Attivazione di una connessione

La creazione di una connessione fra due utenti del livello di trasporto non è un problema banale. Supponiamo, ad esempio, che l'utente trasmettitore non riceva il riscontro per un pacchetto: esso sarà allora costretto ad inviare un duplicato. Se una rete utilizza datagrammi può accadere, invece, visto che ogni pacchetto può seguire percorsi diversi, che un pacchetto intermedio rimanga invischiato nel traffico della rete e raggiunga l'utente in ricezione molto tempo dopo. In alcune applicazioni questa situazione può avere effetti disastrosi. Supponiamo ad esempio che si tratti di una connessione dell'utente di una banca per richiedere che dei fondi vengano trasferiti su un conto per un pagamento. Se dei duplicati dell'ordine di trasferimento dei fondi giungono al destinatario questo penserà che si tratti di una nuova richiesta di effettuare un pagamento.

Il problema fondamentale è dunque l'eliminazione dei duplicati che appaiono in ritardo. Un primo metodo è quello di creare indirizzi TSAP diversi per ogni connessione. Quando viene terminata una connessione questi indirizzi vengono abbandonati. In tal caso un duplicato che continua a circolare sulla rete non avrebbe la possibilità di raggiungere il destinatario dopo che la connessione è terminata.

Un altro modo è quello di contrassegnare ogni connessione con un identificatore. Quest'identificatore viene inserito in ogni TPDU. Le entità di trasporto devono avere, naturalmente, un registro delle connessioni terminate e degli identificatori ad esse associati. In tal modo quando giunge un pacchetto ritardatario appartenente ad una connessione già terminata esso viene scoperto grazie al fatto che esso presenta un

identificatore non più utilizzato. I problemi di questo sistema sono la necessità da parte delle entità di trasporto di tenere questi registri che possono divenire facilmente di grandi dimensioni.

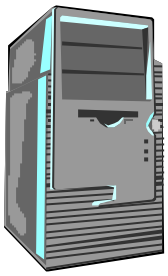
Metodi alternativi, invece di tentare di riconoscere un pacchetto appartenente ad una connessione già terminata, cercano di limitare il tempo di vita di un pacchetto in modo che non possa continuare a circolare per un tempo indefinito sulla rete.

Un primo metodo può essere, ad esempio, quello di imporre vincoli ad una rete. Una possibilità è quella di impedire che un pacchetto possa percorrere percorsi chiusi lungo la rete.

Un secondo metodo può essere quello di dotare il pacchetto di un contatore di tappa che s'incrementa ogni volta che il TPDU giunge ad un nodo. Se il contatore di tappe supera un certo valore, il pacchetto viene automaticamente eliminato.

Un altro metodo può essere quello di controllare l'età del pacchetto, inserendo nel TPDU un campo che indica da quanto tempo esso è stato creato. In tal caso tutti i pacchetti che raggiungono una certa età vengono eliminati. Questa tecnica prevede che tutti i nodi posseggano orologi sincronizzati fra loro.

Vi è poi il metodo detto di handshake a tre vie. L'entità di trasporto A, che vuole iniziare la comunicazione, associa ad essa una sequenza d'identificazione, per cui invia un messaggio di Connection Request all'entità di trasporto B, ponendo al suo interno l'indicatore di sequenza x prescelto.



Entità di trasporto A

CONNECT Request

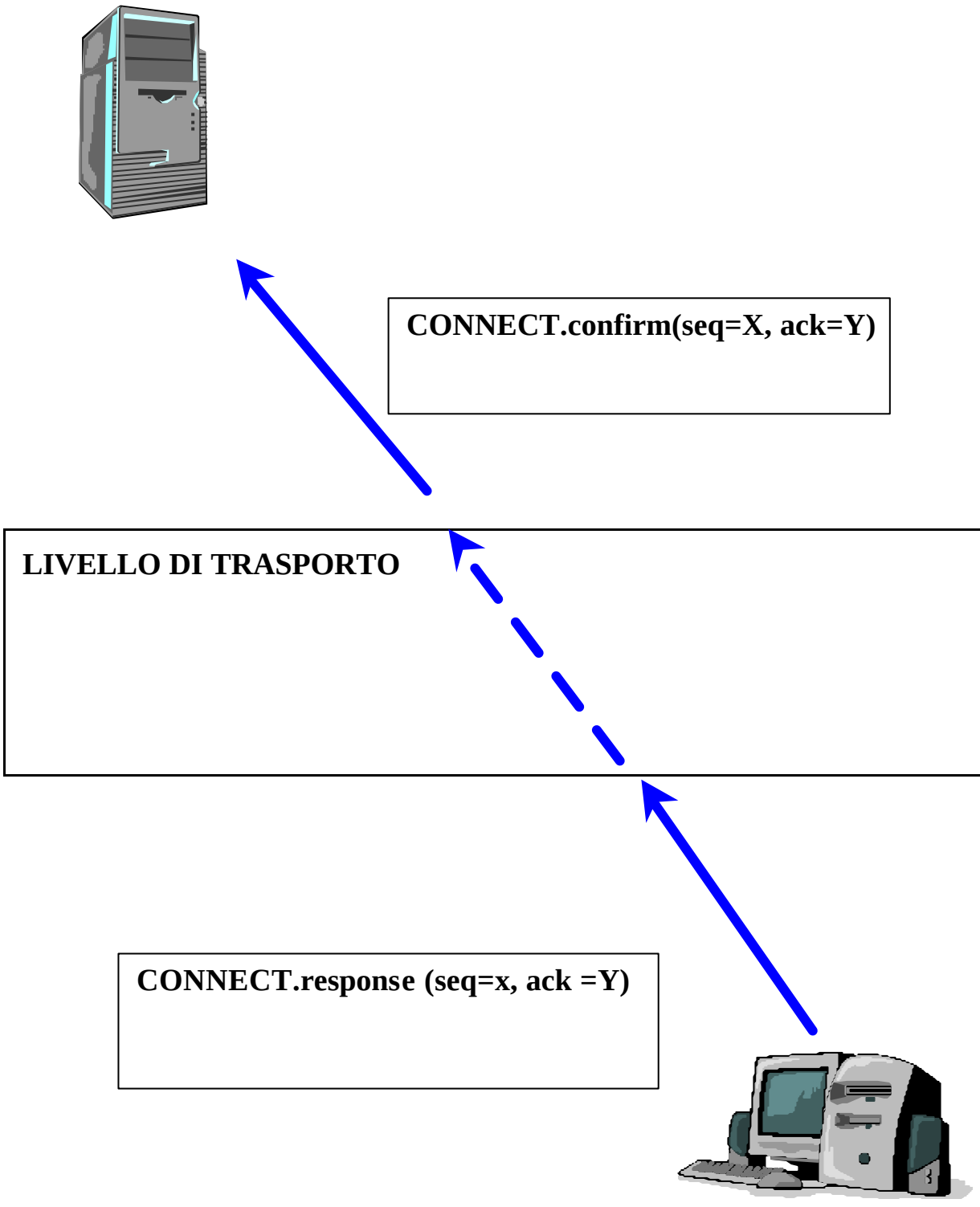
(seq =X)

LIVELLO DI TRASPORTO

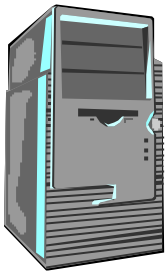
Connect Indication (seq =X)

Entità di trasporto B

L’entità di trasporto B invia in risposta un pacchetto in cui è contenuto l’indicatore di sequenza x, in modo che A sappia che è stato ricevuto ed un proprio indicatore di sequenza Y di riposta (acknowledge)



Da questo momento in poi, tutti i pacchetti che A invierà a B saranno contrassegnati dagli identificatori di sequenza x ed y

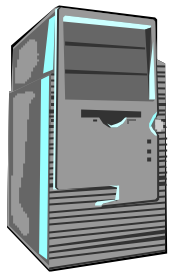


DATA(seq=X, ack=Y)

LIVELLO DI TRASPORTO

DATA(seq=X,ack=Y)

Supponiamo, ora che giunga all'entità di trasporto B un pacchetto di richiesta di connessione contrassegnato da un identificatore Z. Sia esso un pacchetto ritardatario appartenente ad una connessione già terminata

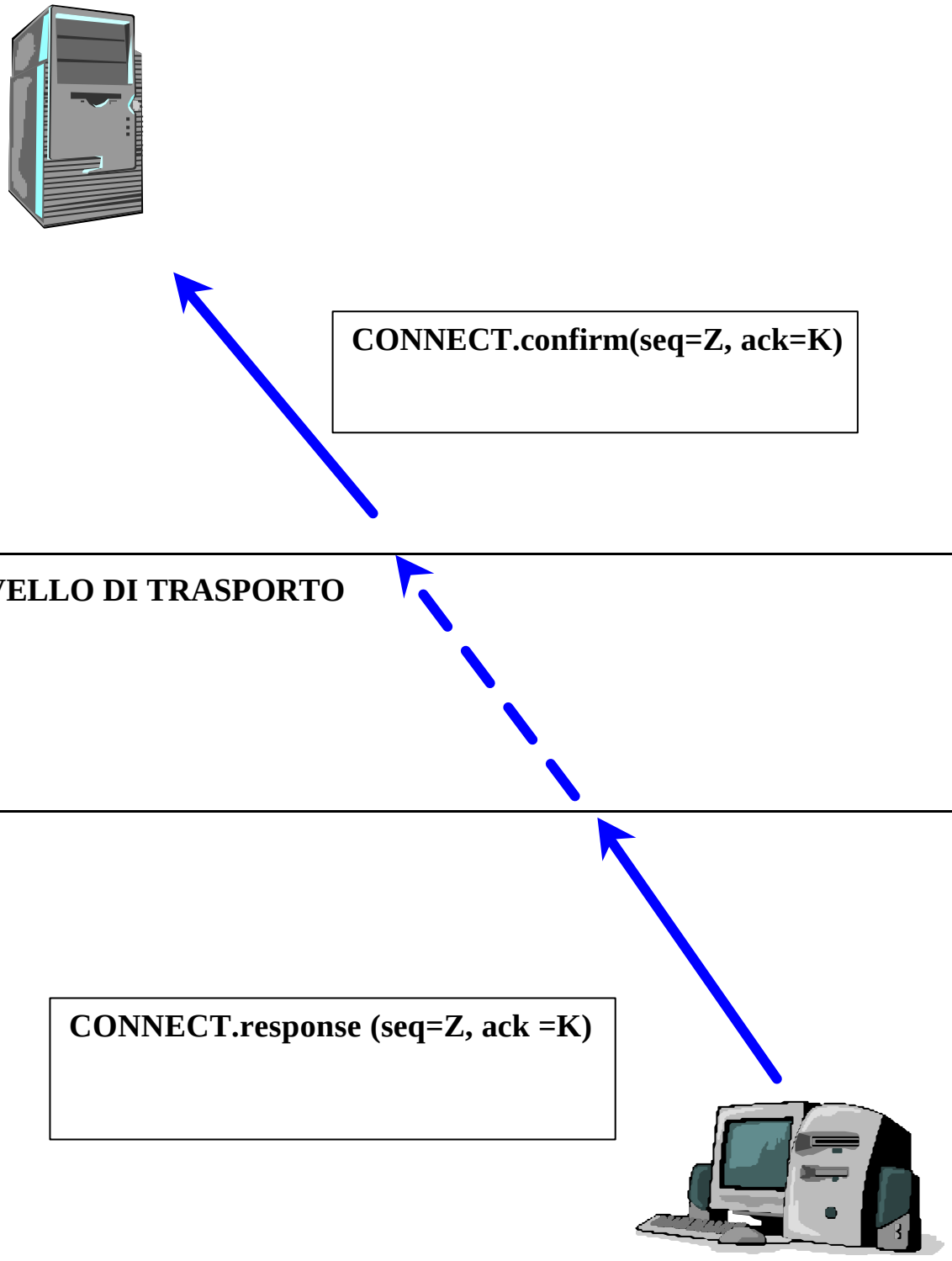


CONNECT.request (seq=Z)

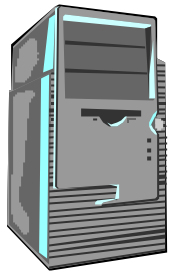
LIVELLO DI TRASPORTO

Connect.request(seq=Z)

L’entità di trasporto B non è in grado di sapere se si tratta di un pacchetto ritardatario, per cui risponde con un pacchetto di Connection Confirm in cui introduce l’identificatore Z che ha ricevuto e quello di risposta K



Ma l'entità di trasporto A è in grado di accorgersi che non ha in corso alcuna connessione identificata da Z e K e invia un messaggio di rifiuto o reject



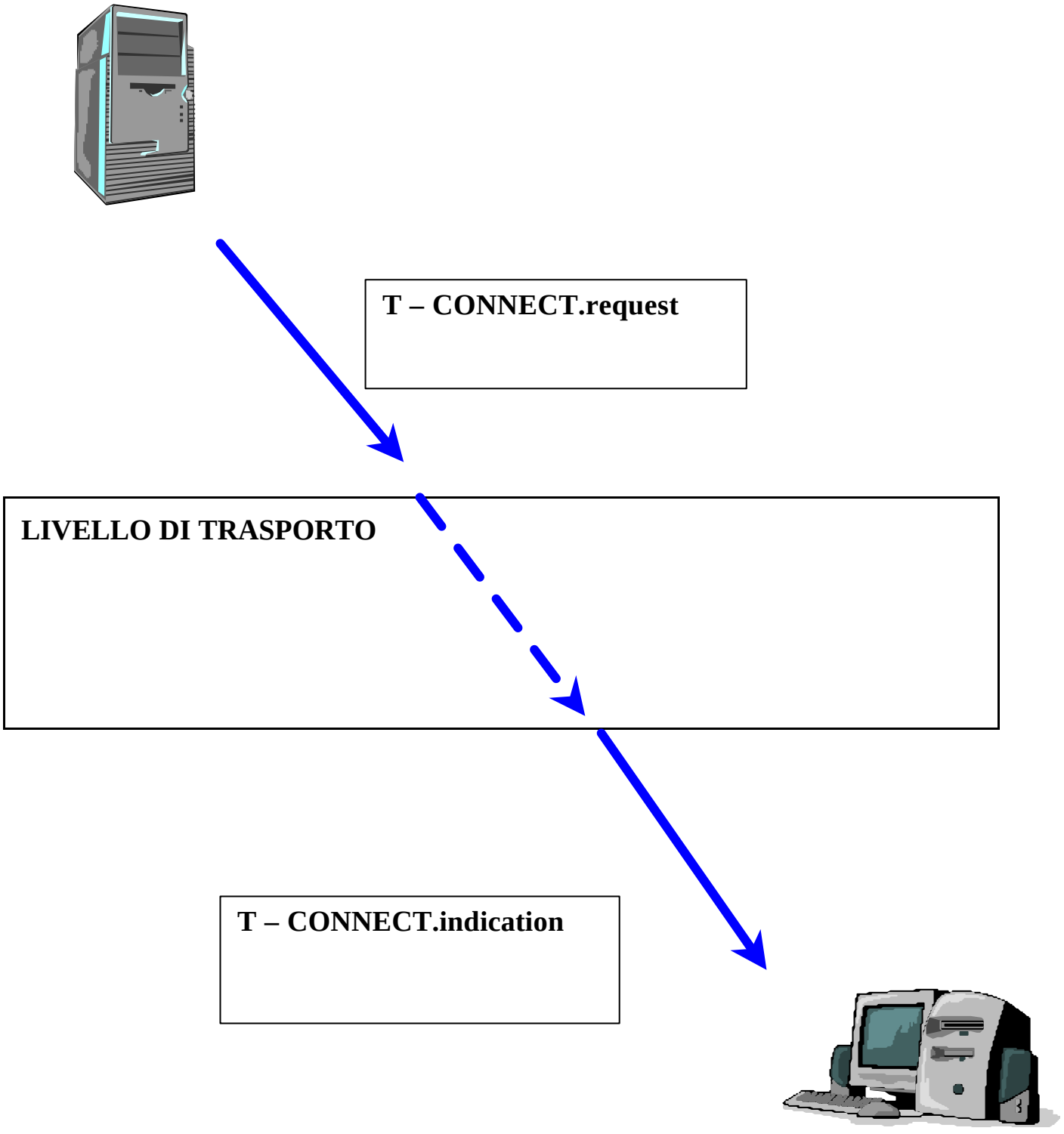
REJ(ack=K)

LIVELLO DI TRASPORTO

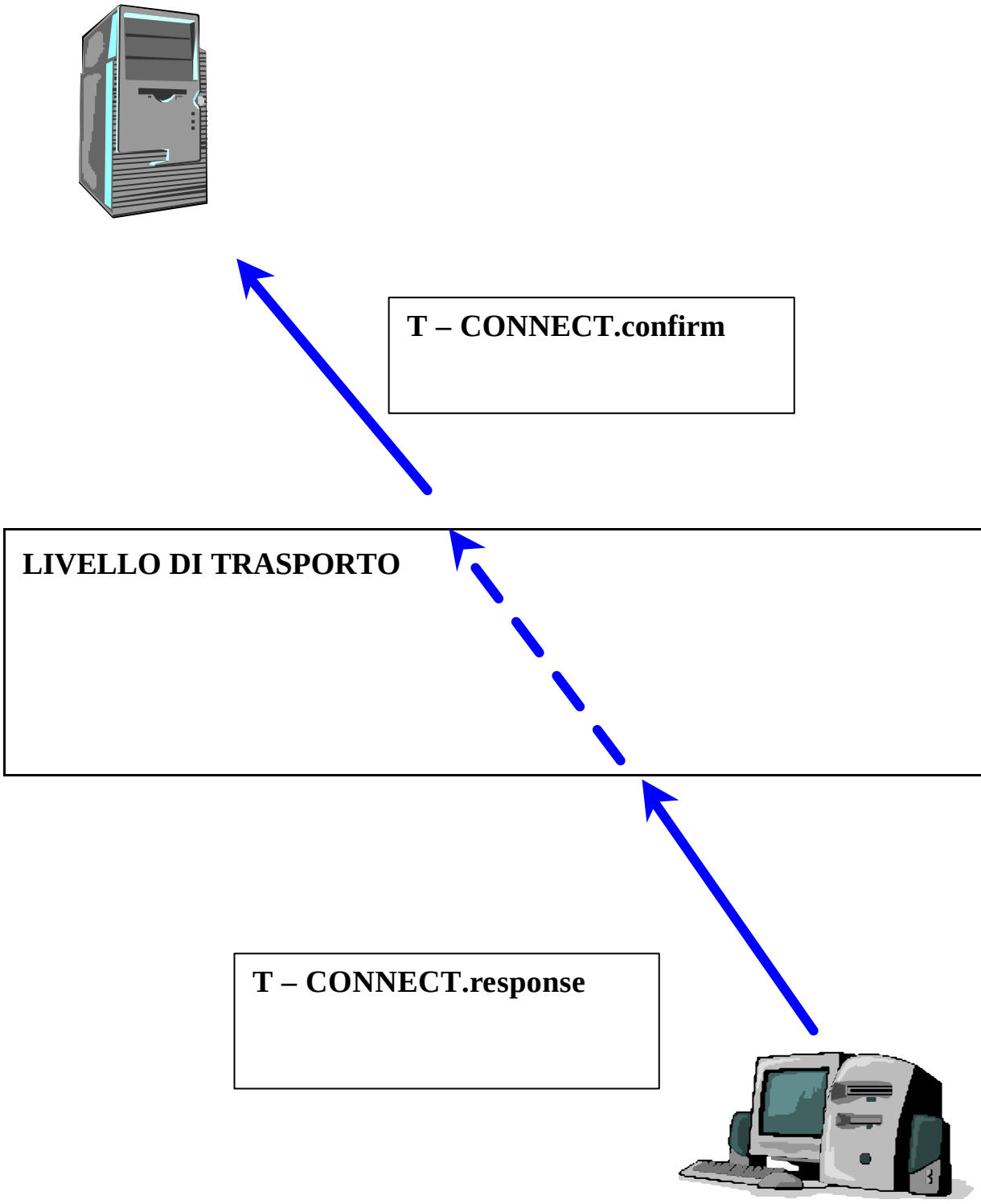
REJ(ack=K)

Rilascio di una connessione

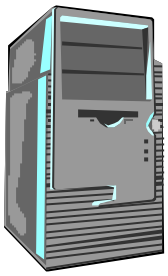
Anche il rilascio di una connessione è un problema non banale. Supponiamo che A invii una richiesta di connessione



B risponde positivamente a tale richiesta



A comincia a trasmettere un primo pacchetto

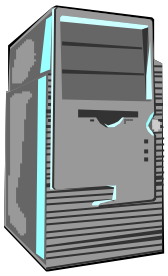


T – Data.request

LIVELLO DI TRASPORTO

T – DATA.indication

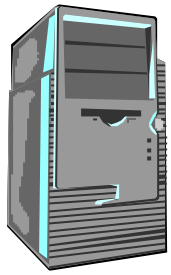
Ora A invia un altro pacchetto

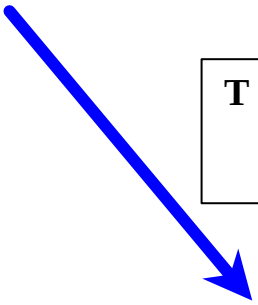


T – Data.request

LIVELLO DI TRASPORTO

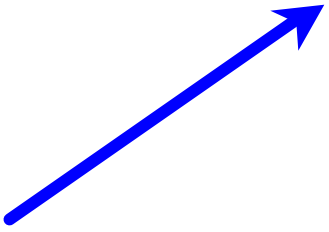
Ma, nel frattempo B ha richiesto di sconnettersi

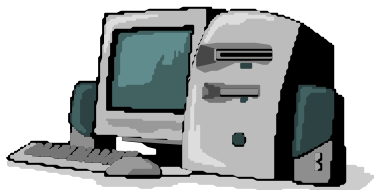




T – DATA.request

LIVELLO DI TRASPORTO





T – DISCONNECT.request

Il risultato è che viene rilasciata la connessione e il secondo pacchetto viene perso.

Si dovrebbe fare in modo che ognuna delle estremità eviti di inviare un T_DISCONNECT.request finché non sia sicura che l'altra abbia finito, come nel seguente dialogo fra esseri umani

Questa soluzione dall'apparenza efficiente non funziona sempre. Per rendere comprensibile il perché si fa spesso riferimento al cosiddetto problema dei due eserciti.

In una vallata si trova accampato l'esercito giallo, mentre sui monti che sovrastano la vallata si trova l'esercito nemico blu. L'esercito blu è diviso in due tronconi che possono comunicare con difficoltà fra di loro poiché sono separati dall'esercito giallo. L'esercito giallo è numericamente superiore ad ognuna delle due parti in cui è diviso l'esercito blu, ma quest'ultimo, se fosse unito, sarebbe superiore all'esercito giallo. Per questo motivo, l'unica speranza per l'esercito blu, di prevalere su quello giallo è che i due tronconi in cui esso è diviso attacchino contemporaneamente il nemico. Se ciò non avvenisse l'esercito giallo potrebbe facilmente vincere l'esercito blu sconfiggendone separatamente ognuna delle sue parti. I comandanti delle due armate dell'esercito blu devono allora mettersi d'accordo sul momento dell'attacco. Il comandante della prima armata invia un messaggio all'altro comandante, proponendo data ed ora dell'attacco e chiedendo un riscontro per sapere se è d'accordo. Se il messaggio di risposta si perde si potrebbe avere un disastro. Infatti, il primo comandante, non ricevendo la risposta, non sa se il secondo comandante è d'accordo e quindi non sa se esso attaccherà alla data e all'ora che era stata proposta. Probabilmente il primo comandante, nel dubbio, non attaccherà. Ma il secondo comandante non può sapere che il primo comandante non attaccherà, perché non ha ricevuto risposta, per cui potrebbe attaccare da solo alla data prefissata e subire una disastrosa sconfitta.

Si propone allora di migliorare il metodo e di istituire un protocollo di handshake a tre vie secondo il quale il primo comandante deve inviare al secondo comandante un messaggio di risposta al riscontro fatto dal secondo comandante.

Il sistema migliora? No, perché il primo comandante comincia a dubitare che il suo messaggio di risposta può non essere giunto al secondo comandante per cui quest'ultimo può dubitare che il suo riscontro non sia giunto al primo comandante, per cui la seconda armata potrebbe non attaccare all'ora prefissata.

Come si può capire il problema non è risolvibile. Per quanti messaggi di riscontro di sicurezza si può pensare di introdurre non avremo mai la sicurezza che il messaggio originario sia stato ricevuto.

Dato un protocollo con n messaggi si può ragionare in questo modo: l'ennesimo messaggio di riscontro è essenziale? Se la risposta è no, dobbiamo chiederci se il messaggio n-1 è essenziale: in ogni caso ci fermeremo ad un messaggio di ordine x che dovrà essere necessariamente essenziale (nella peggiore delle ipotesi il primo). Quando ci fermiamo al messaggio essenziale è evidente che, proprio perché esso è essenziale, una sua perdita avrebbe esiti disastrosi (nel caso dell'esercito blu esso prende una mazziata dall'esercito giallo).

Se agli eserciti sostituiamo le entità di trasporto e al problema dell'attacco sostituiamo il problema di mettersi d'accordo sul rilascio della connessione, scopriamo che la situazione è del tutto identica. Se l'entità di trasporto ad un estremo non è sicura che anche l'altra entità di trasporto all'altro estremo si disconnetterà, la connessione non verrà mai rilasciata. Una possibile soluzione è quella di avere un timer d'inattività su ogni entità di trasporto, in modo che, se passa un certo intervallo di tempo senza che siano giunti pacchetti la connessione viene rilasciata.