GIOVANNI PISTORIO	27-11-2024	Compito scritto in classe
GIOVANNI PISTORIO	06-11-2024	Compito in classe.
GIOVANNI PISTORIO	30-10-2024	Livello 4: Rilascio di una connessione, problema delle due armate, soluzioni e gestione errori.
GIOVANNI PISTORIO	23-10-2024	Livello 4: indirizzamento, three-way handshaking, attivazione della connessione e possibili errori.
GIOVANNI PISTORIO	16-10-2024	Livello 4: primitive, incombenze e apertura connessione.
GIOVANNI PISTORIO	09-10-2024	Controllo e correzione elaborati per casa.
GIOVANNI PISTORIO	02-10-2024	Conclusione ripasso modello OSI e architettura TCP/IP.
GIOVANNI PISTORIO	25-09-2024	Modello OSI ed in particolare i primi 3 livelli.
GIOVANNI PISTORIO	17-09-2024	Introduzione al programma di quinta.

[LIVELLO 4 → TRASPORTO]

TCP / UDP (?)

=> [PROTOCOLLO] [INSIEME DI REGOLE]

[TCP = > AFFIDABILE
→ CONNECTION - ORIENTED

UDP = > NON AFFIDABILE
→ CONNECTION - LESS]

[QOS = QUALITY OF SERVICE
(QUALITÀ)] → [BANDWIDTH = BANDA DATI
THROUGHPUT = QUANTI DATI MANDATI
DELAY = RITARDO
JITTER (SALTI DEL SEGNALE)]

PARAMETRI

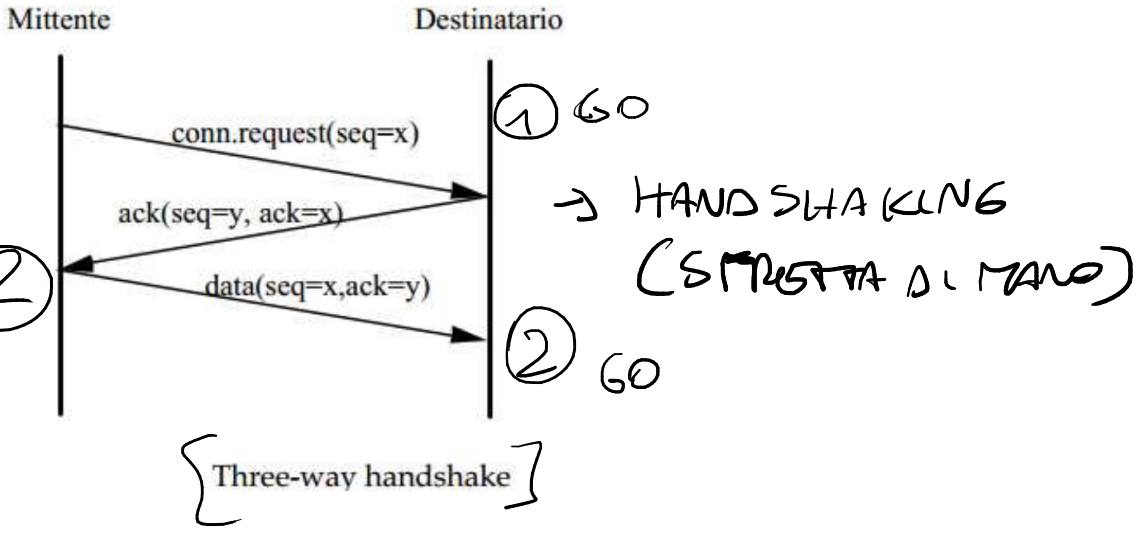
INDIRIZZAMENTO — PORTA
└ IP ADDRESS

→ {IP ADDRESS : PORTA}

[192.168.1.4 | 80]

(IP V4) ↑ IP CLASSE C ↑ TCP

↑
32b = 2^{32} 128 → 2^{128} .... (IPV6)



→ HANDSHAKING (STRETTA DI MANO)

1W - (2)

[1] MANDO PACCHETTO (1)

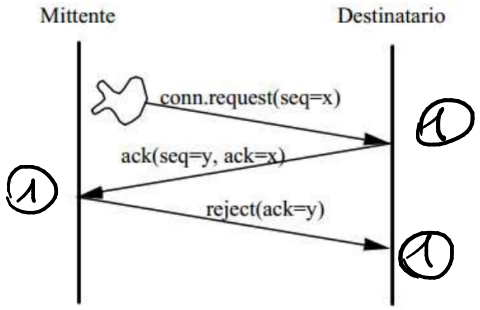
[2] RISPONDO AL PACCHETTO (1) CON UN ACKNOWLEDGEMENT (ACK) → VV

E DICO PARTI CON (2)

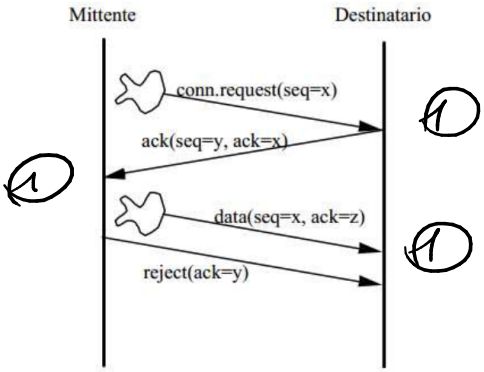
[3] MANDA PACCHETTO (2)

PROBLEMI

DUPLICAZIONE (PACCHETTI FUORI ORDINE OPPURE MANDATI DUE VOLTE)



Duplicato della richiesta di attivazione



Duplicati della richiesta di attivazione e del primo TPDU dati

↳ PACCHETTO

TCP → (1) – (2) – (3) – (4) (ORDINATI)

Primitive di definizione del servizio. Definiscono il modo di accedere ai servizi del livello:

Primitiva	TPDU spedito	Note
accept()	-	Si blocca finché qualcuno cerca di connettersi
connect()	conn.request	Cerca di stabilire una connessione
send()	dati	Invia dei dati
receive()	-	Si blocca finché arriva un TPDU
disconnect()	disconn.request	Chiede di terminare la connessione

HOST H_1
H_1.CONN()
H_2.SEND()
H_3..
ASCOLTO();

Esempio di frammenti di codice di un'applicazione client-server:

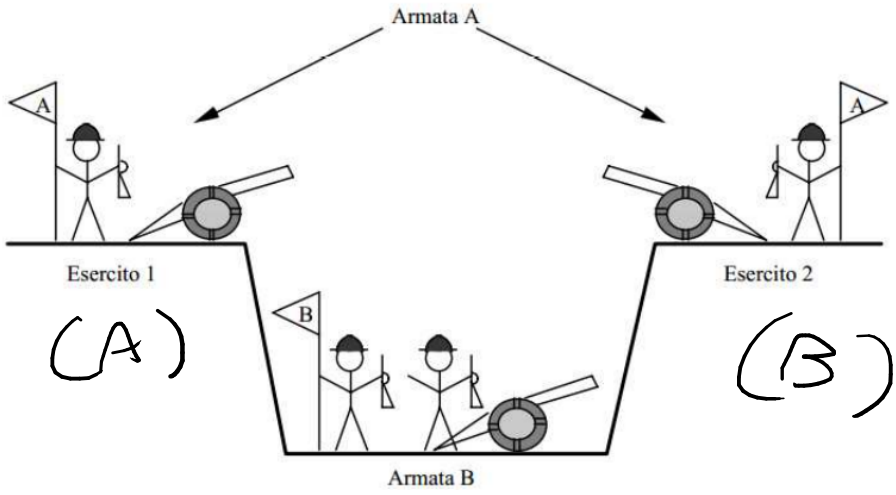
CLIENT	SERVER
...	...
connect()	accept()
receive()	send ()
send()	receive()
...	...
disconnect()	

https://www.geeksforgeeks.org/tcp-server-client-implementation-in-c/

[C] – CODE EXAMPLE

[PROBLEMA DEI DUE AMANTI!]

Purtroppo, le cose non sono così semplici: c'è un problema famoso in proposito, il **problema delle due armate**:



(A) (B)

- DUE ESERCITI
- PER VINCERE, DEVONO ATTACCARE SIMULTANEAMENTE
- COME COORDINARE (!)

> Il dilemma si manifesta così: il primo generale invia un messaggio dicendo "Attacchiamo all'alba". Ma non può essere sicuro che il messaggio sia arrivato, quindi ha bisogno di una conferma. Il secondo generale riceve il messaggio e invia una conferma, ma anche lui non può essere certo che la sua conferma sia arrivata. Il primo generale dovrebbe quindi inviare una conferma della conferma, e così via, creando un ciclo infinito.

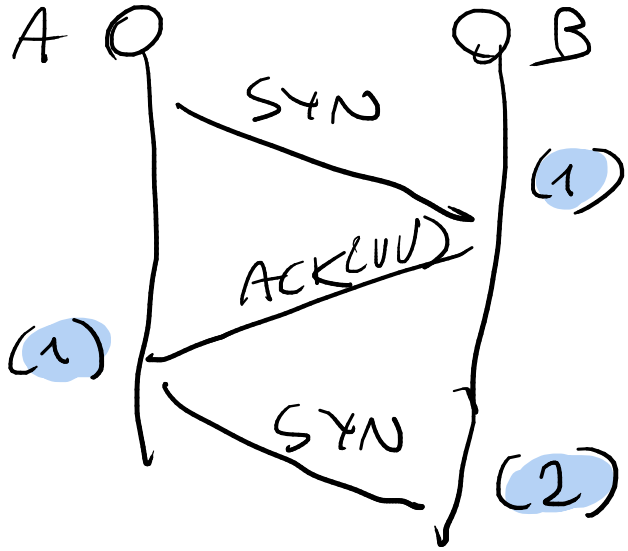
(A) PARTE ... (B) PARTE ...

MA HANNO BISOGNO DI CONFERME!

(STESSO PROBLEMA!)

/

SOLUZIONE: HANDSHAKE A 3 VIE!



→ TRASMISSIONE ORDINATA!

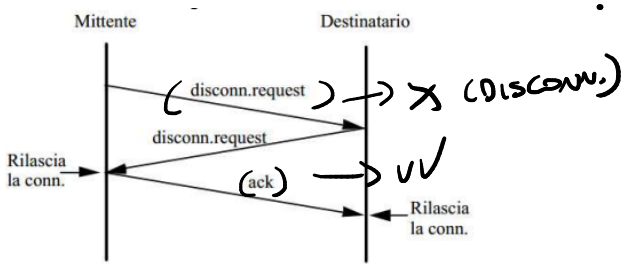
ALTRA SOLUZIONE: TIMEOUT!

PROBLEMI:

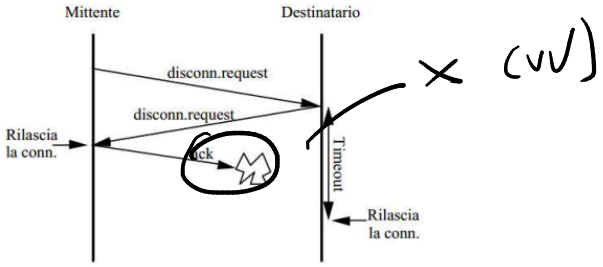
- QUANTO TEMPO CI VUOLE?
- SIAMO SICURI CHE BASTI?

PERDO UN ACK → [FAST RETRANSMIT]

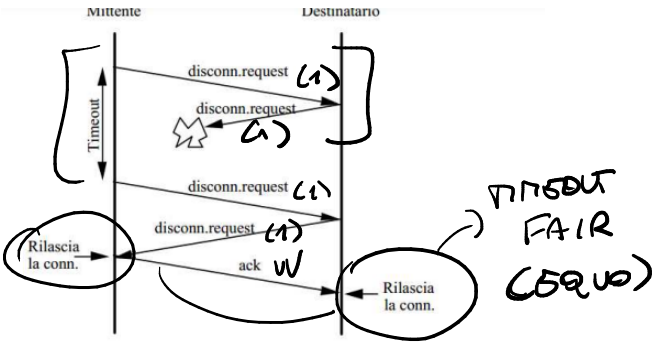
RELASE ↓ CONN.



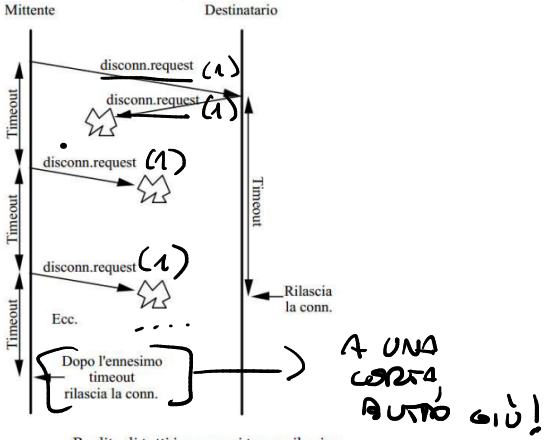
Rilascio concordato di una connessione transport



Perdita dell'ack durante il rilascio della connessione



Perdita della risposta alla proposta di chiusura durante il rilascio della connessione



Perdita di tutti i messaggi tranne il primo