# Simulazione della riconfigurazione delle tabelle di routing in una rete terrestre

## Algoritmi di RPCNET

Renzo Beltrame
Enrico Gregori

Nota informativa C 81-5

Pisa
Giugno 1981

INDICE

## 1.0 SIMULAZIONE DEL ROUTING DI RPCNET.

L'algoritmo di routing di RPCNET calcola il migliore percorso per raggiungere un nodo tenendo conto solo della topologia della rete.
Ne discende che il routing risulta di tipo deterministico poiche' il percorso migliore diventa, in tali condizioni, il cammino piu' breve sul grafo che descrive la topologia attuale della rete e nel caso di cammini di egual lunghezza ne viene scelto uno una volta per tutte.
Inoltre al fine di stabilire o aggiornare le tabelle per il routing ogni nodo deve conoscere inizialmente solo i nodo con cui e' direttamente collegato.

La struttura delle tabelle e' la seguente: Il protocollo che realizza il suddetto algoritmo di routing prende decisioni considerando sempre migliore il percorso piu' breve. Da cio' deriva che per realizzare il routing dell'intera rete ogni nodo deve conoscere solo i suoi vicini e le distanze, cioe' il numero di passaggi da nodo a nodo, degli altri nodi della rete dai vicini considerati;il suddetto protocollo utilizza due tabelle: la DISTANCE TABLE e la ROUTING TABLE. La prima ha tante righe quanti sono i nodi della rete ed una colonna per ogni vicino; la seconda ha tante righe quanti sono i nodi della rete e due colonne, una contenente il numero minimo di cammini necessari per raggiungere il nodo, l'altra indicante il primo nodo del percorso, cioe' il vicino a cui bisogna mandare il pacchetto.
Ai fini della simulazione non e' stato ritenuto importante utilizzare procedure di aggiornamento delle tabelle esattamente uguali a quelle della rete RPCNET, ma e' stato realizzato un meccanismo di aggiornamento utilizzante informazioni ridondanti per rendere piu' breve e' veloce il programma.

## 1.1 MECCANISMO DI UPDATE.

Ogni volta che un nodo (nodo I) scopre una variazione nella topologia della rete manda la sua routing table aggiornata ai suoi vicini.
Il vicino J inizialmente la colonna J e la riga j della tabella ricevuta, poi incrementa di uno tutti i componenti della tabella diversi da zero.
A questo punto la tabella modificata viene utilizzata per fare l'update. Per ogni riga della tabella arrivata viene memorizzato il valore minimo diverso da zero e confrontato col contenuto della corrispondente riga, colonna I della vecchia routing table. Se il

---

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

contenuta della suddetta componente, vecchia routing_table, e' uguale a zero si effettua la sostituzione, altrimenti si inserisce il valore minimo tra i due.
Se un nodo ha una distanza uguale al massimo numero di nodi della rete, e' automaticamente considerato irraggiungibile.
Se e' variata almeno una componente della routing table, la tabella aggiornata viene inviata a tutti i suoi vicini.

## 1.2 STRUTTURA DEL PROGRAMMA.

La rete e' stata simulata mediante un insieme di classi tra lorc collegate. Quindi per cgni nodo e' stato creata una PRCCESS CLASS NODE che contiene al suo interno le procedure necessarie per:

- l'aggiornamento delle tabelle per il routing
- la costruzione dei pacchetti contenenti le infcrmazioni per l'aggicrnamento
- l'invio dei pacchetti.

La parte computazionale della suddetta classe e' un blocco ripetutc in cui vegono analizzate le informazioni ricevute dagli altri nodi della rete; se le suddette informazioni portano ad una variazione delle tabelle di routing, vengono aggicrnate queste tabelle e viene inviata la DISTANCE_TABLE ai nodi vicini, altrimenti si considera terminata la fase di aggiornarneto. Parte computazionale

```
while ACTIVE do
begin
   L_PACK:-PENDING_REQ.First;
   L_PACK.Out;
   if UPDATE(L_PACK) then
   begin
      for I:=1 step 1 until N_NODE do
         if (NEIGB_NODE(I) =/= none) then
            SEND(NEW_PACK(N_NEIGB,ID),NEIGB_NODE(I));
   end;
   if not PENDING_REQ.Empty then
      reactivate Current at(PENDING_REQ.First gua PACK.
                                 GEN_TIME + LATENCY_TIME)
   else
      Passivate;
end;
```

### 1.2.1 struttura della classe NODE.

```
Process class NODE(ID,N_NEIGB,LATENCY_TIME);
   short integer ID,N_NEIGB;
   real LATENCY_TIME;
begin
   ref(NODE) array NEIGB_NODE(1:N_NODE);
   short integer array CONN_TAB(1:N_NODE+1,1:N_NEIGB);
   short integer array ROUT_TAB(1:N_NODE,1:2);
   ref(Head) PENDING_REQ;
   ref(PACK) L_PACK;
   boolean ACTIVE;
   short integer I;

   procedure SEND(C_PACK,C_NODE);
      ref(PACK) C_PACK;
      ref(NODE) C_NODE;
   begin..................... end;

   boolean procedure UPDATE(C_PACK);
      ref(PACK) C_PACK;
   begin..................... end;

   ref(PACK) procedure NEW_PACK(NEIGB,NODE_NUM);
      short integer NEIGB,NODE_NUM;
   begin..................... end;

   procedure START_UPDATE;
   begin..................... end;

   procedure INITIALIZE;
   begin..................... end;

   ...........................

end*** OF CLASS NODE **********************************************;
```

### 1.2.2 Modalita' di scambio dei pacchetti.

Per effettuare lo scambio dei pacchetti contenenti le informazioni di update ogni nodo dispone di una coda di ricezione ed esiste la possibilta' di costruire dei pacchetti da inserire nella suddetta coda.

A tal fine e' stata disegnata un LINK CLASS PACK che contiene le seguenti informazioni:

- tabella modificata

- identificatore del nodo che ha inviato la suddetta tabella.

```
Link class PACK(NEIGB);
   short integer NEIGB;
begin
   short integer array UPD_TAB(1:N_NODE+1,1:NEIGE);
   real GEN_TIME;
   short integer SENDING_NODE;

end*** OF CLASS PACK ***********************************************;
```

## 2.0 FORMATO DEI DATI DA FORNIRE AL PROGRAMMA

I dati da fornire al programma vanno posti su un file avente come ddname "DATI", lunghezza del record logico di 80 caratteri e per il resto nel formato libero con cui se li attende il SIMULA 67.
I dati riguardano:

- la topologia della rete
- il ritardo su ciascuno dei link

### 2.1 TOPOLOGIA DELLA RETE

La topologia della rete e' descritta tramite una matrice N x N, dove N e' il numero dei nodi, in cui ogni elemento di indici (i,k) e' 1 se il nodo i-esimo e' direttamente connesso con il nodo k-esimo, 0 altrimenti.

Il programma si aspetta una prima scheda in cui viene fornito il numero N dei nodi ed N schede che descrivono il modo in cui il relativo nodo e' connesso con tutti gli altri.

Questo e' un esempio per il caso di una rete con 9 nodi:

```
9
0 1 0 0 0 0 0 0 1
1 0 1 0 0 0 1 0 0
0 1 0 1 1 1 1 0 1
0 0 1 0 0 0 0 1 0
0 0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0 0
0 1 1 0 0 0 0 1 0
0 0 0 1 0 0 1 0 0
1 0 1 0 0 0 0 0 0
```

### 2.2 RITARDO SUI LINK

Il ritardo sui vari link e' fornito su M schede successive, una per ogni link, ed e' espresso in sec. Va da se' che M e' pari alla somma elemento per elemento della matrice che descrive la topologia di rete.
I dati che seguono si riferiscono alla topologia descritta

BIBLIOGRAFIA

R.1 F. CANESCHI, "A Computer Network: structure and protocols of RPCNET", IIASA PP-78-12, December 1978.