Programmazione di reti - Relazione Assigment 2

Anis Lico Simone Del Gatto

1 ottobre 2018

Indice

1 Descrizione Simulatore

Il simulatore si articola principalmente in due parti il lato sender rappresentato dall'entità A e il lato reciver rappresentato da B. Con *A_ init()* e *B_ init()*, chiamate subito all'inizio dell'esecuzione, vengono inizializzate variabili e strutture dati relative alle due entità. All'inizio del programma viene creata una struttura dati che indica quali eventi si devono verificare durante l'esecuzione e in che ordine. Tramite apposite funzioni, come ad esempio *tolayer3()* o *starttimer()*, si possono aggiungere eventi a questa lista. Il main si preoccuperà di simulare la la ricezione di un messaggio da parte di A proveniente dal layer 5 e di far si che A lo invii tramite la funzione *A_ output()*. A utilizzando la funzione *tolayer3()* si preoccuperà di inviare il messaggio a B. Dal punto di vista della simulazione questo significa che viene aggiunta alla lista degli eventi un evento che sarà del tipo "B riceve un pacchetto da A". Quando effettivamente questo evento si realizzerà il simulatore chiamerà la funzione *B_ input()* che nel caso il pacchetto ricevuto sia quello corretto spedirà un ack all'entità A tramite la funzione *tolayer3()* e spedirà il pacchetto al livello applicativo soprastante tramite la funzione *tolayer5()*. Quando si attuerà l'evento che determina la ricezione da parte di A di un ack inviatogli da B, se il pacchetto è quello previsto A si prepara a inviare nuovi pacchetti provenienti dal livello 5 (una spiegazione più dettagliata dell'implementazione delle routine è fornità successivamente). Il simulatore mette a disposizione anche un timer che viene controllato tramite le due funzioni *starttimer()* e *stoptimer()*. A utilizzare questo timer sarà unicamente A che lo avvierà al momento dell'invio di un pacchetto e lo farà terminare al momento della corretta ricezione di un ack (la gestione del timer sarà rivista poi in maniera più approfondita successivamente). Allo scadere del timer, dovuta alla non corretta ricezione dell'ack o perché quest'ultimo non è proprio

arrivato o per un fenomeno di corruzione dei pacchetti, il simulatore chiama la funzione *A_timerinterrupt()* che permette il riinvio del pacchetto attualmente non ackato.

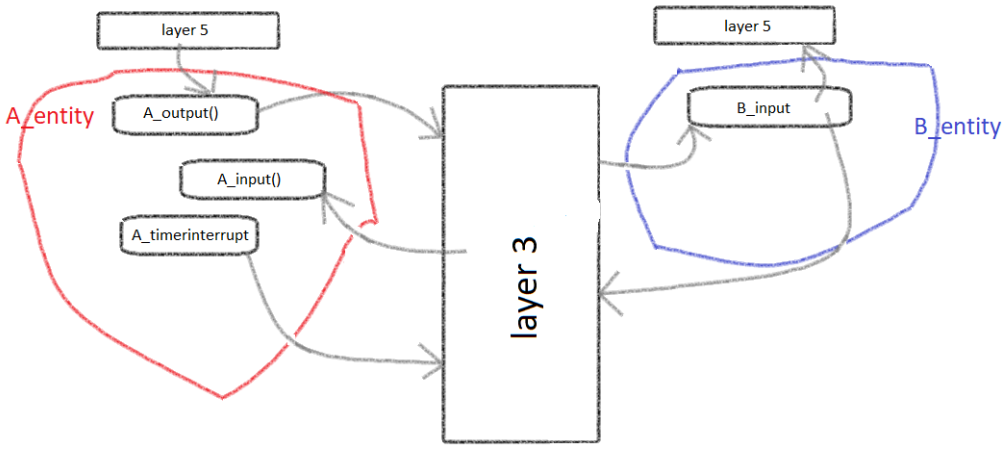

Figura 1: Schema simulatore

2 Strategie Risolutive

2.1 Strutture Dati

- **messToSend**: è una struttura che di permette di realizzare una coda per bufferizzare i messaggi che devono essere inviati da A a B.
- **messagesList**: puntatore alla testa della coda che indica il prossimo messaggio da inviare. Questo è necessario perché il simulatore potrebbe far si che il livello 5 cerchi di far inviare un messaggio tramite A mentre questo è in fase di attesa di un ack da B. Non potendolo inviare in quel momento lo può bufferizzare per poi inviarlo successivamente.
- **currentPacket**: puntatore al pacchetto in inviato da A che deve essere ackato. Questa struttura dati si rende necessaria per avere un riferimento attraverso la quale si può ritrasmettere eventualmente il pacchetto.

2.2 Descrizione delle Routine

- **A_init**: inizializza le strutture e le variabili relative ad A.

- **B_init**: inizializza le strutture e le variabili relative ad B .
- **A_output(message)**: si controlla la variabile *state* che può avere due valori: *CAN_SEND_MESSAGE* che indica che si può procedere con l'invio di un messaggio e *WAITING_ACK* che indica che si è in attesa di un acknolegment da B. E' chiaro che se B ha come valore *CAN_SEND_MESSAGE* si può procedere con l'invio e l'avvio del timer tramite *starttimer()*, in caso contrario il messaggio sarà bufferizzato in *messagesList*.
- **A_input(packet)**: si possono verificare due situazioni
  1. il pacchetto ricevuto non è corrotto e ha il numero di acknolegment uguale a quello atteso da A: viene stoppato il timer e si guarda se nella lista dei messaggi ci sono dei messaggi bufferizzati. In caso affermativo viene inviato il primo della coda e la variabile state viene settata a *WAITING_ACK*. In caso contrario a si prepara a inviare nuovi messaggi provenienti dal livello 5 settando la variabile state a *CAN_SEND_MESSAGE*
  2. il pacchetto è corrotto o ha un numero di ack non atteso: viene riinviato il messaggio puntato da *currentPacket*.
- **A_timerinterrupt()**: viene ritrasmesso il pacchetto puntato da *currentPacket*
- **B_input(packet)**: possono verificarsi due casi:
  1. il pacchetto ricevuto è corretto e il suo sequence number è quello atteso da B. Viene inviato un ack con acknum uguale a seqnum del pacchetto e il pacchetto viene consegnato al livello 5 tramite la funzione *tolayer5()*.
  2. il pacchetto ricevuto o è corrotto o ha un sequence number non atteso (magari frutto di una ritrasmissione). Viene inviato un ack con acknum uguale all'opposto numero di sequenza che B si aspettava di trovare nel pacchetto ricevuto da A

2.3 Diagrammi a stati finiti

Macchine a stati finiti per la descrizione del comportamento dell'entità A e dell'entità B

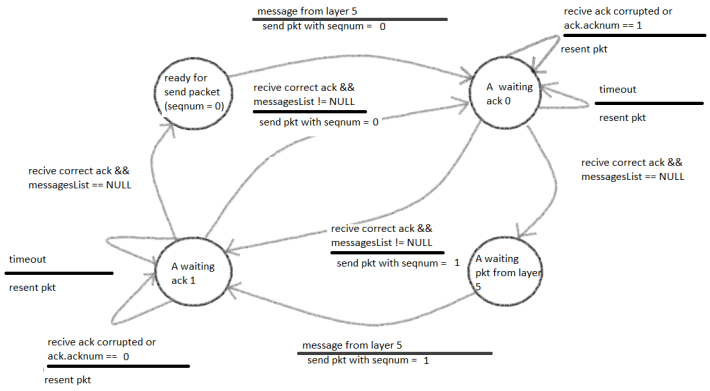

Figura 2: Diagramma a stati entità A

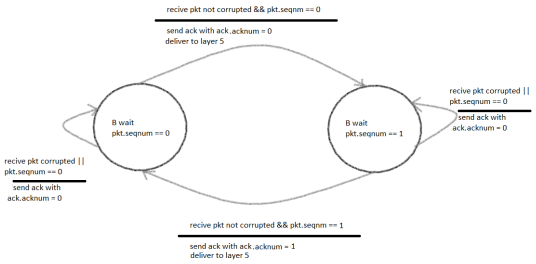

Figura 3: Diagramma a stati entità B

3 Test e Relativo output

L'output dei due test richiesti si trovano nella cartella della relazione rispettivamente con i nomi *outputCase1Assignment2* e *outputCase2Assignment2*. Su in ambiente Windows è possibile visualizzarli con impaginazione corretta con *Notepad*. Con Linux in particolare Ubuntu si visualizzano correttamente utilizzando il blocco note. In generale si possono aprire e visualizzare correttamente con *Atom*.

j d j d j j d j d
d l d l d l v m ,
s m l s v m