Simulazione di schemi di accesso a divisione di tempo su canale comune via satellite.

Algoritmi di SATNET.

Renzo Beltrame

Rapporto interno C80-10

Consiglio Nazionale delle Ricerche

CNUCE

PISA

Simulazione di schemi di accesso a divisione di tempo su canale comune via satellite.

Algoritmi di SATNET.

Renzo Beltrame

Rapporto interno C80-10

Istituto CNUCE
PISA
Dicembre 1980

1a Versione

RIFERIMENTO : 80.P1.COMPUNET.15
COMPUNET/CNUCE/80/006

ORIGINE : R. Beltrame - CNUCE

DISTRIBUZIONE : Componenti GL1

NATURA : Rapporto

DATA : Dicembre 1980

VERSIONI PRECEDENTI :

SOMMARI

RIASSUNTO

Il rapporto descrive un programma che simula i protocolli basici di accesso ad un canale comune via satellite con tecniche TDMA (F-TDMA, S-ALOHA, R-TDMA/F-TDMA, R-TDMA/S-ALOHA). Il programma, scritto in linguaggio SIMULA 67, e' fortemente parametrico e modulare, cosi' da consentire di variare facilmente il numero degli accessi, la struttura della trama, la velocita' del canale, i generatori di traffico, le funzioni di rumore, etc.. Oltre alle usuali misure di media e varianza e' possibile memorizzare a intervalli di tempo regolari il valore delle variabili stocastiche prescelte per elaborazioni piu' sofisticate, ad esempio stima degli intervalli di confidenza o analisi delle serie temporali.

ABSTRACT

The report describes a program which simulates basic Time Division Multiple Access protocols to a satellite common channel (F-TDMA, S-ALOHA, R-TDMA/F-TDMA, R-TDMA/S-ALOHA). The program is written in SIMULA 67, it is highly parametric and modular in structure, so that the number of access points, frame structure, traffic generators, noise functions, etc., my be easily changed. Program records in every run expectaction and variance of the stochastic variables which are involved. It is possible to record optionally the values of selected stochastic variables taken at predefined time intervals. More sofisticated statistical computations are thus allowed, as confidence intervals estimation or time series analysis.

INDICE

1.0 INTRODUZIONE

Il modello concettuale che sottosta' al programma di simulazione considera un mezzo trasmissivo che consente l'accesso multiplo ed ha inerente la diffusione di tipo broadcasting. In tale mezzo qualsiasi numero finito di utenti puo' trasmettere informazioni su uno stesso canale, accesso multiplo, e le informazioni trasmesse possono venir ricevute su un'ampia area da qualsiasi numero di utenti, broadcasting. Un mezzo trasmissivo con tali proprieta' verra' indicato nel seguito come canale comune o, piu' semplicemente, canale.

Nel caso di un canale via satellite, considerato nel modello simulato, si suppone, come accade nella paratica attuale, che un qualsiasi numero prefissato di stazioni di terra, in seguito designate semplicemente stazioni, possa trasmettere segnali al satellite su una portante (il canale ad accesso multiplo); e che i segnali ricevuti dal trasponditore a bordo del satellite siano rinviati a terra in trasparenza su un'altra portante (il canale di tipo broadcasting). A bordo del satellite non e' pertanto prevista alcuna elaborazione.

Il segnale rinviato a terra dal trasponditore di bordo puo' venir ricevuto da tutte le stazioni situate nell'area di copertura del fascio del trasponditore, cosicche' un canale via satellite consistente di due portanti fornisce una topologia di rete completamente connessa per tutte le stazioni di terra situate nell'area di copertura del fascio del trasponditore.

Dal momento che un sistema di comunicazioni radio via satellite geograficamente distribuito comporta ritardi di propagazione, i quali, a seconda della distribuzione geografica delle stazioni di terra possono differire anche notevolmente, si definisce un asse dei tempi di riferimento detto tempo di canale. Quale tempo di canale viene scelto il tempo che sarebbe scandito da un orologio situato nel trasponditore a bordo del satellite. Tale tempo di riferimento puo' ovviamente essere virtuale, nel caso ad esempio di un trasponditore puramente passivo, con sola funzione, cioe', di ripetitore, e in tal caso sono previste, nelle stazioni di terra, opportune funzioni che permettono di calcolare tale tempo virtuale e di sincronizzare su di esso le varie operazioni.

Nel programma di simulazione questo aspetto delle reti via satellite non e' stato introdotto come algoritmo esplicito; le stazioni sono supposte sempre tutte sincronizzate rispetto al tempo di canale.

Questo lavoro e' stato svolto nell'ambito del Progetto Finalizzato Informatica - Obiettivo COMPUNET.

Si suppone anche un canale con ampiezza di banda limitata, come accade sempre nella realta', e tale parametro compare nel programma di simulazione sotto forma di velocita' del canale in bit/sec.

Ogniqualvolta sul ricevitore di una stazione vi sia una sovrapposizione di tempo fra due segnali sulla medesima portante, si considera che nulla sia ricevuto correttamente. Un simile evento verra' chiamato collisione sul canale o, semplicemente, collisione.

Il tempo di canale e' slottizzato diviso cioe' in intervalli di tempo secondo una periodicita' prestabilita, trama, e ciascun utente deve sincronizzare l'inizio della trasmissione e la sua durata temporale con la slottizzazione del tempo di canale.

Tale funzione, nel programma di simulazione, e' svolta tramite un algoritmo distribuito: ogni stazione di terra, cioe', scandisce la trama e decide la trasmissione sulla base dell'algoritmo con cui viene schedulato l'accesso al canale.

Nel programma di simulazione oltre ad avere un cnale slottizzato si hanno stazioni che trasmettono un solo pacchetto per slot. Inoltre, benche' certi protocolli di accesso al canale permettano di usare slot di lunghezza temporale diversa, tale possibilita' e' stata evitata.

Tutto cio' permette di usare quale unita' di misura per il traffico, il throughput e i valori imposti ai generatori di traffico, il numero di pacchetti per slot, avendo cosi' il vantaggio di avere numeri largamente indipendenti dalla velocita' del canale.

La dipendenza si ha, ovviamente, quando si considerano gli effetti della "bit error rate" (BER), ma si tratta di una influenza che si manifesta attraverso la lunghezza del pacchetto nel nostro caso supposta costante per tutti i generatori di traffico.

La lunghezza del pacchetto come parametro da fornire al programma separatamente dalla lunghezza temporale della slot consente di tener conto in modo globale dei tempi di guardia, dei preamboli di acquisizione, e di ogni altro elemento strettamente dipendente dall'hardware impiegato.

E' possibile collegare ad ogni stazione piu' di un generatore di traffico con caratterisitche diverse; questi tuttavia sono trattati in modo eguale dalle stazioni.

E' stata inoltre simulata tramite un opportuno processo una acquisizione di traffico da ogni singola stazione con velocita' e distribuzione che puo' venir variata in ogni sessione di simulazione.

Infine, data la natura del programma, non sono previste procedure per la riconfigurazione dinamica della rete durante una stessa sessione di simulazione.

2.0 ALGORITMI DI ACCESSO AL CANALE

Il programma di simulazione prevede attualmente quattro diversi algoritmi di schedulazione dell'accesso al canale (Channel Access Scheme): un Fixed Time Division Multiple Access (F-TDMA), uno Slotted ALOHA (S-ALOHA), un Reservation-TDMA (R-TDMA), e un R-TDMA/S-ALOHA. I primi tre CAS sono simulati nella forma implementata sulla rete SATNET, la parte satellitaria della rete ARPANET.

In tutti i protocolli simulati abbiamo una trama (frame) composta da N slot dedicate a pacchetti di controllo, dove N e' il numero massimo di accessi che possono essere attivi durante la sessione di simulazione, ed M slot dedicate ai dati (vedi Fig. 1). Le slot di controllo e quelle riservate ai dati hanno di solito lunghezza differente; la loro lunghezza e' uno dei parametri imposti ad ogni sessione di simulazione.

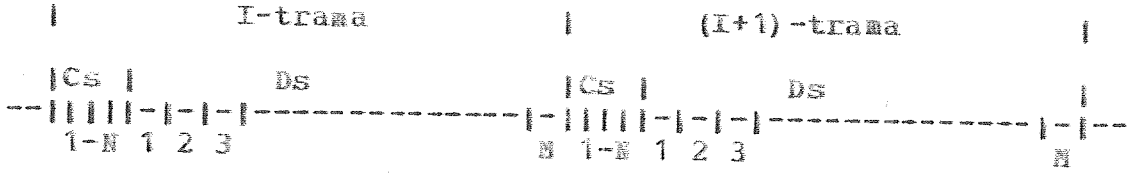


Fig. 1 - Struttura della trama

Ad ogni accesso al canale e' assegnata una slot di controllo e il pacchetto inviato in tale slot puo' assolvere a due funzioni:

fornire un riferimento periodico che puo' venir usato per individuare il venir meno di qualcuno degli accessi attivi;

garantire che la conferma dei dati ricevuti possa venir inviata da ogni stazione con un ritardo massimo prestabilito, indipendentemente dal traffico tra le stazioni.

2.1 FIXED TDMA (F-TDMA)

Questo protocollo ripartisce egualmente

fra tutte le stazioni (vedi Fig. 2) la banda (tempo) offerta dal canale.

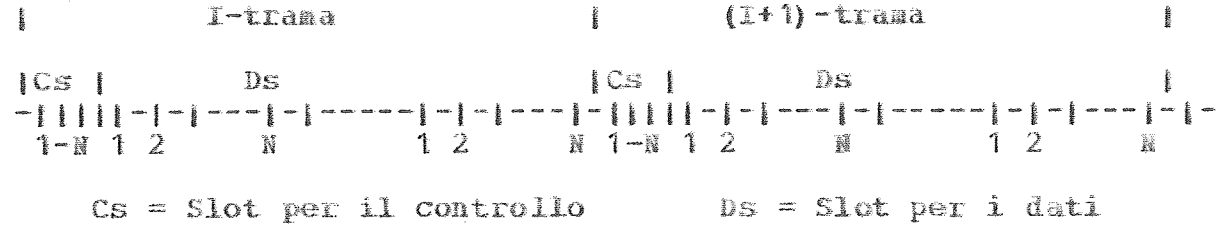


Fig. 2 - Trama F-TDMA

L'algoritmo F-TDMA assegna alle stazioni le slot immediatamente successive l'ultima slot di controllo da 1 ad N nell'ordine, una per stazione ad ogni trama. Per quanto non richiesto dal programma di simulazione e' nella logica di questo CAS avere un numero di slot di dati per ogni trama che sia un multiplo intero delle stazioni attive.

Le slot di controllo in questo protocollo possono anche mancare e in tal caso si ha una assegnazione ciclica delle slot per dati fra le varie stazioni. Tale possibilita' e' realizzata programma di simulazione dando 0 peril numero delle slot di controllo in ogni trama, un parametro che deve essere passato al programma ad ogni run.

2.2 SLOTTED ALOHA (S-ALOHA)

In questo protocollo il canale e' slottizzato o come mostrato in Figura 1, oppure senza slot di controllo, ma le slot riservate ai dati non sono preassegnate alle stazioni. Ogni stazione trasmette entro ogni singola slot di dati con una certa probabilita', posto che abbia in coda pacchetti da inviare. Poiche' le stazioni si contendono fra loro le slot questo protocollo e' classificato anche del tipo a contesa (contention) per contrapporlo a quelli in cui le slot sono preassegnate, vuoi in modo fisso che a domanda.

Nel protocollo S-ALOHA se due o piu' stazioni trasmettono nella stessa slot si avra' una collisione sul canale, e l'informazione portata dai pacchetti che vi sono coinvolti e' a tutti gli effetti perduta. Se viene implementato un algoritmo capace di individuare tale perdita di pacchetti, i

pacchetti collisi verranno accodati dalla stazione trasmittente per la ritrasmissione. Tale algoritmo e' presente nel programma di simulazione, dove ogni stazione ha anche la coda dei pacchetti in attesa di ritrasmissione.

L'algoritmo impiegato da ciascuna stazione per schedulare le proprie trasmissioni e' riportato qui di seguito:

```
procedure QUESCHEDULING(STAT);
   ref(STATION) STAT;

   if not STAT.RETRANSQUEUE.Empty then
   begin
      if (Uniform(0,1,STAT.U1) le STAT.RGATE) then
         <transmit a packet from retransmission queue>;
   end
   else
      if not STAT.NEWQUEUE.Empty then
      begin
         if (Uniform(0,1,STAT.U2) le STAT.NGATE) then
         <transmit a packet from new arrival queue>;
      end;
```

L'algoritmo implementato fa uso di due soglie di probabilita' (gate) per decidere se trasmettere, un R-gate per la ritrasmissione ed un N-gate per la coda dei nuovi arrivi. I pacchetti in coda per la ritrasmissione hanno priorita' sui pacchetti nuovi arrivati.

Se la coda dei pacchetti da ritrasmettere non e' vuota si genera un numero casuale secondo una distribuzione uniforme tra 0 ed 1. Se tale numero e' minore del valore di R-gate si estrae un pacchetto dalla coda e lo si invia in trasmissione, se invece e' maggiore si ripete l'operazione alla slot successiva.

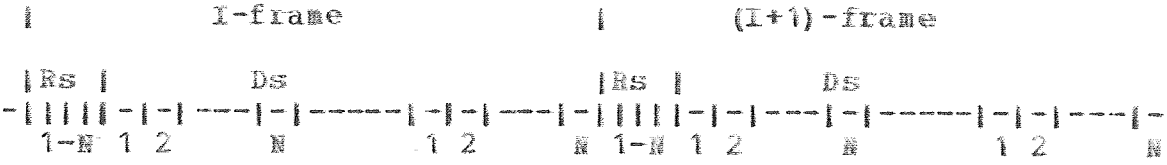
Se non si hanno pacchetti da ritrasmettere, e la coda dei nuovi arrivi non e' vuota si estrae un numero casuale secondo una distribuzione uniforme tra 0 e 1. Se tale numero e' minore di N-gate viene inviato in trasmissione il primo dei pacchetti in coda sulla coda dei nuovi arrivi. In tutti gli altri casi la stazione attende la prossima slot.

I valori di R-gate ed N-gate sono imposti ad ogni stazione all'inizio di ciascun run e rimangono costanti per tutto il tempo di simulazione. In questo programma, infatti, non e' stato implementato alcun algoritmo di controllo dello S-ALOHA, per cui sono possibili situazioni di instabilita' del canale dovute al protocollo stesso, che, come e' noto, presenta una vasta zona di instabilita'.

2.3 RESERVATION TDMA (R-TDMA)

Questo protocollo consente una allocazione dinamica della banda offerta dal canale impiegando un algoritmo distribuito e pertanto implica uno scambio di informazioni di controllo tra le stazioni. La struttura della trama e' descritta nella Figura 3; in ciascuna trama abbiamo:

- una parte riservata alle prenotazioni in cui ogni stazione fa' conoscere a tutte le altre la propria richiesta di banda,
- una parte dati in cui un algoritmo, identico in ogni stazione, schedula l'accesso al canale.



Rs = Slot per la prenotazione Ds = Slot per i dati

Fig. 3 - Trama R-TDMA

L'algoritmo di schedulazione si basa sull'uso della "Central Queue Table" (CQT), di cui si suppone ogni stazione abbia in ogni istante una copia aggiornata in modo identico. Ciascuna voce della tabella rappresenta il numero di prenotazioni ancora attive per la stazione corrispondente, secondo una successione delle stazioni prefissata e costante nel tempo.

Le slot di dati sono assegnate in prima istanza su base F-TDMA. Se la stazione che ha il "possesso" F-TDMA della slot corrente ha prenotazioni da evadere, trasmette un pacchetto in quella slot e ciascuna stazione diminuisce di 1 le prenotazioni relative a quella stazione nella propria copia della CQT. Se invece la stazione ha gia' evaso tutte le proprie prenotazioni, la slot viene assegnata sulla base di un algoritmo round-robin ad una delle stazioni che ancora prenotazioni inevase.

L'algoritmo impiegato e' il seguente:

```
!**** send scheduling**********;
 short integer procedure SENDSCHED(STAT,TB);
```

```
    ref(STATION) STAT;
    long real TM;

 begin
    integer I,I1;

    I1:=Mod(STAT.SENDPSN,STATNUM);
    if STAT.SYNCFLAG and RTDMAFLAG then
    begin
       if (STAT.TOTRES gt 0) then
       begin
         !*** cqtab scheduling ***;
          if (STAT.CQTAB(I1) eq 0) then
          begin
            !*** dynamic assignment ***;
             I:=Mod(STAT.CQPTR+1,STATNUM);
             while (STAT.CQTAB(I) eq 0) do
               !*** round robin ***;
                I:=Mod(I+1,STATNUM);
             STAT.CQPTR:=I;
             I1:=I;
          end;
          STAT.TOTRES:=STAT.TOTRES-1;
          if (STAT.CQTAB(I1) gt 0) then
             STAT.CQTAB(I1):=STAT.CQTAB(I1)-1;
       end** OF CQTAB SCHEDULING **;
    end;
    SENDSCHED:=I1;
 end*** OF PROCEDURE SENDSCHED ***;
```

Si realizza cosi' in uno dei modi piu' semplici una assegnazione dinamica della banda totale disponibile.

Nel programma di simulazione i pacchetti di prenotazione hanno il seguente formato:

```
 MESSAGE class RESVSLOT;
 begin
       integer OLDRES,PTR,NEWRES;

 end****of class resvslot****;
```

Le informazioni sono le stesse previste per l'implementazione su SATNET, solo il formato tradotto in bit e' diverso. Come si vede essi contengono sia le nuove prenotazioni, sia le informazioni necessarie per mantenere identiche le copie della CQT in tutte le stazioni.

La variabile OLDRES contiene il numero di prenotazioni effettuate nella trama precedente; la variabile NEWRES contiene invece il numero di nuove prenotazioni che la stazione intende effettuare, mentre la variabile PTR contiene la posizione corrente del puntatore mobile usato per ridistribuire le slot libere sulla base dell'algoritmo round-robin.

L'ammontare delle nuove prenotazioni e' uguale al numero totale di pacchetti in attesa di trasmissione su quella stazione meno il numero di prenotazioni rimaste inevase nella trama corrente, numero che corrisponde al valore attuale delle prenotazioni nella CQT.

Ogni stazione invia un pacchetto di prenotazione con tutte le informazioni richieste nella parte riservata alle prenotazioni di ogni trama seguendo un protocollo F-TDMA, e cio' anche se l'ammontare delle nuove prenotazioni e' 0. Il pacchetto di prenotazione serve infatti alle stazionianche per effettuare un controllo indiretto che tutti posseggano una copia ugualmente aggiornata della CQT. Il corretto funzionamento di questo CAS si fonda infatti su tale presupposto.

Allorquando un pacchetto di prenotazione e' ricevuto da una stazione le informazioni ricevute vengono memorizzate nella posizione della tabella delle prenotazioni (RES-table) corrispondente alla stazione mittente ed inoltre viene aumentato di uno il valore di una variabile dove viene memorizzato il numero dei pacchetti di prenotazione ricevuti ("heard"-counter).

Trascorso un tempo sufficiente perche' tutte le stazioni possano aver ricevuto tutti i pacchetti di prenotazioni (si deve cioe' tener conto delle differenze nei ritardi di propagazione relativi alle varie stazioni), le informazioni contenute nella RES-table sono impiegate per aggiornare la CQT. L'aggiornamento della CQT e' effettuato da tutte le stazioni nella medesima slot: la "update-slot".

Ad ogni update-slot, viene fatto un test sull'"heard"-counter per vedere se i pacchetti di prenotazione delle stazioni attive sono stati tutti correttamente ricevuti, se il test e' positivo viene eseguita la procedura seguente:

```
!*** updates reservation table *********;
 procedure RSVUPDATE(STAT);
    ref(STATION) STAT;
 begin
    short integer I;

    if (STAT.HEARD eq STAT.STATUP) then
    begin
      !*** cqtab update **********;
       STAT.TOTRES:=0;
       for I:=0 step 1 until NSTAT do
       begin
          STAT.CQTAB(I):=STAT.CQTAB(I)+STAT.RESVTAB(I,3);
          STAT.TOTRES:=STAT.TOTRES+STAT.CQTAB(I);
       end;
    end
    else
      !*** station out of sync ***;
       STAT.SYNCFLAG:=false;
```

```
    !*** reset transient variables ****;
     STAT.HEARD:=0;
     for I:=0 step 1 until NSTAT do
     begin
        STAT.RESVTAB(I,1):=0;
        STAT.RESVTAB(I,2):=0;
        STAT.RESVTAB(I,3):=0;
     end;
  end*** OF PROCEDURE RSVUPDATE ***;
```

Quando invece una stazione non ha ricevuto tutti i pacchetti di prenotazione si considera "fuori sincronismo" e la propria variabile SYNCFLAG assume il valore: "false". La stazione continua ad usare le slot TDMA che le sono preassegnate (come in un protocollo F-TDMA) ed inoltre invia pacchetti di prenotazione, ma non si avvale dell'assegnazione dinamica.

Quando una stazione che e' fuori sincronismo torna a ricevere tutti i pacchetti di prenotazione, deve conoscere, per risincronizzarsi, l'ammontare delle prenotazioni ancora inevase al momento in cui viene spedito il pacchetto di prenotazione, informazione che, proprio per questo motivo, e' inserita da ogni stazione nel proprio pacchetto di prenotazione.
Per tornare in sincronismo deve eseguire allora una schedulazione "dummy" delle slot dati intercorse tra la spedizione delle prenotazioni e l'aggiornamento della CQT. Tale schedulazione "dummy" e' fatta, ovviamente, nell'ipotesi che anche la stazione che sta risincronizzandosi usufruisca di assegnazione dinamica, mettendosi cioe' nella situazione in cui e' vista dalle altre stazioni. Per far questo oltre ad usufruire delle informazioni in suo possesso sulle prenotazioni inevase, si avvale della posizione del puntatore per l'assegnazione dinamica di una stazione prefissata che ha funzione di "leader".
La procedura di risincronizzazione e' la seguente:

```
!*** reset synchronism of scheduling *********;
 procedure SYNCHRESET(STAT,LEADER);
    ref(STATION) STAT;
    short integer LEADER;
 begin
    short integer I;

    procedure DUMMYSEND;
    begin
       short integer I,I1;

       for I1:=0 step 1 until (UPDSLOT-1) do
       begin
          if (STAT.TOTRES gt 0) then
          begin
             I:=Mod(I1,STATNUM);
             if (STAT.CQTAB(I) le 0) then
             begin
```

```
                    !*** dummy dynamic assignment ***;
                     I:=Mod(STAT.CQPTR+1,STATNUM);
                     while (STAT.CQTAB(I) eq 0) do
                        I:=Mod(I+1,STATNUM);
                     STAT.CQPTR:=I;
                  end;
                  STAT.TOTRES:=STAT.TOTRES-1;
                  if (STAT.CQTAB(I) gt 0) then
                     STAT.CQTAB(I):=STAT.CQTAB(I)-1;
               end
               else
                  goto EXIT;
            end;
         EXIT:
         end*** OF PROCEDURE DUMMYSEND ***;

      !*** synchreset procedure body ***************;
         STAT.CQPTR:=STAT.RESVTAB(LEADER,2);
         STAT.TOTRES:=0;
         for I:=0 step 1 until (NSTAT) do
         begin
            STAT.CQTAB(I):=STAT.RESVTAB(I,1);
            STAT.TOTRES:=STAT.TOTRES+STAT.CQTAB(I);
         end;
         DUMMYSEND(STAT);
         STAT.SYNCFLAG:=true;
   end*** OF PROCEDURE SYNCHRESET ***;
```

La perdita e il riacquisto della sincronizzazione sono in questo protocollo eventi abbastanza cruciali e pertanto nel programma di simulazione sono segnalati mediante un opportuno Io sul file di output.

3.0 PROTOCOLLO DI CONFERMA DELLA RICEZIONE

Questo protocollo regola l'invio da parte della stazione destinataria, o delle stazioni destinatarie nel caso di destinazione multipla, di una conferma alla stazione mittente di ogni pacchetto di dati ricevuto correttamente.

Il protocollo di conferma si avvale di un sistema di numerazione delle slot per identificare ogni pacchetto da confermare e la numerazione delle slot per questo scopo e' fatta da ogni stazione contestualmente al meccanismo di sincronizzazione e di schedulazione della trasmissione dei pacchetti di dati. Il modulo di tale numerazione e' scelto tenendo conto dell'intervallo di tempo massimo che si vuol lasciar trascorrere tra la trasmissione di un pacchetto e il ritorno della conferma della sua corretta ricezione.

Un limite inferiore di questo tempo e' dato dalla somma del ritardo di propagazione del canale, del tempo di acquisizione ed elaborazione delle stazioni e del tempo di trasmissione di un pacchetto dati e un pacchetto con conferme (ACK). Il limite superiore, che stabilisce il modulo di numerazione delle slot, cioe' la finestra di ACK e di conseguenza la supertrama per l'ACK, e' determinato dal limite inferiore di cui si e' detto in precedenza piu' il massimo intervallo di tempo che puo' intercorrere tra due opportunita' per la stazione ricevente di inviare la conferma.

In questo programma di simulazione la stazione include le proprie conferme sia nei pacchetti dati sia nei pacchetti di controllo o di prenotazione; quindi il limite superiore e' determinato dalla lunghezza della supertrama.

Per consentire la trasmissione delle conferme sia i pacchetti di controllo che i pacchetti dati contengono un boolean array la cui lunghezza e' eguale alla finestra di ACK, cosi' che si abbia un bit per ogni slot compresa nella finestra. Come si vede, la finestra di ACK e', da un altro punto di vista, il massimo tempo per cui un ACK non puo' essere confuso con quello di un pacchetto diverso.

Una stazione destinataria che ha ricevuto correttamente un pacchetto setta ad 1 il bit di conferma per la slot in cui questo e' stato spedito e invia tale informazione con il primo pacchetto da spedire. Tutte le stazioni riceventi confrontano i bit di conferma con una tabella - denominata WAITFORACK - che memorizza le slot e le stazioni da cui e' in attesa di conferma.
Il controllo e' fatto ricorrendo alla seguente procedura, che funziona ugualmente bene per il punto-punto e il "multidestination":

```
!*** performs ack test ********************;
 procedure ACKTEST(SLOT,STAT,CURRTIME);
    ref(STATION) STAT;
    ref(MESSAGE) SLOT;
    long real CURRTIME;
 begin
    ref(DATASLOT) CURRSLOT;
    long real DT;
    short integer I,J;
    boolean ACKFLAG;

    for I:=0 step 1 until NWIND do
       if SLOT.ACKWORD(I) and STAT.WAITFORACK(SLOT.IDFROM,I) then
       begin
          STAT.WAITFORACK(SLOT.IDFROM,I):=false;
          ACKFLAG:=false;
          for J:=0 step 1 until NSTAT do
             ACKFLAG:=ACKFLAG or STAT.WAITFORACK(J,I);
          if not ACKFLAG then
             <dequeue packet from wait-for-ack queue>;
       end;
end*** OF PROCEDURE ACKTEST ***;
```

Per ridurre gli effetti di errori locali nella ricezione delle conferme, il bit di ACK relativo ad un particolare pacchetto e' settato a 1 dalle stazioni destinatarie in tutti i pacchetti dat e in tutti i pacchetti di controllo inviati durante una finestra di ACK.

Se la stazione mittente non riceve la conferma richiesta entro una finestra di ACK, manda il pacchetto in ritrasmissione accodandolo nell'apposita coda, che ha priorita' su quella dei pacchetti nuovi arrivati. La procedura e' la seguente:

```
!*** timeout procedure ********************;
 procedure TIMEDEQUEUE(STAT,PROGSLOT,CURRTIME);
    ref(STATION) STAT;
    long real CURRTIME;
    short integer PROGSLOT;

 begin
    short integer I;
    boolean TIMEFLAG;

   !*** ready also for broadcasting ***;
    for I:=0 step 1 until NSTAT do
    begin
       TIMEFLAG:=TIMEFLAG or STAT.WAITFORACK(I,PROGSLOT);
       STAT.WAITFORACK(I,PROGSLOT):=false;
    end;
    if TIMEFLAG then
       <put packet in the retransmission queue>;
 end*** OF PROCEDURE TIMEDEQUEUE ***;
```

Si noti che questo modo di procedere non consente di scoprire gli eventuali pacchetti duplicati, poiche' la numerazione di un pacchetto copre solo una finestra di ACK e non l'intero intervallo di tempo in cui il pacchetto potrebbe essere ritrasmesso. A un livello piu' alto si deve percio' disporre di un protocollo che permetta di eliminare i duplicati.

4.0 VARIABILI CONSIDERATE NELLA SIMULAZIONE

Dal momento che il canale e' slottizzato, e' comodo assumere la slot come unita'di misura del tempo (vedi &refintr.); i ritardi sono pertanto espressi il slot, per l'esattezza in virtual slot, mentre i valori di traffico e throughput sono espressi il pacchetti per slot.

I valori delle variabili di seguito considerate sono stati calcolati nel modo seguente.
Si considera l'intervallo di tempo intercorrente tra due eventi significativi: invio od arrivo di un pacchetto, ingresso o uscita di un pacchetto da una coda, etc..
Se si tratta di lunghezza di code o ritardi, il valore misurato e' attribuito a tutto l'intervallo di tempo intercorrente tra due eventi significativi successivi; se si tratta invece di traffico, di throughput o variabili stocastiche analoghe si attribuisce alla variabile per tutto l'intervallo di tempo considerato il numero di pacchetti in gioco diviso per tale intervallo.

Per ognuna delle variabili stocastiche vengono calcolati il valore minimo e il valore massimo durante il tempo di simulazione, la media e la varianza.

Chiaramente media e varianza hanno significato nell'ipotesi che il processo stocastico sia stazionario e pertanto l'utilizzazione dei dati ricavati dal programma di simulazione e' subordinata ad una verifica della stazionarieta' del processo simulato.

4.1 CANALE

4.1.1 Traffico

Il traffico sul canale e' una variabile stocastica il cui valore e' calcolato nel modo descritto considerando tutti i pacchetti trasmessi dalle stazioni in una slot (validi o collisi e affetti da rumore).

4.1.2 Occupazione del canale

L'occupazione del canale e' una variabile stocastica il cui valore e' calcolato nel modo descritto considerando i pacchetti eventualmente collisi come un solo pacchetto.

4.1.3 Throughput

Il throughput e' una variabile stocastica il cui valore e' calcolato nel modo descritto considerando solo i pacchetti correttamente trasferiti dal canale (ne' collisi, ne' affetti da rumore sull'up-link).

4.1.4 Ritardo

Il ritardo e' una variabile stocastica il cui valore attribuito nel modo descritto e' il tempo intercorrente tra la generazione del pacchetto e la corretta ricezione da parte della stazione destinataria.

4.1.5 Rango delle collisioni

Il rango delle collisioni e' una variabile stocastica il cui valore attribuito nel modo descritto e' dato dal numero di pacchetti che hanno colliso in una data slot. Il rango e' definito come il numero di tali pacchetti meno uno.

4.2 STAZIONI

4.2.1 Traffico

Il traffico in uscita dalla stazione e' una variabile stocastica il cui valore e' calcolato nel modo descritto considerando tutti i pacchetti trasmessi dalla stazione in una slot, siano essi correttamente trasmessi o collisi o affetti da rumore o ritrasmessi).

4.2.2 Throughput

Il throughput e' una variabile stocastica il cui valore e' calcolato nel modo descritto considerando solo i pacchetti correttamente trasferiti alla o alle stazioni destinatarie.

4.2.3 One-way delay

Questo ritardo e' una variabile stocastica il cui valore attribuito nel modo descritto e' il tempo intercorrente tra la generazione del pacchetto e la corretta ricezione da parte di tutte le stazioni destinatarie.

4.2.4 Two-way delay

Questo ritardo e' una variabile stocastica il cui valore attribuito nel modo descritto e' il tempo intercorrente tra la generazione del pacchetto e la corretta ricezione dell'ACK proveniente da tutte le stazioni destinatarie. E' il tempo per cui un pacchetto resta nella coda dei pacchetti in attesa di ACK.

4.2.5 Lunghezza delle code

La statistica sulla lunghezza delle code e' fatta separatamente per il periodo di tempo totale della simulazione e per la parte di questo in cui la coda non e' vuota. I valori nei due casi sono infatti significativi insieme al fine di valutare le modalita' di occupazione dei buffer. Ciascuna delle quattro code presenti nella stazione - pacchetti nuovi, pacchetti da ritrasmettere, pacchetti in attesa di ACK e pacchetti in attesa di uscita - ha la propria statistica separata.

4.2.6 Occupazione del pool dei buffer

Per il pool dei buffer viene tracciato l'istogramma che descrive la frazione del tempo totale per cui il pool e' occupato nelle percentuali fissate.

4.3 GENERATORI DI TRAFFICO

Nell'attuale versione non vengono effettutate statistiche sui singoli generatori di traffico, le statistiche riguardano il traffico globale generato per una data stazione.

4.3.1 Traffico effettivamente generato

La generazione di un pacchetto e' subordinata alla disponibilita' di spazio nel pool dei buffer della stazione. Viene effettuata pertanto una statistica del traffico effettivamente generato.

4.3.2 Traffico perso

Per il medesimo motivo viene effettuata una statistica del traffico perso rispetto ai valori imposti inizialmente ai generatori di traffico.

5.0 SIMULAZIONE DEL CANALE COMUNE.

Il canale, per consentire la simulazione delle collisioni e del ritardo di propagazione, e' realizzato nel programma tramite due code e due processi asincroni.

I "sender" delle stazioni di terra schedulate per la trasmissione nella slot corrente, accodano i pacchetti nella prima delle code, INQUEUE, di cui e' composta la simulazione del canale e attivano il primo dei processi asincroni, CHANNELIN. Questo effettua il controllo delle collisioni e le statistiche relative al traffico offerto al canale, alla sua occupazione e al rango delle collisioni stesse. Se non si ha collisione il pacchetto viene accodato nell'altra coda, OUTQUEUE, e ove necessario il secondo processo, CHANNELOUT, viene schedulato al tempo corrente piu' il ritardo di propagazione.
Il "processor" ha la struttura seguente:

```
Process class CHANNELIN(CHAN);
   ref(CHANNEL) CHAN;
begin
    ref(MESSAGE) CURRMSG,CURRMSG1;
    long real DT,TM;
    short integer NMSG;

    while true do
    begin
       Passivate;
       NMSG:=CHAN.INQUEUE.Cardinal;
       TM:=Time;
       if (NMSG gt 0) then
       begin
          CURRMSG:-CHAN.INQUEUE.First;
          <statistics for collision rank>;
          <statistics for channel traffic>;
         !*** simulation of noise and collisions ***;
          if (not CURRMSG.NOISEFLAG) and (NMSG eq 1) then
          begin
             CURRMSG.Out;
             CURRMSG.Into(CHAN.OUTQUEUE);
            !**incoming packet reactivate channelout if idle**;
             if CHAN.CHANOU.Idle then
             begin
                CURRMSG1:-CHAN.OUTQUEUE.First;
                reactivate CHAN.CHANOU at (CURRMSG1.TRANSTIME+
                                    PRDELAY+CURRMSG1.TLENGTH);
             end;
          end;
          CHAN.INQUEUE.Clear;
       end;
    end;
```

```
 end****OF CLASS CHANNELIN****;
```

Il secondo processo, quando attivato, effettua le statistiche sul throughput e sul ritardo medio di canale, poi rende disponibile il pacchetto in una variabile globale, RECMSG, attiva i ricevitori delle varie stazioni e, se ha ulteriori pacchetti in coda, si rischedula ad un tempo pari al tempo di trasmissione del primo pacchetto in coda piu' il ritardo di propagazione. Se invece la coda e' vuota il processo si passiva e verra' riattivato da CHANNELIN.
Il "processor" ha la seguente struttura:

```
 Process class CHANNELOUT(CHAN);
    ref(CHANNEL) CHAN;
 begin
    ref(MESSAGE) NEWMSG;
    long real TM,DT;
    short integer I;

    Passivate;
    while true do
    begin
       TM:=Time;
       RECMSG:-CHAN.OUTQUEUE.First;
       if (RECMSG == none) then
          ERROR("outqueue empty")
       else
       begin
          RECMSG.Out;
          <statistics for channel throughput and delay>;
          for I:=0 step 1 until NSTAT do
             reactivate UPSTATION(I).RECV at TM;
         !**rescheduling of outchannel if outqueue not empty**
              **overrides existing scheduling if any ***********;
          NEWMSG:-CHAN.OUTQUEUE.First;
          if (NEWMSG =/= none) then
             reactivate Current at(NEWMSG.TRANSTIME+PRDELAY+
                                   NEWMSG.TLENGTH)
          else
             Passivate;
       end;
    end;

 end****OF CLASS CHANNELOUT****;
```

Nel programma di simulazione si ha poi un ulteriore classe che funziona da struttura dati e da programma principale per l'attivazione del canale; essa e' stata introdotta essenzialmente per motivi di leggibilita' del programma. La sua struttura e' la seguente:

```
 Process class CHANNEL;
 begin
    ref(CHANNELIN) CHANIN;
```

```
    ref(CHANNELOUT) CHANOU;
    ref(Head) INQUEUE,OUTQUEUE;

    INQUEUE:-new Head;
    OUTQUEUE:-new Head;
    CHANIN:-new CHANNELIN(this CHANNEL);
    CHANOU:-new CHANNELOUT(this CHANNEL);
    activate CHANIN;
    activate CHANOU;
    Passivate;
   !*** reactivated by application main ***;
    Cancel(CHANIN);
    Cancel(CHANOU);

end****OF CLASS CHANNEL****;
```

6.0 FORMATO DEI PACCHETTI

Nel nostro modello si hanno due tipi di pacchetti: i pacchetti dati e i pacchetti di controllo.
Entrambi hanno la seguente struttura di base:

```
Link class MESSAGE;
begin
      boolean array ACKWORD(0:NWIND),IDTO(0:NSTAT);
      long real GENTIME;
      real TLENGTH,TRANSTIME,LBIT;
      integer IDFROM;
      short integer TYPE,ACKSLOT;

end****OF CLASS MESSAGE****;
```

I pacchetti dati hanno la seguente struttura:

```
MESSAGE class DATASLOT;
begin
      text DATA;

end****of class dataslot****;
```

I pacchetti di controllo o sono costituiti dalla sola parte base, oppure hanno la forma gia' vista dei pacchetti di prenotazione.

Della parte base:

- la variabile ACKWORD e' impiegata nel protocollo di conferma dei dati ed e' stata discussa in precedenza (vedi 3.0).

- la variabile IDTO e' un boolean array la cui lunghezza e' pari al numero di stazioni previste in quella particolare simulazione. Quando viene generato un pacchetto vengono settate a TRUE le posizioni corrispondenti alle stazioni destinatarie. L'operazione e' fatta tramite una procedure DESTINATION interna al generatore di pacchetti a cui in fase di inizializzazione vengono fornite le indicazioni sui destinatari. In tal modo e' possibile simulare sia destinazioni singole che multidestination.

- la variabile IDFROM identifica la stazione mittente, e' impiegata nel protocollo di conferma della ricezione dei pacchetti e per memorizzare i dati sull'attivita' delle statzioni.

- la variabile GENTIME contiene il tempo a cui il pacchetto e' stato generato; il valore relativo e' impostato dal generatore di pacchetti e la variabile in questione serve per ottenere dati sul one-way delay e sul two-way delay.

- la variabile TLENGTH contiene la lunghezza del pacchetto in sec., tale lunghezza e' comprensiva dei tempi di guardia ed e' usata per schedulare l'attivita' dei vari processi durante la simulazione. Questa, infatti, non e' un processo real-time, ma il tempo e' simulato assegnando ad una variabile interna, EVTIME, ogniqualvolta un processo diventa attivo, il valore del tempo a cui era stato schedulato.

- la variabile TRANSTIME contiene il tempo a cui il pacchetto e' stato trasmesso, viene usata nella schedulazione dei processi e per la simulazione del ritardo di propagazione.

- le due variabili TYPE e ACKSLOT contengono il tipo di pacchetto e il numero di slot (vedi 3.0), sono entrambe usate nel protocollo di conferma della ricezione dei dati.

- la variabile LBIT contiene la lunghezza in bit del pacchetto ed e' usata nella simulazione degli effetti della BER (vedi 7.0).

7.0 SIMULAZIONE DEL RUMORE E DEI DISTURBI

Per rendere migliore la simulazione dei disturbi nel programma di simulazione sono separati i disturbi sull'up-link da quelli sul down-link.

Entrambi i disturbi sono simulati nella stazione, uno sul trasmettitore e l'altro sul ricevitore. Poiche' nel programma si fa ricorso a due funzioni che hanno come parametro il livello di Bit Error Rate (BER), e' possibile simulare disturbi aventi differenti leggi di variazione nel tempo sostituendo il contenuto delle due funzioni entro il programma, e differenti livelli di BER fornendo in input valori diversi ai parametri passati a tali funzioni.

Un esempio di tale simulazione e' il seguente:

```
real procedure UNOISEN;
begin
   long real X,T;

   T:=Time+TPH;
   X:=8.25+(0.25*Cos(6.28*T))+(2.0*Cos(0.628*T));
   X:=SNOISE*(10.0**(2.8-(0.8*X)));
   UNOISEN:=X;
end*** OF PROCEDURE UNOISEN **********************;

real procedure DNOISEN;
begin
   long real X,T;

   T:=Time+TPH;
   X:=8.25+(0.25*Cos(6.28*T))+(2.0*Cos(0.628*T));
   X:=RNOISE*(10.0**(2.8-(0.8*X)));
   DNOISEN:=X;
end*** OF PROCEDURE DNOISEN **********************;
```

8.0 SIMULAZIONE DELLE STAZIONI DI TERRA

Ciascuna stazione di terra e' simulata mediante due processi che operano in modo asincrono: il trasmettitore (sender) e il ricevitore. Vi e' poi un insieme di variabili e di code per la gestione dei pacchetti in ingresso e uscita e per realizzare la ritrasmissione dei pacchetti persi. Non si ha invece nel programma di simulazione un meccanismo per scartare gli eventuali duplicati.

Il sender svolge le funzioni seguenti:

- mantiene la scansione temporale della slottizzazione TDMA del canale
- mantiene la scansione temporale della trama e della finestra di ACK (vedi 3.0)
- effettua la trasmissione sulla base dell'algoritmo distribuito prescelto per la scmedulazione dell'accesso al canale
- decide la ritrasmissione dei pacchetti trascorsa la finestra di ACK secondo il protocollo visto (vedi 3.0)
- effettua la simulazione dei disturbi sul proprio up-link (vedi 7.0).
- le statistiche relative al traffico in uscita dalla stazione, al two-way delay e alle code

Il processo che esegue le operazioni indicate ha la struttura seguente:

```
Process class SEND(STAT);
   ref(STATION) STAT;
begin
   long real TM;
   real TRLENGTH;
   integer I,J,ISLOT;
   boolean TRAN,RESFLAG;

  !*** avoid false statistics at the start ***;
   RESFLAG:=0 ne STATRES;
   TM:=Time;
   while (TM le SIMTIME) do
   begin
      if RESFLAG then
      begin
         for I:=0 step 1 until (STATRES-1) do
         begin
            if (I eq STAT.ID) then
            begin
```

```
                !*** transmit reservation slots ***;
                 TM:=Time;
                 UTI.RTRANSMIT(STAT,TM);
                 if CHAN.CHANIN.Idle then
                    reactivate CHAN.CHANIN at (TM+LRESVSLOT) prior;
              end;
              Hold(LRESVSLOT);
           end;
           RESFLAG:=false;
        end
        else
        begin
           TM:=Time;
           if (ISLOT eq UPDSLOT) and RTDMAFLAG then
             !*** update CQtable only for R-TDMA ***
                  *** already all entries are 0       ***;
              UTI.RSVUPDATE(STAT);
          !**timeout test**************;
           UTI.TIMEDEQUEUE(STAT,STAT.SENDPSN,TM);
          !**transmission of data**********;
           I:=UTI.SENDSCHED(STAT,TM);
           TRLENGTH:=UPSTATION(I).LDATA;
           TRAN:=false;
           if SALOHAFLAG then
             !*** S-ALOHA  **********************
                  *** slots must be of equal length***;
              TRAN:=UTI.SALQSCHED(STAT,TM)
           else
              if (I eq STAT.ID) then
                 TRAN:=UTI.RTDMAQSCHED(STAT,TM);
           if TRAN and CHAN.CHANIN.Idle then
             !**avoid multiple reactivations at the same time**;
              reactivate CHAN.CHANIN at (TM+STAT.LDATA) prior;
           STAT.SENDPSN:=Mod(STAT.SENDPSN+1,ACKWIND);
           STAT.STARTFLAG:=STAT.STARTFLAG or (0 eq STAT.SENDPSN);
           ISLOT:=Mod(STAT.SENDPSN,LRFRAME);
           RESFLAG:=(0 eq ISLOT);
           Hold(TRLENGTH);
        end;
     end;

end*****OF CLASS SEND *****;
```

Il ricevitore a sua volta effettua:

- l'acquisizione dei pacchetti destinati alla stazione e il loro accodamento sulla coda di uscita
- la parte del protocollo di ACK di sua competenza (vedi 3.0)
- la simulazione dei disturbi sul proprio down-link (vedi 7.0)
- le statistiche relative al throughput e al one-way delay delle altre stazioni.

Il processo che esegue le operazioni indicate ha la struttura seguente:

```
Process class RECEIVE(STAT);
   ref(STATION) STAT;
begin
   ref(DATASLOT) RMSG;
   long real TM,DT;
   integer I,U;

   U:=(2350*((STAT.ID+1)*2))+1;
   for I:=1 step 1 until 1000 do
      Draw(0.5,U);
   while true do
   begin
      Passivate;
      if RECMSG.IDTO(STAT.ID) then
      begin
         if (not Draw(STAT.DNOISEN*RECMSG.LBIT,U)) then
         begin
           !***good message and destination station***;
            TM:=Time;
            if (RECMSG.TYPE eq 1) then
            begin
               UTI.ACKTEST(RECMSG,STAT,TM);
               if RTDMAFLAG then
                 !***store reservation for R-TDMA only***;
                  UTI.RESVSTORE(RECMSG qua RESVSLOT,STAT);
            end
            else
               if (RECMSG.TYPE eq 2) then
               begin
                  if (UTI.BUFFLOAD(STAT) lt STAT.MAXBUFF) then
                  begin
                     <statistics on throughput>;
                     <statistics on one-way delay>;
                     <statistics on two-way delay>;
                    !***ack for received packet***;
                     STAT.RACKWORD(RECMSG.ACKSLOT):=true;
                     UTI.ACKTEST(RECMSG,STAT,TM);
                     RMSG:-new DATASLOT;
                     RECMSG qua DATASLOT.SLCOPY(RMSG);
                     <statistics on output queue length>;
                     RMSG.Into(STAT.QUEUE(4));
                     if USINKMSG(STAT.ID).Idle then
                        reactivate USINKMSG(STAT.ID) at
                          (TM+USINKMSG(STAT.ID).WTIME);
                  end;
               end
               else
                  ERROR("Msg type err. 01 ");
         end;
      end;
   end;
```

```
end*****OF CLASS RECEIVE *****;
```

Sia il sender che il receiver operano come processi asincroni.

A questi sono da aggiungere la classe STATION che funziona da base per le attivazioni dei due processi descritti e contiene le code e le altre variabili proprie della stazione.
La sua struttura e' la seguente:

```
Process class STATION(ID,AGATE,NGATE,RNOISE,SNOISE,
                      MAXBUFF,LDATA,LPACK);
   short integer ID,MAXBUFF;
   real RGATE,NGATE,RNOISE,SNOISE,LDATA,LPACK;
begin
   ref(Head) array QUEUE(1:4);
   ref(Head) array WFAQUEUE(0:NSTAT);
   integer array CQTAB(0:NSTAT),RESVTAB(0:NSTAT,1:3);
   boolean array RACKWORD(0:NWIND);
   boolean array WAITFORACK(0:NSTAT,0:NWIND);
   real array DISTBUFPAT(0:10),DISTBUF(0:11);
   ref(RECEIVE) RECV;
   long real BUFTIME,TBUFTIME;
   real TSTART,TEND,RATEI,SD1I,TPH;
   integer I,CQPTR,TOTRES,HEARD,STATUP,SENDPSN,U1,U2,UN;
   boolean SYNCFLAG,STARTFLAG;

   real procedure UNOISEN.....;
   real procedure DNOISEN.....;
   U1:=(150*(ID+1)*2)+1;
   U2:=(350*(ID+1)*2)+1;
   UN:=(250*(ID+1)*2)+1;
   for I:=1 step 1 until 100 do
   begin
      Uniform(0,1,U1);
      Uniform(0,1,U2);
   end;
  !*** random phase to avoid correlation in noise ****;
   TPH:=Uniform(0.0,10.0,U1);
   for I:=0 step 1 until 10 do
       DISTBUFPAT(I):=I/10.0;
   for I:=1 step 1 until 4 do
      QUEUE(I):-new Head;
   SYNCFLAG:=true;
   STATUP:=UPSTAT;
   Passivate;
  !*** after traffic generator activation ***;
   RECV:-new RECEIVE(this STATION);
   activate new SEND(this STATION);
   activate RECV;
   Passivate;
  !***reactivated by application main ***;
   Cancel(RECV);
```

```
end***OF CLASS STATION****;
```

9.0 SIMULAZIONE DEGLI HOST

Gli host sono simulati in maniera molto semplificata, cioe' come uno o piu' generatori di traffico in ingresso alla stazione e come un unico processo che preleva traffico dalla stazione.

I generatori del traffico in ingresso alla stazione hanno la struttura seguente:

```
!*** generate packets ****************;
Process class GENMSG(STAT,RATE,SD1,TYPE,NPACK,MPACK,MPACKFLAG,DESTINA);
   value DESTINA; boolean array DESTINA;
   ref(STATION) STAT;
   real RATE,SD1;
   short integer TYPE,NPACK,MPACK;
   boolean MPACKFLAG;
begin
   ref(DATASLOT) GCURRMSG;
   long real TN,DT;
   real INPACK,RATEI,SD1I;
   integer UP,UW,UN,CURRPACK,RPACK;
   short integer J;
   boolean DESTFLAG;

   real procedure WAITIME(A,B,TYPE);
      real A,B;
      short integer TYPE;
   begin
      WAITIME:=if (TYPE eq 0) then
                  Erlang(A,B,UW)
               else
                  A;
   end*** OF PROCEDURE WAITIME ***;

   procedure DESTINATION(DESTINA,GCURRMSG);
      value DESTINA; boolean array DESTINA;
      ref(DATASLOT) GCURRMSG;
   begin
      short integer IDEN,I;

      if DESTFLAG then
      begin
         for I:=0 step 1 until NSTAT do
            GCURRMSG.IDTO(I):=DESTINA(I);
      end
      else
      begin
        !** destination has to be random distributed **;
         for IDEN:=(Randint(1,SPATNUM,UP)-1)
                     while (IDEN eq STAT.ID) do;
         GCURRMSG.IDTO(IDEN):=true;
```

```
    end;
 end*** OF PROCEDURE DESTINATION ***;

!*** print data of traffic generator ***************;
 procedure GENDUMP;
 begin
    Outtext("Station");Outint(STAT.ID+1,6);Outimage;
    Outtext("Interarrival time distribution: ");
    Outtext(if (TYPE eq 0 ) then "Erlang" else "fixed rate");
    Outimage;
    Outtext("Imposed messages generation rate (pkt/slot) ");
    Outreal(RATE,6,12);
    Outimage;
    if (TYPE eq 0) then
    begin
       Outtext("Imposed standard deviation                    ");
       Outreal(SD1,6,12);
       Outimage;
    end;
    if MPACKFLAG then
    begin
       Outtext("Multipacket traffic ");Outimage;
       Outtext("Multipacket number distribution  "); Outimage;
    end;
    Outimage; Outimage;
 end*** OF PROCEDURE GENDUMP ********************;

 UN:=(800*(STAT.ID+1)*2)+1;
 INPACK:=1.0/NPACK;
 UP:=(1000*(STAT.ID+1)*2)+1;
 UW:=(2500*(STAT.ID+1)*2)+1;
 for I:=0 step 1 until NSTAT do
    DESTFLAG:=DESTFLAG or DESTINA(I);
 if (TYPE eq 0) then
 begin
    RATEI:=RATE/STAT.LDATA;
    SD1I:=(1/RATEI*SD1)**2.0;
 end
 else

    RATEI:=STAT.LDATA/RATE;
 GENDUMP;
   !*** avoid coincidence at the start ***;
 reactivate Current at(TM+WAITIME(RATEI,SD1I,
                                  TYPE));
   !**multipacket generation, 1 eq single packet**;
 NPACK:=if (MPACK ne 0) then  NPACK else 0;
 TM:=Time;
 while (TM le SIMTIME) do
 begin
    TM:=Time;
    if (MPACK eq 0) then
       NPACK:=Entier(Negexp(INPACK,UN))+1;
    CURRPACK:=STAT.MAXBUFF-UTI.BUFFLOAD(STAT);
```

```
            RPACK:=NPACK-CURRPACK;
            if (RPACK gt 0) then
               <statistics on refused packets>;
            if (CURRPACK gt 0) then
            begin
               <statistics on new packets queue>;
               CURRPACK:=if (CURRPACK ge NPACK) then NPACK else CURRPACK;
               for J:=1 step 1 until CURRPACK do
               begin
                  GCURRMSG:-new DATASLOT;
                  DESTINATION(DESTINA,GCURRMSG);
                  GCURRMSG.IDFROM:=STAT.ID;
                  GCURRMSG.GENTIME:=TM;
                  GCURRMSG.LBIT:=STAT.LPACK;
                  GCURRMSG.TLENGTH:=STAT.LDATA;
                  GCURRMSG.Into(STAT.QUEUE(2));
               end;
               <statistics on effective rate>;
            end;
            reactivate Current at (TM+WAITIME(RATEI,SD1I,TYPE));
         end;

      end****OF CLASS GENMSG****;
```

I generatori di traffico hanno la possibilita' di generare pacchetti di dati:

* ad intervalli di tempo tra una generazione e l'altra che possono:
  - essere costanti
  - seguire una distribuzione di Erlang
* ad ogni generazione possono venir generati:
  - un solo pacchetto
  - piu' pacchetti e in questo caso il loro numero puo essere fisso o variare secondo una distribuzione esponenziale negativa troncata
* destinati
  - a prefissati accessi o gruppi di accessi al canale
  - a tutti gli accessi secondo una distribuzione uniforme tra questi

La possibilita' di accodare pacchetti nella coda dei nuovi arrivi alla stazione e' legata alla disponibilita' di buffer. Per questo ogni generatore di traffico contribuisce ad una statistica relativa al traffico effettivamente immesso nella stazione.

Il processo che preleva traffico dalla stazione ha invece la struttura seguente:

```
!*** sink packets ****************;
Process class SINKMSG(STAT,SINKRATE);
   ref(STATION) STAT;
   real SINKRATE;
begin
   ref(DATASLOT) GCURRMSG;
   long real TM,DT;
   real SINKTIME,WTIME;
   short integer J;

   procedure SINKDUMP;
   begin
      Outtext("Station");Outint(STAT.ID+1,6);Outimage;
      Outtext("Imposed messages sink rate (pkt/slot)");
      Outreal(SINKRATE,6,12); Outimage; Outimage;
   end*** OF PROCEDURE SINKDUMP ******************;

   WTIME:=STAT.LDATA/SINKRATE;
   SINKTIME:=SIMTIME+(2*PRDELAY);
   SINKDUMP;
   TM:=Time;
   while (TM lt SINKTIME) do
   begin
      TM:=Time;
      J:=STAT.QUEUE(4).Cardinal;
      if (J gt 0) then
      begin
         GCURRMSG:-STAT.QUEUE(4).First;
         <statistics on effective rate>;
         <statistics on output packets queue>;
         GCURRMSG.Out;
      end;
      if (J gt 1) then
         Hold(WTIME)
      else
         Passivate;
   end;

end****OF CLASS SINKMSG****;
```

Per semplicita' il traffico e' prelevato con cadenza costante; anche in questo caso viene eseguita una statistica del traffico prelevato dalla stazione.

```
    begin
       short integer I,J;

       for I:=6 step 1 until 15 do
       begin
          for J:=1 step 1 until 4 do
             REPORTS.Outfix(0.0,6,12);
          REPORTS.Outimage;
       end;
    end;
    Outimage; Eject(100);
    Outtext("STATIONS ");
end***OF PROCEDURE CHANDUMP***;
```

* per i dati relativi alle singole stazioni, memorizzati sul file 'REPORTS' e riportati sul printout:

```
procedure STATDUMP(STAT);
   ref(STATION) STAT;
begin
   Outimage; Outimage; Outimage;
   Outtext("Station number "); Outint(STAT.ID+1,4);
   Outimage; Outimage;
   Outtext("Rgate   ");Outreal(RGATE,6,12);Outimage;
   Outtext("Ngate   ");Outreal(NGATE,6,12);Outimage;
   Outimage;
   Outtext("Data slot length  ");
   Outfix(STAT.LDATA,8,14);
   Outtext("    (slot)");Outimage;
   Outtext("                  ");
   Outfix(STAT.LPACK,8,14);
   Outtext("    (bit)");Outimage; Outimage;
   Outtext("Up link noise factor     ");
   Outreal(STAT.SNOISE,6,12); Outimage;
   Outtext("Down link noise factor   ");
   Outreal(STAT.RNOISE,6,12); Outimage; Outimage;
   Outtext("Buffer pool allocation (pkt)  ");
   Outint(STAT.MAXBUFF,6); Outimage; Outimage;
   Outtext("Effective messages generation rate (pkt/slot)");
   Outimage;
   WREPORT(STAT.ID,0,false);  Outimage;
   Outtext("Generated messages loss rate (pkt/slot)");
   Outimage;
   WREPORT(STAT.ID,9,false);  Outimage;
   Outtext("Effective messages sink rate (pkt/slot)");
   Outimage;
   WREPORT(STAT.ID,10,false);  Outimage;
   Outtext("Traffic of outgoing packets (pkt/slot) "); Outimage;
   WREPORT(STAT.ID,1,false);  Outimage;
   Outtext("Throughput of outgoing packets (pkt/slot) ");
   Outimage;
   WREPORT(STAT.ID,2,false);  Outimage;
   Outtext("One-way delay of outgoing packets (slot) "); Outimage;
```

10.0 STRUTTURA DEL PROGRAMMA DI SIMULAZIONE

```
begin
external assembly procedure ERROR;

ref(Infile) DATA;
ref(Outfile) REPORTS,HEADREP,DISTRIB;
long real LREFSLOT;     !**to improve precision in statistics**;
real SIMTIME,PRDELAY,RGATE,NGATE,FUZZ,RATE1,RATE2,ERRBITC,CRATE,
       RNOISE,LRESVSLOT,LDATASLOT,MAXBUFF,LDATAPACK,RATELEAD,SD1,
       LRESVPACK,NOISELEVD,NOISELEVR,NOISELEV,LOADFAC;
short integer LRFRAME,STATNUM,I,J,K,K1,LEADER,ACKWIND,UPSTAT,
       UPDSLOT,NWIND,NSTAT,MAXRES,STATRES,TYPE,NPACK,MPACK,
       NPACKLEAD,NLFRAME;
boolean RTDMAFLAG,SALOHAFLAG,HISTOFLAG,BALANCEFLAG,MPACKFLAG;

procedure READGLOBALS;
  ..................
Simulation begin
   ref(UTILITY) UTI;
   ref(REPORT) REP;
   ref(MESSAGE) RECMSG;
   ref(CHANNEL) CHAN;
   ref(STATION) array UPSTATION(0:NSTAT);
   ref(SINKMSG) array USINKMSG(0:NSTAT);
   boolean array DESTINA(0:NSTAT);

   Process class CHANNELIN(CHAN).............;
   Process class CHANNELOUT(CHAN)............;
   Process class CHANNEL.....................;
   Link class MESSAGE........................;

   MESSAGE class RESVSLOT....................;
   begin
      procedure SLCOPY(MSLOT)................;
      ...............
   end****OF CLASS RESVSLOT****;

   MESSAGE class DATASLOT....................;
   begin
      procedure SLCOPY(MSLOT).........;
      ........................
   end****OF CLASS DATASLOT****;

   !*** generate packets ***************;
   Process class GENMSG(STAT,RATE,SD1,TYPE,NPACK,
                  MPACK,MPACKFLAG,DESTINA)..........;
   begin

      real procedure WAITIME(A,B,TYPE).........;
      procedure DESTINATION....................;
```

```
        ..........................
      end*** OF PROCEDURE SYNCRESET ***;
    end*****OF CLASS UTILITY*****;

    class DISTPACK(VBEG,VEND,STEPS,GEOFLAG).........;
    begin
      procedure DUMP.........;
      ......................
    end***** OF CLASS DISTPACK **********************;

    class REPORT;
    begin
      !*** spill data for report procedure **************;
       procedure MEASURE(ID,TYPE,VAL,PROB).......;

      !*** calculates and writes data for the report *********;
       procedure WREPORT(ID,TYPE,SKIP)...........;

      !*** calculates and writes data for the report *********;
       procedure WREPORT1(ID,TYPE,SKIP)..........;

       procedure REPHEAD.........;

      !*** dump for channel report ***************;
       procedure CHANDUMP........;

      !*** dump for station report *********;
       procedure STATDUMP(STAT).......;

       procedure HISTODUMP.........;
       ...................................
    end****OF CLASS REPORT****;

    Process class SEND(STAT)................;
    Process class RECEIVE(STAT).........;
    Process class STATION(ID,RGATE,NGATE,RNOISE,
                          SNOISE,MAXBUFF,LDATA,LPACK).........;
    begin
      real procedure UNOISE.....;
      real procedure DNOISE.....;
      real procedure UNOISEN.....;
      real procedure DNOISEN.....;
      ......................
    end***OF CLASS STATION****;
    Process class TIMESERIES(ID,TYPE,DTIME,FILENAM).......;
    procedure READSTATDATA.............;
    procedure READGENTRAF..............;
    procedure READSINKTRAF.............;
    procedure READTSERIES..............;
    ......................
end*** OF SIMULATION BLOCK ****;
    ..........................
end******** OF THE PROGRAM ************
```

- il fattore per cui devono essere moltiplicati i valori di traffico imposti ai vari generatori, cio' per facilitare serie di prove intese a tracciare curve di performance in situazioni analoghe.

Per ogni punto di accesso (stazione) debbono essere forniti una serie di dati che la caratterizzano. La procedura seguente li riporta nell'ordine:

```
!*** reads data for station from data file ***************;
procedure READSTATDATA;
begin
   !*** new line ************************;
    DATA.Inimage;
    if DATA.Endfile then
    begin
       Outtext("error in input data "); Outimage;
       ERROR("user err. 01");
    end;
   !*** data slot length ********************;
    LDATASLOT:=DATA.Inreal;
   !*** data packet length (bit) *************;
    LDATAPACK:=DATA.Inreal;
   !*** rgate and ngate values for S-ALOHA *****;
    RGATE:=DATA.Inreal; NGATE:=DATA.Inreal;
   !*** up-link bit error rate ***************;
    ERRBITC:=DATA.Inreal;
   !*** down-link bit rate *****************;
    RNOISE:=DATA.Inreal;
   !*** buffer pool allocation ***************;
    MAXBUFF:=DATA.Inint;
end*** OF PROCEDURE READSTATDATA ***************;
```

Come si vede debbono essere fornite nell'ordine le seguenti informazioni:

- numero di slot virtuali che compongono la slot per i dati
- lunghezza in bit del pacchetto dati
- i valori di R-GATE ed N-GATE per il protocollo S-ALOHA
- i valori della BER sull'up-link e sul down-link
- l'ampiezza del pool dei buffer in pacchetti dati
- il numero di generatori del traffico in ingresso alla stazione.

Per ogni generatore del traffico in ingresso alla stazione debbono venir forniti i dati che lo caratterizzano. La procedura e' la seguente:

```
    LRESVPACK:=DATA.Inreal;
   !*** number of stations ***************;
    STATNUM:=DATA.Inint;

   !*** simulation time (sec) ************;
    SIMTIME:=DATA.Inreal;
   !*** load reduction factor ************;
    LOADFAC:=DATA.Inreal;
   !*** number of traffic generators *******;
    K:=DATA.Inint;
 end*** OF PROCEDURE READGLOBALS ***********;
```

Come si puo' vedere la prima scheda contiene i dati relativi al canale nel seguente ordine:

- velocita' del canale in bit/sec
- ritardo di propagazione del canale in sec
  tale ritardo e' supposto uguale per tutte le stazioni non essendo simulati fenomeni di perdita di sincronismo sulla trama

Nelle schede successive, di solito una, sono contenute le informazioni relative al CAS adottato e alla struttura della trama. L'ordine e' il seguente:

- due flag che permettono di selezionare il CAS secondo lo schema seguente:

1	0	R-TDMA/F-TDMA
1	1	R-TDMA/S-ALOHA
0	0	F-TDMA multi-ack
0	1	S-ALOHA multi-ack

- il numero di slot riservate ai dati in goni trama
- il numero di slot di controllo in ogni trama, in particolare questo numero puo' essere 0, anche se nella pratica occorrono sempre slot di controllo per problemi di sincronizzazione di trama, controllo degli accessi attivi, etc.
- la finestra di ACK espressa in numero di slot
- la durata temporale delle slot di controllo in sec
- la lunghezza in bit del pacchetto di controllo, supposta uguale per tutte le stazioni
- il numero massimo di punti di accesso (stazioni)
- il tempo di simulazione in sec

```
!*** reads data for traffic generators ********************;
procedure READGENTRAF;
begin
      !*** new line ***************************;
    DATA.Inimage;
      !*** input packets distribution *******
              0 - erlang distribution of interarrival time
              1 - fixed rate                 *******;
    TYPE:=DATA.Inint;
      !*** input packets rate (pkt/slot) ****;
    RATE1:=LOADFAC*DATA.Inreal;
      !*** interarrival time distribution S.D. ***;
    SD1:=LOADFAC*DATA.Inreal;
      !*** number of packets per generation *;
    NPACK:=DATA.Inreal;
    NPACKFLAG:=NPACK gt 1;
      !*** distribution of the number of packets **
            per generation:
             0 - exp. distribution
             1 - fixed rate                 *******;
    NPACK:=DATA.Inreal;
      !*** set destination stations ****;
    for K:=0 step 1 until NSTAT do
       DESTINA(K):=1 eq DATA.Inint;
end*** OF PROCEDURE READGENTRAF ***************;
.sp
.fo
```

Come si vede debbono essere fornite su una o piu' schede per ogni generatore di traffico le informazioni seguenti:

- un flag per selezionare tra:
  0 - una distribuzione di Erlang del tempo tra due generazioni
  1 - un intervallo costante

- il valor medio della velocita' di generazione dei pacchetti in pacchetti per slot

- la deviazione standard per la distribuzione del tempo intercorrente tra due successive generazioni (nel caso non sia applicabile va' fornito un valore che il programma legge e ignora)

- il numero di pacchetti generati ogni volta, cosi da consentire la simulazione di generatori multipacchetto

- un flag per seleziomare tra:
  0 - una distribuzione esponenziale troncata del numero di pacchetti
  1 - un numero costante di pacchetti per ogni generazione

- una serie di 1 o di 0 in numero pari al numero delle stazioni per indicare se la stazione in questione e' destinataria o no del traffico generato; se compaiono tutti valori 0 il traffico viene ripartito tra tutte le stazioni secondo una distribuzione uniforme.

Per il traffico in uscita dalla stazione si ha la seguente procedura:

```
!*** reads data for sink ****************************************;
procedure READSINKTRAF;
begin
   !*** new line ************************;
    DATA.Inimage;
   !*** output packets rate (pkt/slot) ***;
    RATE2:=DATA.Inreal;
end*** OF PROCEDURE READSINKTRAF ***************;
```

Come si vede, deve essere fornita una scheda su cui e' riportata:

- la velocita' in pacchetti per slot a cui vengono prelevati i pacchetti

infatti si e' supposto in questa versione del programma un prelievo di pacchetti a velocita' costante.

Le schede sin qui descritte sono obbligatorie, quelle che seguono sono invece opzionali.

Per memorizzare dati relativi alla sequenza temporale dei valori assunti dalle variabili misurate e' stata predisposta la procedura seguente:

```
!*** reads data for time series recording (if any) ******;
procedure READTSERIES;
begin
    while not DATA.Endfile do
    begin
       activate new TIMESERIES(DATA.Inint-1,DATA.Inint,DATA.Inreal,
                 DATA.Intext(9).Sub(2,8));
       DATA.Inimage;
    end;
end*** OF PROCEDURE READTSERIES ***************;
```

Come si vede debbono essere fornite tante schede quante sono le variabili di cui debbono venir memorizzati i valori, ciascuna contenente nell'ordine le informazioni seguenti:

- il numero d'ordine della stazione o il numero totale delle stazioni + 1 per il canale

- un identificatore della variabile, secondo la tabella seguente:

```
0 - velocita' effettiva di ingresso dei pacchetti
1 - traffico offerto al canale
2 - throughput
3 - one-way delay
4 - two-way delay
5 - coda dei pacchetti in ritrasmissione
```

6 - coda dei pacchetti nuovi
7 - coda dei pacchetti in attesa di ACK
8 - coda dei pacchetti in attesa di output
9 - pacchetti rifiutati in ingresso
10 - velocita' effettiva di prelievo dei pacchetti

per le stazioni, e:

0 - traffico offerto al canale
1 - occupazione del canale
2 - throughput
3 - one-way delay
4 - rango delle collisioni

per il canale

- il periodo di campionamento in virtual slot

- il ddname del file su cui debbono venir memorizzati i valori si noti che solo i primi 7 caratteri di questo nome sono significativi e soltanto uno spazio deve separare il nome del file dal dato precedente.

11.2 FORMATO DEI RISULTATI IN USCITA

I risultati della simulazione sono memorizzati su due file di ddname 'HEADREP' e 'REPORTS' aventi entrambi lunghezza del record logico di 80 caratteri; inoltre viene prodotto un printout su stampante o su terminale o su disco a seconda delle opzioni usate a run-time.
Le procedure che producono gli output sono le seguenti:

- per i dati generali della sessione di simulazione, memorizzati sul file 'HEADREP' e riportati sul printout:

```
procedure REPHEAD;
begin
   Eject(100);
   Outtext("END-TO-END  ");
   HEADREP.Open(Blanks(80));
   HEADREP.Outtext("END-TO-END++");
   if RTDMAFLAG then
   begin
      HEADREP.Outtext("R-TDMA/");
      Outtext("R-TDMA/");
   end;
   Outtext(if SALOHAFLAG then "S-ALOHA" else "F-TDMA");
   HEADREP.Outtext(if SALOHAFLAG then "S-ALOHA " else "F-TDMA ");
   Outimage; Outimage; HEADREP.Outimage;
   HEADREP.Outtext("Simtime="); HEADREP.Outint(SIMTIME,6);
   HEADREP.Outimage;
   Outtext("Simulation time  ");
   HEADREP.Outfix(SIMTIME,6,12);
```

```
    HEADREP.Outimage;
    Outreal(SIMTIME,6,12); Outimage;Outimage;Outimage;
end*** OF PROCEDURE REPHEAD ********************;
```

* per i dati relativi al canale, memorizzati parte sul file 'HEADREP' e parte sul file 'REPORTS', e riportati sul printout:

```
procedure CHANDUMP;
begin
    if not BALANCEFLAG then Outtext("UN");
    Outtext("BALANCED TRAFFIC "); Outimage; Outimage;
    if not BALANCEFLAG then HEADREP.Outtext("un");
    HEADREP.Outtext("balanced-traffic"); HEADREP.Outimage;
    Eject(100);
    Outtext("CHANNEL "); Outimage; Outimage;
    Outtext("Speed rate (bps)  ");
    HEADREP.Outfix(NOISELEV,6,12);
    HEADREP.Outimage;
    HEADREP.Close;
    Outreal(CRATE,6,12); Outimage;
    Outtext("Virtual slot length (sec)      ");
    Outreal(LVSLOT,6,12); Outimage;
    Outtext("Average propagation delay      ");
    Outreal(PRDELAYS,6,12);
    Outtext("   (sec)"); Outimage ;
    Outtext("                               ");
    Outreal(PRDELAY,6,12);
    Outtext("   (slot)"); Outimage ;
    Outtext("Data slots per frame  ");
    Outint(LRFRAME,6); Outimage;
    Outtext("Ack window ");
    Outint(ACKWIND,6); Outimage;
    Outtext("Control slot length  ");
    Outreal(LRESVSLOT,6,12);
    Outtext("   (slot)"); Outimage ;
    Outtext("                        ");
    Outreal(LRESVPACK,6,12);
    Outtext("   (bit)"); Outimage;
    Outtext("Channel efficiency factor (max)    ");
    Outfix(1.0-((STATRES*LRESVSLOT)/(LRFRAME*UPSTATION(LEADER).
    LDATA)),6,12); Outimage; Outimage;
    Outtext("Traffic offered (pkt/slot)   "); Outimage;
    WREPORT(STATNUM,1,true); Outimage;
    Outtext("Channel occupancy  "); Outimage;
    WREPORT(STATNUM,2,true); Outimage;
    Outtext("Channel throughput (pkt/slot)  "); Outimage;
    WREPORT(STATNUM,3,true); Outimage;
    Outtext("Channel delay (slot) "); Outimage;
    WREPORT(STATNUM,4,false); Outimage;
    Outtext("Collision rank "); Outimage;
    WREPORT(STATNUM,5,false);
   !*** fill the unused part of rep matrix ***;
```

```
        WREPORT(STAT.ID,3,true);  Outimage;
        Outtext("Two-way delay of outgoing packets (slot) "); Outimage;
        WREPORT(STAT.ID,4,true);  Outimage;
        Outtext("Retransmission queue length    "); Outimage;
        Outtext("   conditional probability     "); Outimage;
        WREPORT(STAT.ID,5,false); Outimage;
        Outtext("   unconditional probability     "); Outimage;
        WREPORT1(STAT.ID,5,false); Outimage;
        Outtext("New packets queue length    "); Outimage;
        Outtext("   conditional probability     "); Outimage;
        WREPORT(STAT.ID,6,false); Outimage;
        Outtext("   unconditional probability     "); Outimage;
        WREPORT1(STAT.ID,6,false); Outimage;
        Outtext("Wait-for-acknowledgment queue length    "); Outimage;
        Outtext("   conditional probability     "); Outimage;
        WREPORT(STAT.ID,7,false); Outimage;
        Outtext("   unconditional probability     "); Outimage;
        WREPORT1(STAT.ID,7,false); Outimage;
        Outtext("Output packets queue length    "); Outimage;
        Outtext("   conditional probability     "); Outimage;
        WREPORT(STAT.ID,8,false); Outimage;
        Outtext("   unconditional probability     "); Outimage;
        WREPORT1(STAT.ID,8,false); Outimage;
        Outtext("Buffer pool occupancy histogram "); Outimage;
        for I:=0 step 1 until 10 do
        begin
            Outfix(STAT.DISTBUFPAT(I),1,6);
            Outfix(STAT.DISTBUF(I)/STAT.TBUFTIME,4,8);
            Outimage;
        end;
        Eject(100);
    end***OF PROCEDURE STATDUMP***;
```

Le procedure WREPORT e WREPORT1 che effettuano il calcolo delle medie, delle varianze e le stampe dei dati ottenuti sono le seguenti:

```
procedure WREPORT(ID,TYPE,SKIP);
   short integer ID,TYPE;
   boolean SKIP;
begin
   real array AUX(1:4);
   real MAXR;
   short integer I,J;

   MAXR:=1.0&16;
   if (REPSTAT(ID,TYPE,2) le FUZZ) then
   begin
      REPSTAT(ID,TYPE,2):=MAXR;
      REPSTAT(ID,TYPE,0):=0.0;
   end
   else
      if (REPSTAT(ID,TYPE,3) lt FUZZ) and SKIP then
```

```
        begin
            REPSTAT(ID,TYPE,1):=MAXR;
            REPSTAT(ID,TYPE,3):=MAXR;
        end;
      MOM(1):=REP.REPSTAT(ID,TYPE,0);
      MOM(2):=REP.REPSTAT(ID,TYPE,1);
      MOM(4):=(REPSTAT(ID,TYPE,4)-
            ((REPSTAT(ID,TYPE,3)**2)/REPSTAT(ID,TYPE,2)))/
              REPSTAT(ID,TYPE,2);
      MOM(3):=REPSTAT(ID,TYPE,3)/REPSTAT(ID,TYPE,2);
    !** test for fuzz in expectation and variance **;
      for I:=3,4 do
        if (Abs(MOM(I)) lt FUZZ) then
            MOM(I):=0.0;
    !** print results ***************************;
      for I:=1 step 1 until 4 do
      begin
        Outtext(TESTHEAD(I));
        Outreal(MOM(I),6,12); Outimage;
        REPORTS.Outfix(MOM(I),6,12);
      end;
      REPORTS.Outimage;
end*** OP PROCEDURE WREPORT ***;

!*** calculates and writes data for the report *********;
procedure WREPORT1(ID,TYPE,SKIP);
    short integer ID,TYPE;
    boolean SKIP;
begin
    real array MOM(1:4);
    real MAXR;
    short integer I,J;

    MAXR:=1.0&16;
    if (REPSTAT(ID,TYPE,2) le FUZZ) then
    begin
        REPSTAT(ID,TYPE,0):=0.0;
        REPSTAT(ID,TYPE,2):=MAXR;
    end
    else
        if (REPSTAT(ID,TYPE,3) lt FUZZ) and SKIP then
        begin
            REPSTAT(ID,TYPE,1):=MAXR;
            REPSTAT(ID,TYPE,3):=MAXR;
        end;
    MOM(1):=REP.REPSTAT(ID,TYPE,0);
    MOM(2):=REP.REPSTAT(ID,TYPE,1);
    MOM(4):=(REPSTAT(ID,TYPE,4)-
            ((REPSTAT(ID,TYPE,3)**2)/SIMTIME))/SIMTIME;
    MOM(3):=REPSTAT(ID,TYPE,3)/SIMTIME;
  !** test for fuzz in expectation and variance **;
    for I:=3,4 do
        if (Abs(MOM(I)) lt FUZZ) then
            MOM(I):=0.0;
```

```
  !** print results ***************************;
   for I:=1 step 1 until 4 do
   begin
      Outtext(TESTHEAD(I));
      Outreal(MOM(I),6,12); Outimage;
      REPORTS.Outfix(MOM(I),6,12);
   end;
   REPORTS.Outimage;
end*** OF PROCEDURE WREPORT1 ******************;
```

la prima viene impiegata per variabili di cui interessi solo la probabilita' incondizionata mentre la seconda viene impiegata per le variabili di cui interessa sia la probabilita' incondizionata che una probabilita' condizionata. Tutte queste procedure sono interne alla class REPORT e possono venir facilmente modificate.

Sul file "HEADREP" vengono accumulate informazioni relative al tipo di run di simulazione eseguiti.

I dati accumulati sul file "REPORTS" sono nell'ordine dati relativi a:

- traffico
- occupazione
- throughput
- ritardo
- rango delle collisioni

sul canale, e:

- traffico effettivamente generato
- traffico perso in generazione
- traffico offerto al canale
- throughput
- one-way delay
- two-way delay
- lunghezza della coda dei pacchetti da ritrasmettere
- lunghezza della coda dei pacchetti nuovi
- lunghezza della coda dei pacchetti in attesa di ACK
- lunghezza della coda dei pacchetti in attesa di uscita

per ogni stazione.

I dati relativi alla lunghezza delle code sono doppi:

- il primo si riferisce al tempo per cui la coda non e' vuota, ed esprime pertanto una probabilita' condizionata
- il secondo si riferisce invece al tempo di simulazione, ed esprime cosi' una probabilita' incondizionata.

Per ognuno dei dati ricordati si hanno 4 valori:

- il valore massimo
- il valore minimo
- il valore medio
- la varianza

Il formato di questi dati e' una successione di matrici 15x4, ove ciascun dato e' in formato fisso, con 6 cifre decimali e lunghezza totale del campo pari a 12 caratteri.

Nel caso del canale si ha sempre una matrice 15x4 riempita con 0 nelle posizioni non significative.

12.0 VALIDAZIONE DEL PROGRAMMA.

Per il controllo e la validazione del programma ci si e' avvalsi di risultati relativi a misure eseguite sulla rete SATNET.

Come dati base si sono assunti quelli relativi alla relazione tra throughput e one-way delay per il protocollo F-TDMA (vedi R.6,Tabella 3.3, p.11), che si riportano di seguito:

Channel utilization	Delay 1-> 2	Delay 2-> 1
0.26	9.1	10.0
0.42	9.8	9.7
0.54	9.1	9.3
0.72	9.9	9.7
1.00	125.7	125.7

dove il ritardo e' espresso in slot di dati.

I dati in questione sono stati calcolati, come si desume dal rapporto citato, avendo quale ritardo di propagazione 240 msec, una lunghezza di trama pari a 32 slot di dati, 30 slot di dati per trama, due stazioni che si inviano traffico bilanciato, sessioni di misura lunghe 5 min ciascuna e messaggi costituiti da un solo pacchetto. L'utilizzazione del canale e' stata calcolata facendo uguale ad 1 l'utilizzazione di 15 slot per trama, cioe' la massima possibile nelle condizioni indicate.

L'ultimo valore dipende chiaramente dalla lunghezza dei buffer nella stazione essendo il canale in condizioni al limite dell'instabilita'. Per questo motivo il dato in questione e' stato sostituito nel programma di simulazione con un valore di utilizzazione pari a 0.975.

Con questi dati il programma di simulazione ha fornito i risultati riportati nella tabella seguente:

Channel utilization	Delay 1-> 2	Delay 2-> 1
0.26	9.1	10.1
0.42	10.2	10.2
0.54	10.3	10.2
0.72	10.4	10.4

Come si puo' vedere gli scostamenti sono molto bassi. Fa' eccezione, come e' da prevedere, l'ultimo valore. In queste condizioni il canale e' in situazione di instabilita', pur con velocita' di generazione costanti si ha una varianza tanto elevata nei ritardi da rendere non significativo il valor medio calcolato. Poiche', inoltre, mancano sul rapporto citato i valori della

varianza, si e' preferito omettere il confronto per l'ultimo valore.

Si noti che nel programma di simulazione il rumore e' rigorosamente nullo e la velocita' di generazione del traffico costante, cio' spiega la maggiore uniformita' dei risultati per gli altri valori.

BIBLIOGRAFIA

R.1 N. ABRAMSON, "Packet Switching with Satellites", Proc. AFIPS Conf., vol. 42, June 1973, pp. 695-702.

R.2 L. KLEINROCK, S. LAM, "Packet Switching in a Slotted Satellite Channel", Proc. AFIPS Conf., vol. 42, June 1973, pp. 703-710.

R.3 L. ROBERTS, "Dynamic Allocation of Satellite Capacity through Packet Reservation", Proc. AFIPS Conf., vol. 42, June 1973, pp. 711-716.

R.4 R. BINDER, "A Dynamic Packet-Switching System for Satellite Broadcast Channels", ICC 1975, San Francisco, June 1975, vol. 3, pp. 41-1 41-5.

R.5 R. BINDER, "Current SIMP Protocols", PSPNN N.11, Bolt Beranek and Newman, April 1976.

R.6 P. SPILLING, "F-TDMA Measurements performed on the Satellite Channel", PSPNN N.57, Norwegian Defence Research Establishment, Feb. 1977.

R.7 M. GERLA, L. KLEINROCK, L. NELSON, "Packet Satellite Multiple Access: Models and Measurements", Proc. of the National Telecommunications Conference, Los Angeles, Dec. 1977, pp. 12.2-1 12.2-8.

R.8 P. SPILLING, "R-TDMA Measurements performed on the Satellite Channel", PSPNN N.93, Norwegian Defence Research Establishment, Feb. 1978.

R.9 R. BINDER, E.V. HOVERSTEN, J.M. JACOBS, "General Purpose Packet Satellite Networks", Proc. IEEE, vol. 66, Nov. 1978, pp. 1448-1467.

APPENDICE A : UN ESEMPIO DI SIMULAZIONE

A.1 DATI DI INGRESSO

Il file di ingresso dati con ddname DATA e' il seguente:

```
64.0E+3 0.240 0.006
1 0 32 05 64 001 300 03 060.0 1.00
005 1500 0.0E-1 0.0E-2 0.00 0.00 200 1
1  7.60E-1 0.00E-1  1 1  0 0 0
7.0E-1
005 1500 0.0E-2 0.0E-2 0.00 0.00 200 1
1  1.15E-1 0.00E-1  1 1  0 0 0
7.0E-1
005 1500 0.0E-2 0.0E-2 0.00 0.00 200 1
1  1.15E-1 0.00E-1  1 1  0 0 0
7.0E-1
```

A.2 USCITA SU STAMPANTE

Questo invece e' il printout:

```
 END-TO-END  R-TDMA/F-TDMA

 Simulation time    6.00000E+01    (sec)
                    1.00000E+04    (slot)


 Station      1
 Interarrival time distribution: fixed rate
 Imposed messages generation rate (pkt/slot)  7.60000E-01


 Station      1
 Imposed messages sink rate (pkt/slot) 7.00000E-01

 Station      2
 Interarrival time distribution: fixed rate
 Imposed messages generation rate (pkt/slot)  1.15000E-01


 Station      2
 Imposed messages sink rate (pkt/slot) 7.00000E-01
```

```
Station      3
Interarrival time distribution: fixed rate
Imposed messages generation rate (pkt/slot)   1.150000E-01

Station      3
Imposed messages sink rate (pkt/slot) 7.000000E-01

UNBALANCED TRAFFIC
```

CHANNEL

Speed rate (bps) 6.40000E+04
Virtual slot length (sec) 6.00000E-03
Average propagation delay 2.40000E-01 (sec)
4.00000E+01 (slot)
Data slots per frame 32
Ack window 64

Control slot length 1.00000E+00 (slot)
3.00000E+02 (bit)
Channel efficency factor (max) 0.968750

Traffic offered (pkt/slot)

minimum	0.00000E+00
maximum	1.00000E+00
expectation	9.64918E-01
variance	1.14244E-02

Channel occupancy

minimum	0.00000E+00
maximum	1.00000E+00
expectation	9.64918E-01
variance	1.14244E-02

Channel throughput (pkt/slot)

minimum	1.66667E-01
maximum	1.00000E+00
expectation	9.71300E-01
variance	1.31864E-02

Channel delay (slot)

minimum	4.10000E+01
maximum	4.72105E+02
expectation	2.96427E+02
variance	9.70139E+03

Collision rank

minimum	0.00000E+00
maximum	0.00000E+00
expectation	0.00000E+00
variance	0.00000E+00

STATIONS

Station number 1

Rgate 0.00000E+00
Ngate 0.00000E+00

```
Data slot length          5.00000000    (slot)
                       1500.00000000    (bit)

Up link noise factor          0.00000E+00
Down link noise factor        0.00000E+00

Buffer pool allocation (pkt)       200

Effective messages generation rate (pkt/slot)
        minimum          7.60000E-01
        maximum          7.60000E-01
        expectation      7.60000E-01
        variance         0.00000E+00

Generated messages loss rate (pkt/slot)
        minimum          0.00000E+00
        maximum          0.00000E+00
        expectation      0.00000E+00
        variance         0.00000E+00

Effective messages sink rate (pkt/slot)
        minimum          2.94118E-02
        maximum          7.00000E-01
        expectation      1.06862E-01
        variance         2.56610E-02

Traffic of outgoing packets (pkt/slot)
        minimum          2.00000E-01
        maximum          1.00000E+00
        expectation      7.29000E-01
        variance         1.01534E-01

Throughput of outgoing packets (pkt/slot)
        minimum          2.00000E-01
        maximum          1.00000E+00
        expectation      7.28779E-01
        variance         1.01536E-01

One-way delay of outgoing packets (slot)
        minimum          5.84211E+01
        maximum          4.72105E+02
        expectation      3.32560E+02
        variance         5.95618E+03

Two-way delay of outgoing packets (slot)
        minimum          1.60947E+02
        maximum          6.44895E+02
        expectation      4.24879E+02
        variance         7.46871E+03

Retransmission queue length
   conditional probability
        minimum          0.00000E+00
```

```
        maximum        0.00000E+00
        expectation    0.00000E+00
        variance       0.00000E+00

    unconditional probability
        minimum        0.00000E+00
        maximum        0.00000E+00
        expectation    0.00000E+00
        variance       0.00000E+00

New packets queue length
    conditional probability
        minimum        0.00000E+00
        maximum        6.50000E+01
        expectation    4.38072E+01
        variance       1.32218E+02

    unconditional probability
        minimum        0.00000E+00
        maximum        6.50000E+01
        expectation    4.38072E+01
        variance       1.32218E+02

Wait-for-acknowledgment queue length
    conditional probability
        minimum        0.00000E+00
        maximum        3.40000E+01
        expectation    1.86788E+01
        variance       2.57007E+01

    unconditional probability
        minimum        0.00000E+00
        maximum        3.40000E+01
        expectation    1.86695E+01
        variance       2.58623E+01

Output packets queue length
    conditional probability
        minimum        0.00000E+00
        maximum        2.00000E+00
        expectation    1.01102E+00
        variance       1.09015E-02

    unconditional probability
        minimum        0.00000E+00
        maximum        2.00000E+00
        expectation    1.00236E+00
        variance       1.94944E-02

Buffer pool occupancy histogram
    0.0   0.0007
    0.1   0.0132
    0.2   0.0320
    0.3   0.3834
```

```
One-way delay of outgoing packets (slot)
        minimum         5.08694E+01
        maximum         2.96954E+02
        expectation     2.26049E+02
        variance        3.60817E+03

Two-way delay of outgoing packets (slot)
        minimum         1.00869E+02
        maximum         3.71042E+02
        expectation     2.85167E+02
        variance        4.26932E+03

Retransmission queue length
   conditional probability
        minimum         0.00000E+00
        maximum         0.00000E+00
        expectation     0.00000E+00
        variance        0.00000E+00

   unconditional probability
        minimum         0.00000E+00
        maximum         0.00000E+00
        expectation     0.00000E+00
        variance        0.00000E+00

New packets queue length
   conditional probability
        minimum         0.00000E+00
        maximum         6.00000E+00
        expectation     3.62630E+00
        variance        1.83559E+00

   unconditional probability
        minimum         0.00000E+00
        maximum         6.00000E+00
        expectation     3.62086E+00
        variance        1.85253E+00

Wait-for-acknowledgment queue length
   conditional probability
        minimum         0.00000E+00
        maximum         4.00000E+00
        expectation     1.35413E+00
        variance        5.45314E-01

   unconditional probability
        minimum         0.00000E+00
        maximum         4.00000E+00
        expectation     1.33450E+00
        variance        5.63610E-01

Output packets queue length
   conditional probability
        minimum         0.00000E+00
```

```
        maximum        4.00000E+00
        expectation    1.13590E+00
        variance       1.32718E-01

    unconditional probability
        minimum        0.00000E+00
        maximum        4.00000E+00
        expectation    1.13557E+00
        variance       1.33049E-01

Buffer pool occupancy histogram
    0.0   0.0043
    0.1   0.9954
    0.2   0.0000
    0.3   0.0000
    0.4   0.0000
    0.5   0.0000
    0.6   0.0000
    0.7   0.0000
    0.8   0.0000
    0.9   0.0000
    1.0   0.0000


Station number     3

Rgate    0.00000E+00
Ngate    0.00000E+00

Data slot length        5.00000000    (slot)
                     1500.00000000    (bit)

Up link noise factor        0.00000E+00
Down link noise factor      0.00000E+00

Buffer pool allocation (pkt)       200

Effective messages generation rate (pkt/slot)
        minimum        1.15000E-01
        maximum        1.15000E-01
        expectation    1.15000E-01
        variance       0.00000E+00

Generated messages loss rate (pkt/slot)
        minimum        0.00000E+00
        maximum        0.00000E+00
        expectation    0.00000E+00
        variance       0.00000E+00

Effective messages sink rate (pkt/slot)
        minimum        6.48148E-02
        maximum        7.00000E-01
```

```
        expectation   4.14970E-01
        variance      4.59138E-02

Traffic of outgoing packets (pkt/slot)
        minimum       3.22581E-02
        maximum       5.00000E-01
        expectation   1.12056E-01
        variance      1.55837E-02

Throughput of outgoing packets (pkt/slot)
        minimum       3.22581E-02
        maximum       5.00000E-01
        expectation   1.13026E-01
        variance      1.57167E-02

One-way delay of outgoing packets (slot)
        minimum       4.52173E+01
        maximum       3.01954E+02
        expectation   2.24458E+02
        variance      4.19410E+03

Two-way delay of outgoing packets (slot)
        minimum       9.52173E+01
        maximum       4.39172E+02
        expectation   2.88571E+02
        variance      4.82889E+03

Retransmission queue length
   conditional probability
        minimum       0.00000E+00
        maximum       0.00000E+00
        expectation   0.00000E+00
        variance      0.00000E+00

   unconditional probability
        minimum       0.00000E+00
        maximum       0.00000E+00
        expectation   0.00000E+00
        variance      0.00000E+00

New packets queue length
   conditional probability
        minimum       0.00000E+00
        maximum       6.00000E+00
        expectation   3.60157E+00
        variance      2.06011E+00

   unconditional probability
        minimum       0.00000E+00
        maximum       6.00000E+00
        expectation   3.59797E+00
        variance      2.07101E+00
```

```
Wait-for-acknowledgment queue length
   conditional probability
        minimum       0.00000E+00
        maximum       5.00000E+00
        expectation   1.43408E+00
        variance      6.29021E-01

   unconditional probability
        minimum       0.00000E+00
        maximum       5.00000E+00
        expectation   1.41400E+00
        variance      6.48604E-01

Output packets queue length
   conditional probability
        minimum       0.00000E+00
        maximum       4.00000E+00
        expectation   1.13684E+00
        variance      1.38126E-01

   unconditional probability
        minimum       0.00000E+00
        maximum       4.00000E+00
        expectation   1.13636E+00
        variance      1.38621E-01

Buffer pool occupancy histogram
    0.0   0.0043
    0.1   0.9954
    0.2   0.0000
    0.3   0.0000
    0.4   0.0000
    0.5   0.0000
    0.6   0.0000
    0.7   0.0000
    0.8   0.0000
    0.9   0.0000
    1.0   0.0000
```

A.3 USCITA SU DISCO

Questo e' il file di ddname "HEADREP":

```
END-TO-END++R-TDMA/F-TDMA
Simtime= 10000
   60.000000
unbalanced-traffic
    0.000000
```

Questo e' il file di ddname "REPORTS":

0.000000	1.000000	0.964918	0.011424
0.000000	1.000000	0.964918	0.011424
0.166667	1.000000	0.971300	0.013186
41.000000	472.105469	296.426758	9701.386719
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.760000	0.760000	0.760000	0.000000
0.000000	0.000000	0.000000	0.000000
0.029412	0.700000	0.106862	0.025661
0.200000	1.000000	0.729000	0.101534
0.200000	1.000000	0.728779	0.101536
58.421051	472.105469	332.560303	5956.179687
160.947357	644.895020	424.878662	7468.707031
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	65.000000	43.807190	132.218079
0.000000	65.000000	43.807175	132.218170
0.000000	34.000000	18.678833	25.700745
0.000000	34.000000	18.669495	25.862259
0.000000	2.000000	1.011023	0.010902
0.000000	2.000000	1.002357	0.019494
0.115000	0.115000	0.115000	0.000000
0.000000	0.000000	0.000000	0.000000
0.048951	0.700000	0.426409	0.045164
0.035714	0.333333	0.112112	0.015512
0.035714	0.333333	0.113083	0.015644
50.869385	296.954346	226.049332	3608.165039
100.869385	371.041504	285.167236	4269.324219
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000
0.000000	6.000000	3.626303	1.835586
0.000000	6.000000	3.620863	1.852528
0.000000	4.000000	1.354135	0.545314
0.000000	4.000000	1.334499	0.563610
0.000000	4.000000	1.135896	0.132718
0.000000	4.000000	1.135571	0.133049
0.115000	0.115000	0.115000	0.000000
0.000000	0.000000	0.000000	0.000000
0.064815	0.700000	0.414970	0.045914
0.032258	0.500000	0.112056	0.015584
0.032258	0.500000	0.113026	0.015717
45.217285	301.954346	224.458145	4194.101562
95.217285	439.171875	288.570557	4828.886719
0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	0.000000

0.000000	6.000000	3.601573	2.060115
0.000000	6.000000	3.597972	2.071013
0.000000	5.000000	1.434076	0.629021
0.000000	5.000000	1.414000	0.648604
0.000000	4.000000	1.136844	0.138126