A	A	u
D	A	A	8
E	A	A	9

A C D E
u C X ∞
 CA X ∞ 8 A ∞
 CAD X X 9 D
 CADE X X X X



Esercizio Trasporto - CWND

Fondamenti di Reti di
Telecomunicazioni

Fondamenti di Reti di Telecomunicazioni

Esercizio 1

Dati i seguenti parametri determinare quanto tempo impiega il nodo A a trasmettere 53600Byte.

- 1 MSS = 536Byte;
- Δ = 52Byte; $L = 536$; $C = 1$; $R = 1$

Payload
 MSS → Maximum segment size
 CWND → finestra

Dati i seguenti parametri determinare quanto tempo impiega il nodo A a trasmettere 53600Byte.

- 1 MSS = 536Byte;
- CWND_start = 1 MSS; $= 536 \text{ byte} = 536 \cdot 8 \text{ bit}$
- Finestra del Ricevitore = 16MSS; $= 16 \cdot 536 \text{ byte} = 16 \cdot 536 \cdot 8 \text{ bit}$
- capacità del canale = 200Kbps;

Inoltre si vuole sapere quanti byte sono trasferiti in CA e quanti in SS. Mostrare l'andamento della finestra di congestione.

Calcoliamo il **numero di MSS totali** da dover trasferire:

$$N_{MSS} = \lceil \frac{53600}{536} \rceil = 100$$

Calcoliamo ora quanti MSS saranno trasmessi in **CA** e quanti in **SS**. Per poter effettuare questo calcolo **dobbiamo sapere quale sarà l'upper bound della CWND**. Per effettuare questo calcolo **dobbiamo conoscere il limite imposto dalla capacità del canale**.

$$MAXMSS = \lfloor \frac{(200000)}{602 + 8} \rfloor = 41 \frac{\text{MSS}}{\text{s}}$$

L'upper bound sarà quindi dato dal MaxWin = $\min(S_Win, R_Win) = 16 \text{ MSS}$

$$T_{SS} = RTT * \log_2(16) = 4RTT$$

Di conseguenza 4 saranno gli RTT necessari a raggiungere la MaxWin.

Calcoliamo quanti MSS sono stati trasferiti in 4RTT:

$$B_{SS} = (1 + 2 + 4 + 8 + 16)MSS$$

$$\underline{2^0 \ 2^1 \ 2^2 \ 2^3 \ 2^4}$$

Fondamenti di Reti di Telecomunicazioni

$$\sum_{i=0}^k 2^i MSS = \\ B_{SS} = MSS * (2^{k+1} - 1)$$

Sono rimasti da inviare 100-31 MSS = 69 MSS = 36984 Byte

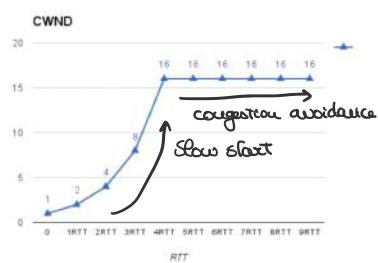
Quanti RTT durerà ancora la trasmissione?

$$\lceil \frac{69}{16} \rceil = 5RTT$$

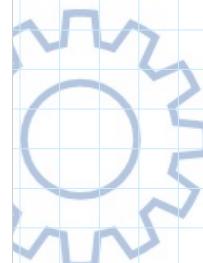
Dove 16 sono gli MSS inviati ad ogni RTT (MaxWin)

Calcoliamo la durata complessiva come somma delle componenti trovate

$$4RTT + 5RTT = 9RTT$$



Fondamenti di Reti di Telecomunicazioni



DI INGEGNERIA
FORMATICA
BERTOLINO

Esercizio 2

Due stazioni connesse attraverso un router attivano una sessione FTP per scambiare un file di dimensioni 373760 Byte. Si consideri la dimensione del segmento trasporto, al netto degli header, pari a 1460 Byte. Si consideri che il ricevitore possa contenere all'interno del suo buffer al massimo 64MSS. La sorgente potrà invece inviare al massimo 32 MSS.

Si vuole conoscere :

$$MSS = 1460 \text{ byte}$$

$$WR = 64 \text{ MSS}$$

$$WS = 32 \text{ MSS}$$

- la durata complessiva della trasmissione in RTT;
- i byte trasmessi in CA e quelli trasmessi in SS;
- Mostrare l'andamento della CWND nel caso in cui si applichi il TCP Tahoe;

Risoluzione

Procediamo a calcolare il numero di segmenti da inviare

$$N = \left\lceil \frac{373760}{1460} \right\rceil = 256$$

Dobbiamo ora identificare l'upper bound della CWND che è dato da:

$$\min(R_Win, S_Win) = \min(64, 32) = 32 \text{ MSS}$$

Adesso che abbiamo l'upperbound possiamo determinare quanti RTT occorrono per poter arrivare a 32 MSS

$$N_{RTT} = \log_2(32) = 5 = k$$

Calcoliamo quanti Byte possiamo inviare in questa fase dello SS

$$B(SS) = (2^{k+1} - 1) * MSS = 63 MSS$$

$$B(SS) = 91980 \text{ Byte}$$

Abbiamo quindi inviato 63 MSS su un totale di 256 MSS, questo indica che dobbiamo ancora trasferire (256-63) MSS = 193 MSS

Fondamenti di Reti di Telecomunicazioni

Quanti RTT ci occorrono per trasferire questi MSS?

$$RTT(SS)_2 = \lceil \frac{193}{32} \rceil = 7$$

In totale per trasferire tutti i segmenti occorrono $5 + 7 = 12$ RTT;

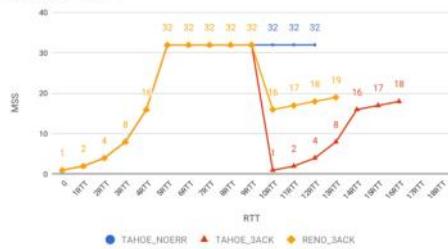
N.B.

Nell'ultimo RTT, in particolare nel 7, saranno inviati solo gli MSS rimanenti che sono dati dalla seguente espressione

$$193 - (32 * 6) = 1MSS$$

L'andamento della CWND è mostrato di seguito

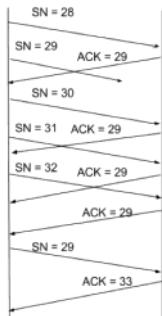
Andamento CWND



Consideriamo ora una piccola variante: supponiamo che al 9° RTT si verifichi un evento di perdita di segmento che genererà la ricezione di 3-ack duplicati, come si comporterà il protocollo Tahoe e poi quello Reno. Il segmento della finestra del 9° RTT che si perde è il 29.

Fondamenti di Reti di Telecomunicazioni





Nella versione Tahoe, otteniamo le seguenti fasi.

1. CWND = 1;
2. SSThreshold = CWND/2 = 32/2 = 16 MSS;
3. Inizio con fase di SS perché (CWND < SSThreshold);
4. quando CWND >= SSThreshold => Inizio CA;
5. Avremo quindi

$$k' = \log_2(SSThreshold) = \log_2(16) = 4$$

$$B(SS)' = [2^{k'+1} - 1]MSS = 31MSS$$

Dove B(SS)' sono ulteriori Byte inviati in SS;

Non resta che calcolare quanti Byte saranno inviati in CA => B(CA)

In totale in SS sono stati inviati $191 + 31$ MSS = 222 MSS. Dobbiamo ancora inviare $256 - 222 = 34$ MSS che saranno tutti inviati in CA

$B(CA) = 34$ MSS occorrono ulteriori 2 RTT ($17 + 18$)

Fondamenti di Reti di Telecomunicazioni

Durata = $9 + 5 + 2 = 16$ RTT

Dove 9 sono i primi RTT fin tanto che tutto va bene. Al 10° ripartiamo con la SS a partire da 1MSS + altri 4 RTT per arrivare al valore CWND = 16, poi abbiamo altri 2 RTT per inviare i restanti 34 MSS.

Nella versione RENO

1. SSThreshold = CWND/2 = 32/2 = 16;
2. CWND = CWND/2
3. CWND >= SSThreshold => Inizio CA;
4. Dobbiamo calcolare quanti Byte sono trasmessi in CA => B(CA);

La differenza rispetto al caso precedente è che questa volta non si parte dalla condizione di SS ma si riparte da una CA

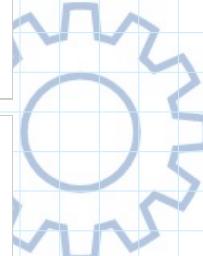
$B(CA) = 256 - 191 = 65$ MSS;

per inviarli occorrono ulteriori $16 + 17 + 18 + 19 = 60$ RTT

Durata = $9 + 4 = 13$ RTT

Ultima finestra sarà composta da 14 MSS.

Fondamenti di Reti di Telecomunicazioni



DI INGEGNERIA
FORMATICA
BERTOLINO

Trasporto

lunedì 14 febbraio 2022

17:47

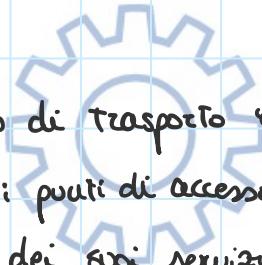
Internet è composta da host connessi a reti a commutazione di pacchetto e interconnessi tramite router.

I **processi** servono ai dispositivi per comunicare.

Nel livello Trasporto si possono implementare due Tipi di Trasporto nella suite IP:

- TCP (Transmission Control Protocol) → orientato alla connessione e affidabile
- UDP (User Datagram Protocol) → senza connessione e non affidabile

Il livello di trasporto nella rete IP è implementato solo nei sistemi finali.

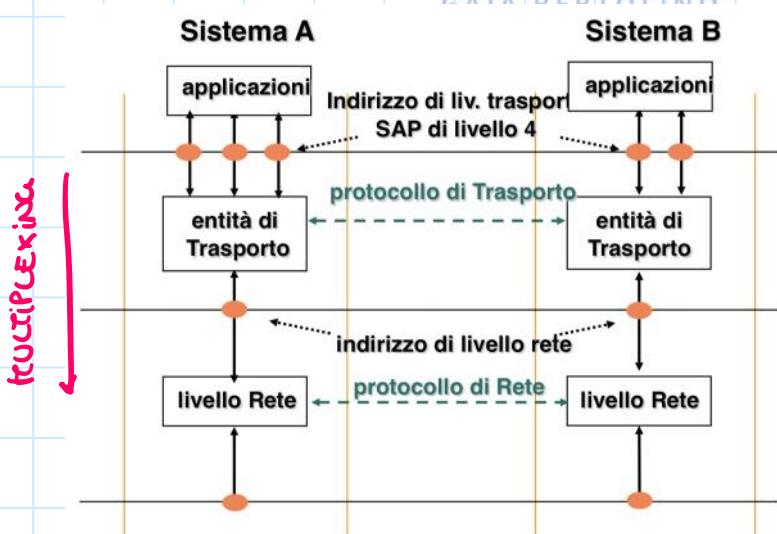


Il primo compito del livello di trasporto è quello di indirizzare i SAP (Service Access Point) dove i SAP sono i punti di accesso che un livello offre al suo superiore in modo che possa usufruire dei suoi servizi.

APPUNTI DI INGEGNERIA INFORMATICA

GAIA PERTOLINO

esempio



Leto sorgente si applica il **multiplexing** ovvero lo sussistamento dei dati da elaborare ai processi associati ad ogni livello. A livello destinazione invece la ricomposizione prende il nome di **demultiplexing**.

La ricostituzione prende il nome di **demultiplexing**.

Una **porta** è un SAP del livello 4 che si trova tra livello applicativo e di trasporto e serve ad identificare univocamente una specifica entità di destinazione.

L'indirizzo TCP / IP è dunque costituito da indirizzo IP e dal numero di porta. Tale indirizzo dunque un processo in esecuzione in un host.

Dunque lo strato IP si occupa del trasferimento dei dati fra reti interconnesse mentre lo strato di UDP o TCP dello sussistamento dei dati all'interno dell'host stesso utilizzando il numero di porta.

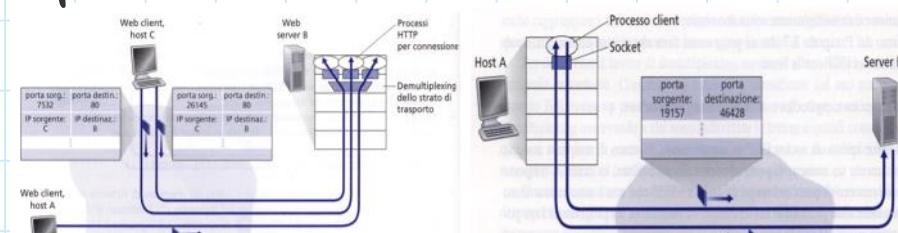
La componente "port" sarà contenuta nell'header del livello di trasporto.

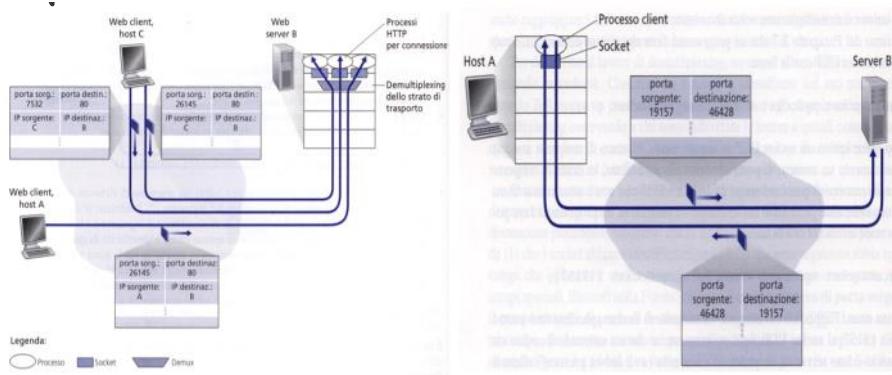
Le porte TCP e UDP possono essere di

- destinazione → detta server port individua il servizio richiesto
- sorgente → detta client port permette di attivare diverse connessioni allo stesso servizio

L'assegnamento delle porte è in parte universale (poche ben definite) o dinamica che avviene a seguito di una negoziazione. La porta è individuata tramite un numero a 16 bit. Le porte **well-known port** sono da 0 a 1023, da 1024 a 65535 vi sono le porte utente cioè utilizzate e scelte dall'applicativo client.

Esempio





UDP

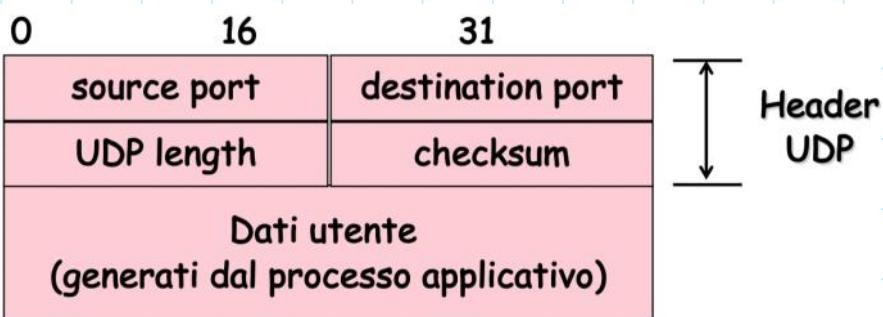
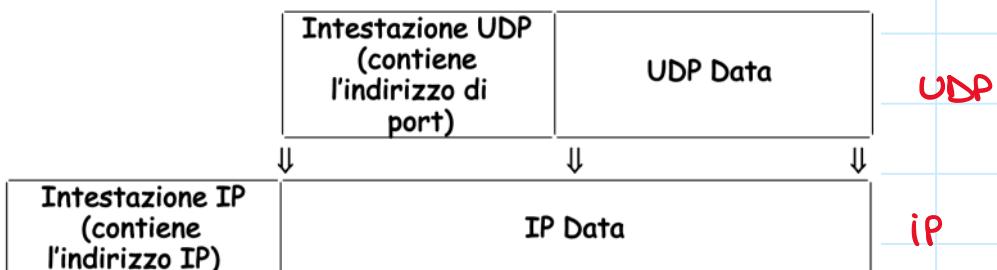
È un protocollo più semplice del TCP e collega livello applicativo e di trasporto.

A quelle di IP aggiungono il demultiplexing dei dati e una checksum (controllo d'errore) dell'header.

Non garantisce la consegna poiché non verifica la disponibilità di risorse prima di inviare, non ha meccanismi di ritrasmissione e non di recupero.

Impacchettamento:

APPUNTI DI INGEGNERIA



Controllo dell'errore

Controllo dell'errore

La checksum copre il datagramma UDP e il cosiddetto pseudo-header ovvero una parte dell'indirizzo IP.

Tuttavia non applica protocolli di controllo del flusso al fine di evitare problemi di congestione → controllo più generale del flusso

TCP

È un protocollo con connessione che fornisce un processo affidabile.

L'indirizzo TCP/IP è chiamato **socket**. Una connessione TCP è fornita da due socket. Dunque un socket distingue diversi processi su una stessa porta grazie alla parte IP (dunque un end point può avere attive più connessioni).

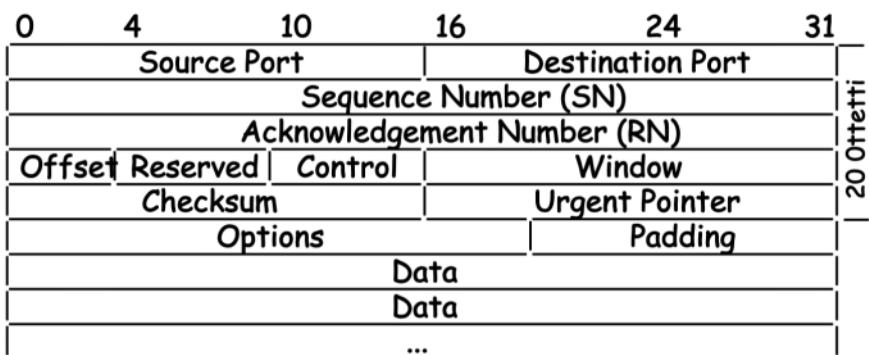
L'indirizzo TCP può essere assegnato dinamicamente ovvero.

La dimensione massiccia del segnale è specificata nel campo MSS ed è decisa all'inizio della connessione; tale campo non può essere maggiore di 64 kbyte poiché andrà poi a far parte degli altri livelli.

Il TCP usa dei buffer per immagazzinare i dati.

Formato cui fa dati TCP:

- L'header è lungo 20 byte se il campo opzioni non viene utilizzato. Il campo dati può essere vuoto (trame di acknowledgment, connessione, ecc.).



Source Port (16 bit): definisce l'indirizzo logico del processo sorgente dei dati

Destination Port (16 bit): definisce l'indirizzo logico del processo destinatario dei dati

Sequence Number (32 bit): numero di sequenza in trasmissione (SN); contiene il numero di sequenza del primo byte di dati contenuti nel segmento a partire dall'inizio della sessione (se SN=m ed il segmento contiene n byte il prossimo SN sarà pari a m+n)

Acknowledgement Number (32 bit): numero di sequenza in ricezione (RN); nei segmenti in cui il bit ACK è 1, contiene il numero di sequenza del prossimo byte che il ricevitore si aspetta di ricevere. Il meccanismo di riscontro usato è cumulativo, così l'ack di un SN X indica che sono stati ricevuti tutti i byte fino a X escluso X. In caso di connessioni interattive bidirezionali gli ack sono inviati in piggybacking (nei segmenti di risposta contenenti dati di utente). I numeri SN e RN vengono utilizzati per il controllo d'errore e di flusso

I numeri di sequenza e gli ack rendono affidabile la trasmissione TCP. Ad ogni byte di dati si assegna un numero di sequenza. In ogni segmento TCP si inserisce il numero di sequenza del primo byte di dati contenuto nel segmento (SN)

I segmenti in direzione opposta portano anche un numero di acknowledgement che è il numero di sequenza del successivo byte di dati da trasmettere atteso dal ricevitore

Quando il TCP trasmette un segmento dati, ne conserva una copia in una coda di ritrasmissione e fa partire un timer; quando riceve l'ack per quei dati, allora cancella il segmento dalla coda. Se l'ack non è ricevuto prima della scadenza del time-out, il segmento viene ritrasmesso

Così il TCP mantiene la corretta sequenza dei segmenti in ricezione; cioè prima di inviare una nuova sequenza di byte aspetta che la sequenza precedente venga riscontrata

Offset (4 bit): contiene il numero di parole di 32 bit contenute nell'intestazione TCP (da Source port a Padding). L'intestazione TCP non supera i 64 byte ed è un multiplo di 32

Reserved (6 bit): riservato per usi futuri, contiene zeri

Control bit (6 bit): i bit di controllo sono 6:

- » **URG:** vale 1 quando il campo Urgent Pointer contiene un valore significativo; è usato per indicare dati urgenti che vengono trasmessi al di fuori del controllo di flusso con un meccanismo di segnalazione end-to-end tra processi remoti
 - » **ACK:** vale 1 quando il campo Acknowledgement Number contiene un valore significativo
 - » **PSH:** vale 1 quando l'applicazione esige che i dati forniti vengano trasmessi e consegnati all'applicazione ricevente prescindendo dal riempimento dei buffer allocati fra applicazione e TCP e viceversa (di solito il riempimento dei buffer scandisce la trasmissione e la consegna dei dati)
 - **RST:** vale 1 quando un malfunzionamento impone il reset della connessione
 - **SYN:** indica l'inizio della connessione; vale 1 solo nel primo segmento inviato durante la fase di sincronizzazione fra le entità TCP (3-way handshaking)
 - **FIN:** indica la fine della connessione; vale 1 quando la sorgente ha esaurito i dati da trasmettere
- » **Window (16 bit):** larghezza della finestra per il controllo di flusso; il TCP ricevente riporta al TCP trasmittente il valore di una "window", che contiene il numero di byte che, a cominciare dal valore del campo Acknowledgement Number, il TCP ricevente è disposto a ricevere. Il controllo di flusso è orientato al byte
- » **Checksum (16 bit):** contiene la sequenza che permette al TCP ricevente di verificare la correttezza del segmento; protegge l'intero segmento più alcuni campi dell'header IP, cioè uno pseudo-header di 96 bit

Urgent Pointer (16 bit): indica i dati urgenti all'interno del segmento; contiene il numero di sequenza dell'ultimo byte di dati che devono essere consegnati con urgenza al processo ricevente. Tipicamente sono messaggi di controllo che esulano dalla comunicazione in senso stretto. A tale traffico ci si riferisce di solito con il nome di out-of-band

Options (lunghezza variabile): sono presenti solo raramente; le più note sono End of Option List, No-operation e Maximum Segment Size (di default è 536 byte)

- » MSS serve per comunicare al TCP trasmittente la dimensione massima del segmento accettabile in ricezione; il campo è inviato soltanto nella richiesta iniziale di connessione (cioè, nei segmenti col bit SYN settato); se non si usa questa opzione è ammessa una qualsiasi dimensione del segmento

Padding (lunghezza variabile): contiene sempre degli zeri. Serve come riempitivo per far sì che l'intestazione abbia una lunghezza multipla di 32 bit

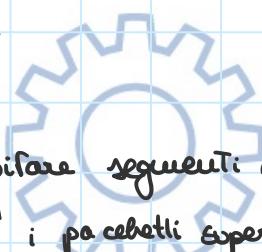
La checksum è il complemento a 1 del complemento a 1 della somma di tutte le word da 16 bit nell'header e nel testo; se un segmento contiene un numero dispari di byte, l'ultimo byte è riempito di zeri a destra (padding) per formare una word di 16 bit. Il pad non è trasmesso come parte del segmento

TCP Length è la lunghezza del TCP header più la lunghezza dei dati in byte (questa quantità non viene esplicitamente trasmessa, ma è calcolata), e non tiene conto dei 12 byte dello pseudo-header

0	8	16	24	31
Source IP address				
Destination IP address				
Padding	Protocol	TCP length		

Iustaurazione di una connessione

Una connessione richiede un meccanismo di sincronizzazione in tre fasi detto **three-way handshake**.



L'obiettivo delle connessioni è evitare segmenti duplicati (verificando il numero di sequenza) e che "muoicano" i pacchetti aspettato in certo tempo.

Ogni host dunque utilizza un clock che misura i byte inviati in quanto regola quale è la velocità di trasferimento dei dati. Tale clock deve funzionare anche quando l'host va in crash.

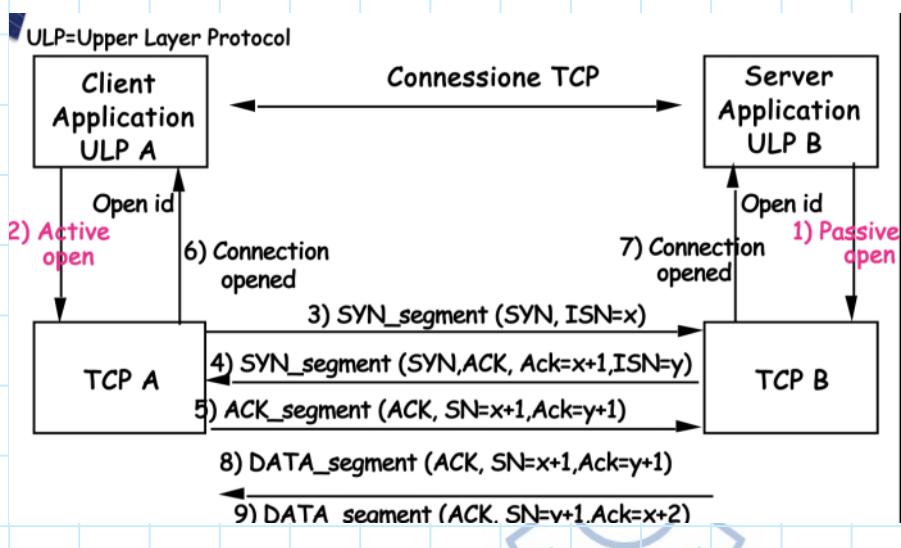
Il numero di bit del controllore deve essere maggiore del numero di bit dei valori di sequenza, dunque 32 bit bastano.

Il numero iniziale di sequenza (ISN) è casuale per dare una piccola modalità di sicurezza.

Il tempo di vita di un segmento (detto TSL) deve essere minore del ciclo di clock (che come detto dipende dalla velocità di rete). Tipicamente l'TSL è minore di 1,7 ore per cui si può assumere che l'ISN sia unico.

La sincronizzazione iniziale consiste nell'invio da entrambe le parti del proprio ISN e della ricezione di Acknowledgment.

- 1) A → B SYN
 - 2) A ← B ACK
 - 3) A ← B SYN
 - 4) A → B ACK
- combinati in un unico messaggio
- ue conseguono un handshake a tre fasi

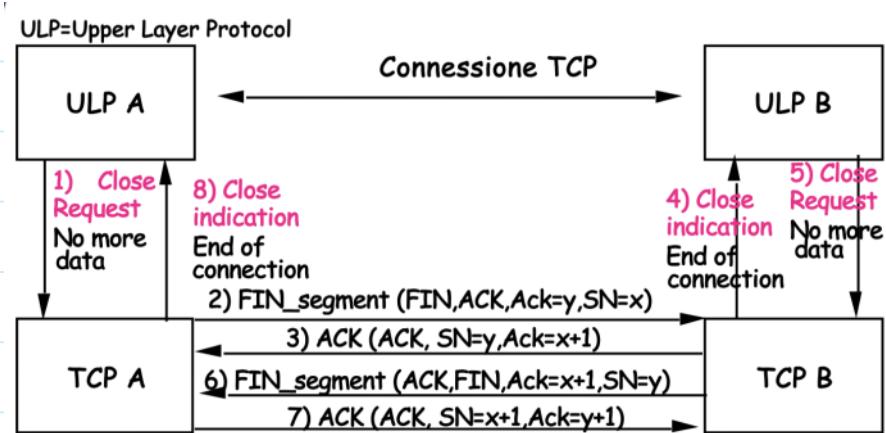


La scelta dell'MSS dipende dalla memoria a disposizione della TCP e dalla dimensione dell'MTU.

es. $MSS = 536 \text{ byte} + 40 \text{ byte di header IP + TCP} = \text{datagramma IP di 576 byte.}$

Il rilascio di una connessione avviene invece attraverso segnalazioni di fine connessione, non per forza è di tipo three-phase closing perché la chiusura fra due host può essere asincrona.

- es. 1) A → B FIN
- 2) A ← B ACK
- 3) A ← B FIN
- 4) A → B ACK



Se TCP utilizza riscontri positivi basati su finestre ovvero finestre di byte che possono essere inviati senza ricevere riscontro (e in TCP anche il numero di byte fuori sequenza che possono essere ricevuti).

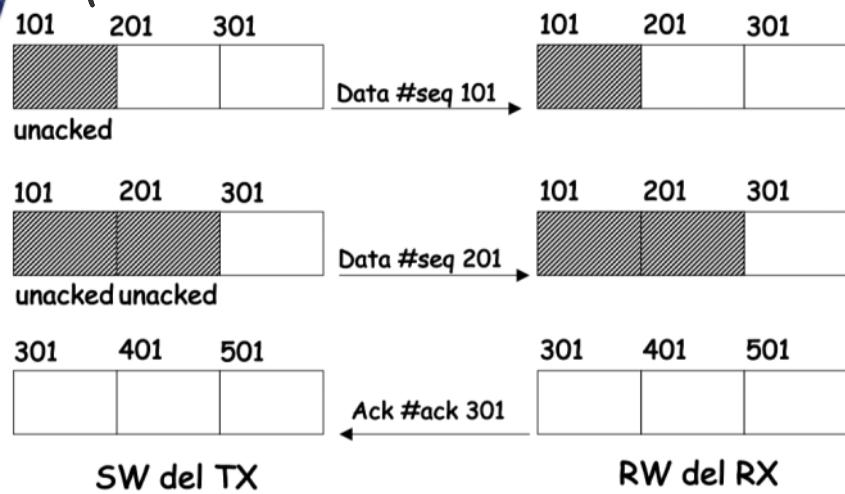
La dimensione della finestra è decisa dal destinatario ed è comunicata al mittente "oggi" volta che viene mandato un segmento per cui ha una dimensione variabile e tale meccanismo è detto di TCP Advertised.

Controllo di flusso

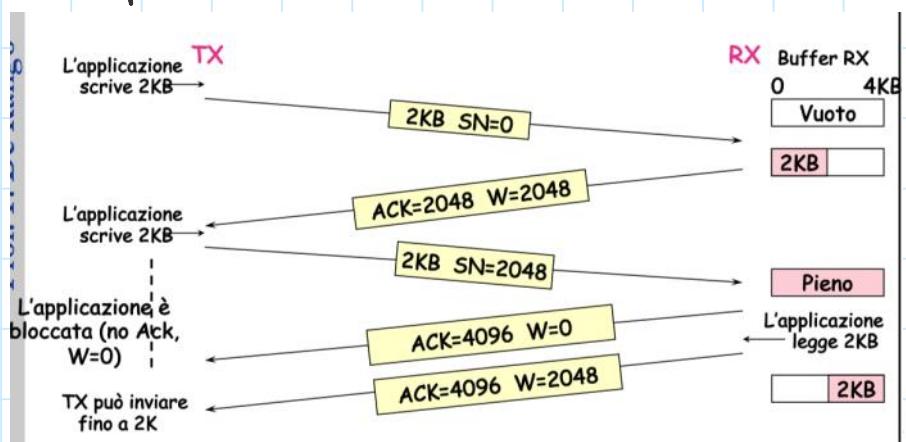
Involge solo sorgente e destinazione (non i nodi intermedi) e mira a limitare il flusso di dati trasmessi. Cioè avviene verificando i dati mandati e modificando opportunamente la finestra del ricevente in modo da rispettare la disponibilità di risorse elaborative.

- Un riscontro (ACK Number=x e Window=W) significa che:
 - sono riscontrati tutti gli ottetti ricevuti fino a quello numerato con X-1
 - il trasmittente è autorizzato a trasmettere, senza attendere altri riscontri, fino a ulteriori W ottetti, ovvero fino all'ottetto numerato con x+W-1
- Solitamente la RW ha un'ampiezza pari a n*MSS, dove n è un numero intero per evitare che il TCP trasmittente esegua una segmentazione poco efficiente
 - se la finestra di ricezione è pari a 301 byte e la MSS 100 byte, il TCP trasmittente sarebbe indotto a formare segmenti da 1 byte, a cui aggiungere l'header del TCP e di IP (tipicamente di 20 byte ciascuno). Se a ciò si aggiunge l'header di livello 2 (8 byte nel caso di PPP), per inviare un'informazione di 1 byte è necessario inviarne 49

Esempio 1

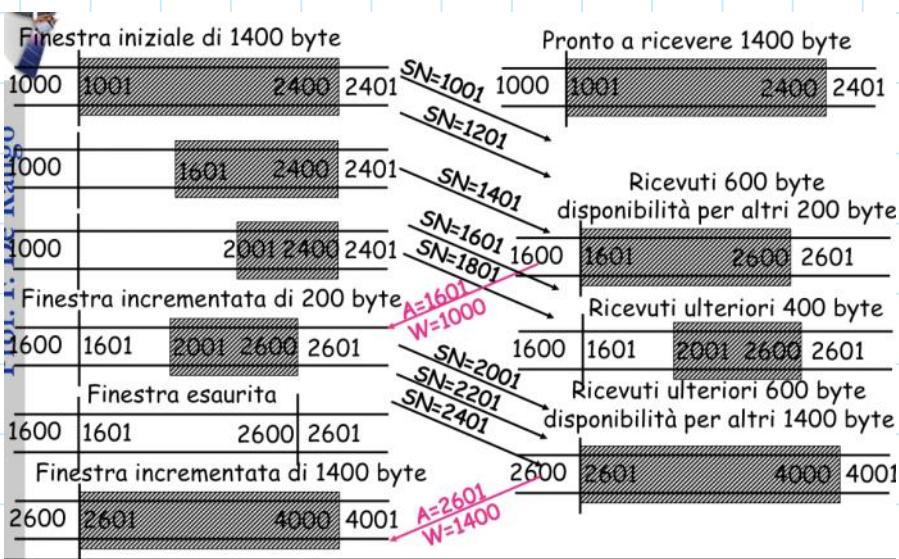


Esempio 2



INFORMATICA
GAIA BERTOLINO

Esempio 3



LIVELLO DATA LINK - Protocollo:

- SIMPLEX STOP-AND-WAIT

Venne inviato un frame e, prima di mandare il successivo, si aspetta un frame di acknowledge.

Presenta un timer alla scadenza del quale rimanda il pacchetto.

Dunque, i pacchetti devono avere un numero di sequenza che li contraddistingua

→ ARQ / PAR

- A FINESTRA SCORREVOLE

La finestra di invio è il numero di frame che la sorgente può inviare.

La finestra di ricezione è il numero di frame che la destinazione può ricevere.

Utilizza il piggybacking ovvero un frame in uscita contiene l'ack del precedente frame ricevuto.

- a 1 bit

Il campo di ack contiene il numero dell'ultimo frame ricevuto senza errore

- GO-BACK-N

La destinazione scatta tutti i frame successivi all'errore.

Quando la sorgente va in timeout a causa della mancanza di ack, invia tutti i frame successivi a quello con l'errore.

Poò fare uso di ack cumulativi che indicano l'arrivo dei frame fino a quello di ack

- a più bit

Vengono inviati vari frame secondo la tecnica di pipelining dando così modo di procedere all'invio ricevendo nel frattempo gli ack dei frame precedenti.

- SELECTIVE REPEAT

La destinazione scatta solo il frame con l'errore mentre salva gli altri in un buffer in attesa di analizzarli.

Quando la sorgente va in timeout a causa di un mancato ack rimanda solo il pacchetto con l'errore.

La selective repeat può fare uso di un NAK (ack negativo)

Poò fare uso di ack cumulativi e si comporta come il go-back-n in caso di perdita o errore ovvero il destinatario scatta tutti i pacchetti dopo quelli di errore.

Rispetto agli altri protocolli non è detto che riceva i frame in ordine.



SOTTO LIVELLO MAC

- ALOHA

- ALOHA PURO

Un trasmettitore manda il proprio frame al computer centrale: in quale, una volta ricevuto lo riemanda in broadcast o manda un ack in modo che il trasmettitore possa verificare la correttezza. Se uno è corretto o si perde l'ack allora viene riemandato dopo un tempo casuale. Dunque il successo è determinato dalla sovrapposizione (anche parziale) o meno dei frame. Il tempo è considerato come continuo e cioè gli invii possono avvenire in qualsiasi momento.

- **SLOTTED ALOHA**

Elimina la presenza di collisioni parziali in quanto il tempo viene discretizzato in slot e una stazione può trasmettere solo all'inizio di uno slot.

- **CSMA**

I trasmettitori si mettono in ascolto sul canale per verificare se è libero o meno.

- **CSMA 1-persistent**

Appena il canale si libera, manda il frame.

È soggetto a collisioni se più stazioni trasmettono insieme o c'è un ritardo a comunicare che il canale è occupato.

- **CSMA p-persistent**

Appena il canale si libera, manda il frame con una probabilità p e riserva l'invio di un eventuale futuro frame con una probabilità di $1-p$.

- **CSMA non persistent**

Non manda il frame appena si libera il canale ma dopo un tempo casuale.

- **CSMA con rilevamento delle collisioni**

L'hardware del trasmettitore rimane in ascolto e verifica che il messaggio non si sia alterato (rilevamento di collisione). In caso affermativo blocca la trasmissione.

È simile all'ALOHA

APPUNTI DI INGEGNERIA
INFORMATICA

GAIA BERTOLINO

Formulario

sabato 16 luglio 2022 15:23

• TEMPO DI RITARDO

Dipende dalla distanza delle stazioni

$$TR = \frac{\text{distanza}}{\text{velocità}}$$

M_{TU} → dimensione massima totale RETE
M_S → dimensione payload TRASPORTO
PDU → dimensione payload RETE
SDU → dimensione payload TRASPORTO

• TEMPO DI TRASMISSIONE

$$T_p = \frac{\text{dimensione Frame (in bit)}}{\text{Capacità (A,B)}}$$

Nel caso dell'ack, dipende dalla dimensione minima dell'header di datalink

• DURATA TOTALE DI TRASMISSIONE

Tiene in considerazione la divisione in segmenti del payload trasporto.

$$n = \left\lceil \frac{\text{flusso dati}}{MSS} \right\rceil$$

MSS → payload a livello trasporto

$$D = n \cdot T_c$$

APPUNTI DI INGEGNERIA INFORMATICA

GAIA BERTOLINO

• CANALE CON PERDITE

- Numero di pacchetti trascorsi

$$x = \left\lfloor \frac{T_{ON}}{T_c} \right\rfloor$$

- Tempo fisso di trasmissione

$$T_F = T_{OFF} - x \cdot T_c \rightarrow \text{tempo di ciclo}$$

- Trasmissioni perse

$$y = \left\lceil \frac{T_{OFF}}{T_O} \right\rceil \rightarrow \text{tempo di timeout}$$

- Tempo fisso alla trasmissione corretta

$$T = T_F + y \cdot T_O$$

- Tempo totale

$$D = n \cdot T_c + y \cdot T_O$$

• NUMERO DI FRAME TRASMESSI

Bisogna tenere in considerazione il payload del trasporto che definisce gli effettivi dati al netto degli header

- PORTATA MEDIA MASSIMA

$$P = w_s \frac{L+4t}{C} \geq T_c$$

Soluzione con continuità

- TEMPO DI TIMEOUT

$$T_0 = 2 \cdot T_c$$

- ALOHA PURO

- TEMPO DI RITRASMISSIONE (dalla fine del pacchetto)

$$z = \text{Somma Cifre Tazio} * \text{numero Collisione} + \text{Tempo Tazio Trama}$$

- THROUGHPUT MEDIO

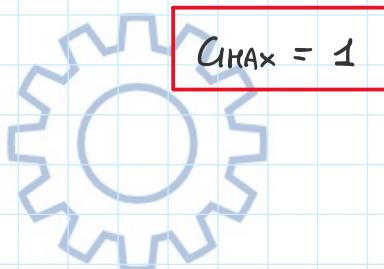
$$S = G \cdot e^{-2G}$$

- THROUGHPUT MASSIMO

$$S = \frac{1}{2e}$$

- FRAME INVIATI

$$G = \lambda \cdot T \quad \text{con} \quad T = \frac{L}{C}$$



APPUNTI DI INGEGNERIA

INFORMATICA

GAIA BERTOLINO

- ALOHA A SLOT

- THROUGHPUT MEDIO

$$S = G \cdot e^{-G}$$

- THROUGHPUT MASSIMO

$$S = \frac{1}{e}$$

- CSMA

- VELOCITÀ DI PROPAGAZIONE

$$v_p = \frac{2}{3} 300000 \text{ m/s} = 200000 \text{ m/s}$$

- CONDIZIONE DI FUNZIONAMENTO

$$T \geq 2 \cdot \tau \quad \text{dove} \quad \tau = \frac{d}{v_p} \quad \text{e} \quad \tau = \frac{L}{C}$$

- THROUGHPUT MEDIO

$$S = \frac{G \cdot e^{-\alpha G}}{\alpha} \quad \text{dove} \quad \alpha = \frac{d}{v_p}$$

$$S = \frac{G \cdot e^{-\alpha G}}{G(1+2\alpha) + e^{-\alpha G}} \quad \text{dove } \alpha = \frac{\zeta}{T}$$

- INSTRADAMENTO

- CLASSI

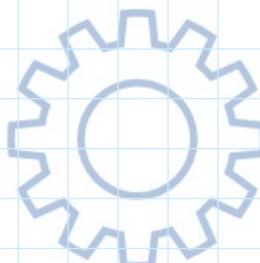
Classe	Bit Iniziali	Net_Id	Host_Id	"reti logiche" disponibili	"indirizzi" disponibili
A	0	7 bit	24 bit	128	16 777 216
B	10	14 bit	16 bit	16 384	65 536
C	110	21 bit	8 bit	2 097 152	256
D	1110			indirizzi multicast: 28 bit indirizzi possibili: $2^{32}/16=268\ 435\ 456$	
E	11110			riservati per usi futuri e ricerca: 27 bit indirizzi possibili: $2^{32}/32=134\ 217\ 728$	

- TIMEOUT

$$T_0 = 2 \cdot RTT$$

- MSS INVIATI

$$N = \left\lceil \frac{F}{MSS} \right\rceil$$



APPUNTI DI INGEGNERIA INFORMATICA

GAIA BERTOLINO

- FINESTRA SORGENTE

$$W_S = \left\lfloor \frac{C}{frame} \right\rfloor$$

- UPPER BOUND CWND

$$MAXWIN = min(W_C, W_S)$$

- TEMPO PER RAGGIUNGERE LA MAXWIN

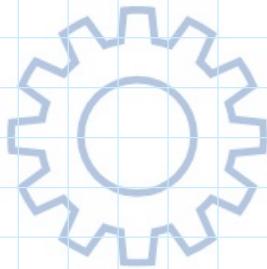
$$T_{SS} = \log_2 (MAXWIN) \quad [RTT]$$

- MSS TRASFERITI

$$B_{SS} = (2^{T_{SS}+1} - 1) \quad [MSS]$$

- ULTERIORI RTT

$$\left[\frac{N - B_{SS}}{MAXWIN} \right]$$



APPUNTI DI INGEGNERIA INFORMATICA

GAIA BERTOLINO

Esercitazione

sabato 16 luglio 2022 16:36

- CLASSLESS di CLASSE C

$L_1 : 115 \text{ host}$

$L_2 : 28 \text{ host}$

$L_3 : 59 \text{ host}$

$L_4 : 127 \text{ host}$

$$b = 200 \cdot 11 \cdot 15 \cdot 0$$

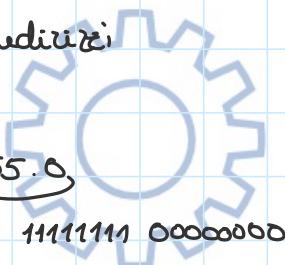
Ordine: $L_4 \rightarrow L_1 \rightarrow L_3 \rightarrow L_2$

(L_4)

$$I = b + bb + R_4 + 127 = 130 \text{ host}$$

Quindi $2^7 = 128$ indirizzi

MASK: $\underbrace{255.255.255.0}_{\text{11111111 11111111 11111111 00000000}}$



$$b = 200 \cdot 11 \cdot 15 \cdot 0 / 24$$

$$bb = 200 \cdot 11 \cdot 15 \cdot 255 / 24$$

$$R_4 = 200 \cdot 11 \cdot 15 \cdot 1 / 24$$

Presa:

$$200 \cdot 11 \cdot 15 \cdot 1$$

$$\begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \quad \begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \quad \begin{array}{cccccccc} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array} \quad \begin{array}{cccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

indirizzo base

MASK: $255.255.255.0$

(L_1) $I = b + bb + R_1 + 115 = 118$

quindi $2^7 = 128$ indirizzi

7 bit per l'host

$$b = 200 \cdot 11 \cdot 16 \cdot 0 / 25$$

25 per la rete
 $\underbrace{8.8.8.1}_{\text{g.g.g.g}}$

$$bb = 200 \cdot 11 \cdot 16 \cdot 127 / 25$$

$$R_4 = 200 \cdot 11 \cdot 16 \cdot 1 / 25$$

$$\text{MASK} : 255 \cdot 255 \cdot 255 \cdot 128$$

11111111 10000000
 255 128

(L₃) $I = b + bb + R_3 + R_4 + 59 = 63$

Quindi $2^6 = 64$ indirizzi

$$b = 200 \cdot 11 \cdot 16 \cdot 128 / 26$$

$$bb = 200 \cdot 11 \cdot 16 \cdot 191 / 26$$

$$R_3 = 200 \cdot 11 \cdot 16 \cdot 129 / 26$$

$$R_4 = 200 \cdot 11 \cdot 16 \cdot 130 / 26$$

$$\text{MASK} : 255 \cdot 255 \cdot 255 \cdot 192$$

$$6 \times \text{host} = 32 - 6 \text{ net} = 26 \text{ net}$$

$$8 \cdot 8 \cdot 8 \cdot 2$$

$$11000000 = 2^7 + 2^6 = 192$$

(L₂) $I = b + bb + R_2 + 28 = 31$

Quindi $2^5 = 32$ indirizzi

$$b = 200 \cdot 11 \cdot 16 \cdot 192 / 27$$

$$bb = 200 \cdot 11 \cdot 16 \cdot 223 / 27$$

$$R_2 = 200 \cdot 11 \cdot 16 \cdot 193 / 27$$

$$\text{MASK} : 255 \cdot 255 \cdot 255 \cdot 224$$

$$5 \text{ host}$$

$$32 - 5 = 27 \text{ ind.}$$

$$8 \cdot 8 \cdot 8 \cdot 3$$

$$11100000$$

224

Dividi: 200.11.18.0 - 200.11.18.235 in blocco da 64

APPUNTI DI INGEGNERIA

Subnetting fisso = ogni subnet avrà la stessa divisione
 quindi in questo caso essendo di classe C una subnet
 corrisponde ad un 255.255.255....

MASCHERA VARIABILE

(L₄) $I = b + bb + R_4 + 127 = 130 \text{ host}$

Quindi $2^8 = 256$ indirizzi

$$\text{MASK} : \underbrace{255 \cdot 255 \cdot 255 \cdot 0}_{\dots}$$

11111111 11111111 11111111 00000000

$$b = 200 \cdot 11 \cdot 15 \cdot 0 / 24$$

$$B = 200 \cdot 11 \cdot 15 \cdot 255 / 2^6$$

$$R_4 = 200 \cdot 11 \cdot 15 \cdot 1 / 2^6$$

Presa:

$$200 \cdot 11 \cdot 15 \cdot 1$$

$$\begin{array}{r} 11111111111111111111111100000001 \\ 11111111111111111111111100000000 \\ \hline \end{array}$$

indirizzo base

MASK: 255.255.255.0

(L1)

$$I = b + bb + R_1 + 115 = 118$$

quiudi $2^7 = 128$ indirizzi

$$b = 200 \cdot 11 \cdot 16 \cdot 0 / 2^5$$

$$bb = 200 \cdot 11 \cdot 16 \cdot 127 / 2^5 - 256$$

$$R_4 = 200 \cdot 11 \cdot 16 \cdot 1 / 2^5$$

MASK: 255.255.255.128

7 bit per l'host

25 per la rete

$$\begin{array}{r} 1111111100000000 \\ \text{255} \quad \text{128} \\ \hline 8.8.8.1 \end{array}$$

(L3)

$$I = b + bb + R_3 + R_4 + 59 = 63$$

quiudi $2^6 = 64$ indirizzi $6 \times \text{host} = 32 - 6 \text{ net} = 26 \text{ net}$

$$b = 200 \cdot 11 \cdot 17 \cdot 0 / 2^6$$

$$bb = 200 \cdot 11 \cdot 17 \cdot 63 / 2^6$$

$$R_3 = 200 \cdot 11 \cdot 17 \cdot 1 / 2^6$$

$$R_4 = 200 \cdot 11 \cdot 17 \cdot 2 / 2^6$$

MASK: 255.255.255.192

8.8.8.2

$$11000000 = 2^7 + 2^6 = 192$$

(L2)

$$I = b + bb + R_2 + 28 = 31$$

quiudi $2^5 = 32$ indirizzi

$$b = 200 \cdot 11 \cdot 18 \cdot 0 / 2^7$$

$$bb = 200 \cdot 11 \cdot 18 \cdot 31 / 2^7$$

$$R_2 = 200 \cdot 11 \cdot 18 \cdot 1 / 2^7$$

MASK: 255.255.255.224

Esercizi algoritmo di DIJKSTRA

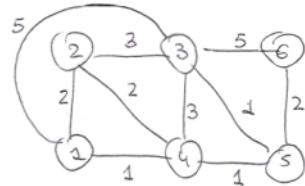
lunedì 18 luglio 2022 12:00



E07 -
Algoritmo...

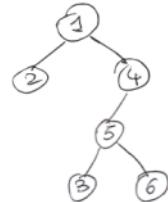
$$\begin{aligned} k=1 \\ T = \{1\} \\ L(2) = w(1,2) = 2 \\ L(3) = w(1,3) = 5 \\ L(4) = w(1,4) = 1 \\ L(5) = w(1,5) = \infty \\ L(6) = w(1,6) = \infty \end{aligned}$$

$$\begin{aligned} k=2 \\ L(x) = \min_{v \in V \setminus T} L(v) \Rightarrow L(4) \Rightarrow x=4 \\ T = \{1, 4\} \\ L(2) = \min [L(2), L(4) + w(4,2)] = 2 \\ L(3) = \min [L(3), L(4) + w(4,3)] = 4 \\ L(5) = \min [L(5), L(4) + w(4,5)] = 2 \\ L(6) = \min [L(6), L(4) + w(4,6)] = \infty \end{aligned}$$



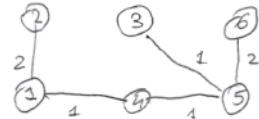
$$\begin{aligned} k=3 \\ L(x) = \min_{v \in V \setminus T} L(v) = L(2) \Rightarrow x=2 \\ T = \{1, 4, 2\} \\ L(3) = \min [L(3), L(2) + w(2,3)] = 4 \\ L(5) = \min [L(5), L(2) + w(2,5)] = 2 \\ L(6) = \min [L(6), L(2) + w(2,6)] = \infty \end{aligned}$$

$$\begin{aligned} k=4 \\ L(x) = L(5) \quad x=5 \\ T = \{1, 4, 2, 5\} \\ L(3) = \min [L(3), L(5) + w(5,3)] = 3 \\ L(6) = \min [L(6), L(5) + w(5,6)] = 4 \end{aligned}$$



$$\begin{aligned} k=5 \\ L(x) = L(3) \quad x=3 \\ T = \{1, 4, 2, 5, 3\} \\ L(6) = \min [L(6), L(3) + w(3,6)] = 4 \end{aligned}$$

$$\begin{aligned} k=6 \\ L(x) = L(6) \quad x=6 \\ T = \{1, 4, 2, 5, 3, 6\} \end{aligned}$$



**APPUNTI DI INGEGNERIA
INFORMATICA**

GAIA BERTOLINO