SAPIENZA Università di Roma
Facoltà di Ingegneria

Tesina di Metodi formali nell'ingegneria del software

Autori: **Piacentini Vanda - Rocco Germano**

Anno accademico 2006/2007

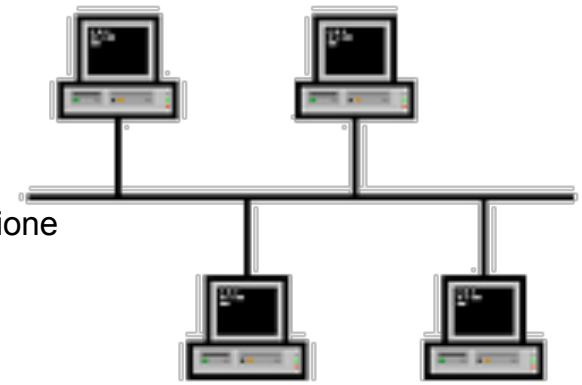
MODELLAZIONE DEL SISTEMA DI COMUNICAZIONE DELLA RETE ETHERNET: IL PROTOCOLLO CSMA/CD E VERIFICA DELLE PROPRIETA' TEMPORALI ATTRAVERSO SPIN E LTL

Scopo della tesina

- Studio della rete Ethernet
- Studio del protocollo di comunicazione CSMA/CD
- Rappresentazione del sistema in diagramma Stati & Transizioni
- Modellazione del sistema tramite SPIN
- Studio di proprietà temporali caratterizzanti il sistema attraverso LTL

Il sistema fisico

- Denominazione 10base-T
- Stazioni (peer)
- Cavo coassiale con terminatori
- Collegamento stazioni – cavo tramite Transceiver
- Funzionamento ethernet
  - Rete broadcast
    - il peer inserisce l'indirizzo di destinazione nel messaggio e lo spedisce a tutti i peer connessi alla rete
    - ogni peer della rete riceve il messaggio e confronta l'indirizzo di destinazione con il proprio
      - se sono uguali trattiene il messaggio
      - altrimenti lo scarta
  - Stazioni indipendenti
    - non esiste nessun "peer master" che gestisce la comunicazione



Il protocollo CSMA/CD

Assunzioni

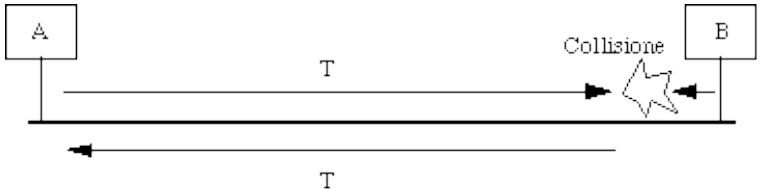
Istante di trasmissione $t_0 = 0$

Tempo di trasmissione ℓ

Tempo di propagazione T = $\frac{|A.posizione - B.posizione|}{V}$ $\quad V = \frac{c}{\sqrt{\varepsilon\mu}}$

Funzionamento

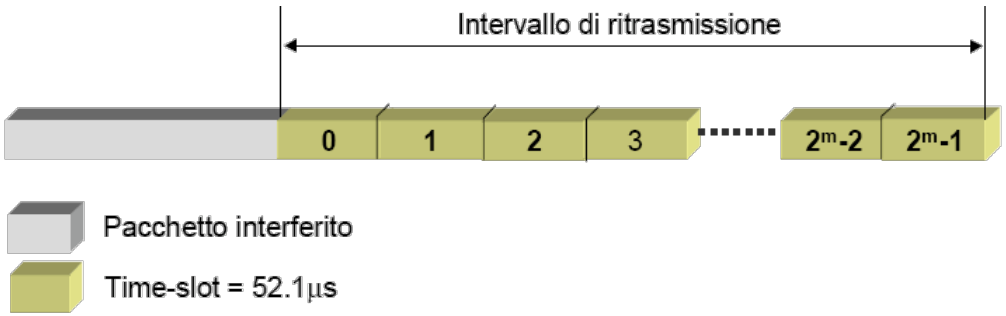
1. A ascolta il canale
   - libero → spedisce messaggio
   - occupato → attende un tempo casuale prima di trasmettere
3. A monitora la rete
   - nessun cambiamento di energia in ℓ+T → frame spedito
   - cambiamento di energia in ℓ+T → collisione
5. A interrompe la trasmissione e invia il segnale di jam
7. A esegue l'algoritmo di back-off esponenziale
   - A attende 2T * t prima di ritrasmettere, dove t ϵ {0; 2^m -1}



Algoritmo di Back-off esponenziale

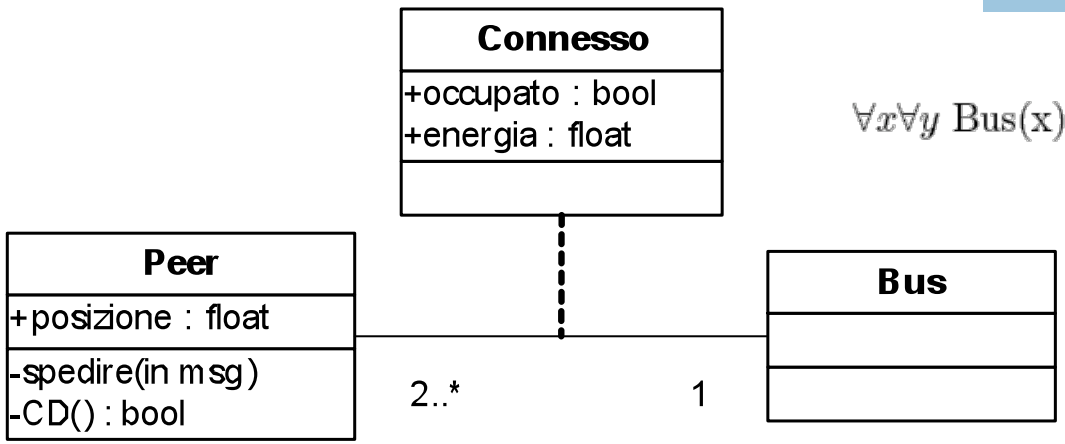
Funzionamento

- Suddivisione del tempo in 2^m slot di 52,1 μs (512 bit in un time slot)



- Scelta di **m** in base al numero **n** di collisioni:
  - se n ≤10 si pone m = n;
  - se 10<n<16 si pone m = 10;
  - dopo 16 tentativi senza successo il pacchetto viene eliminato
- Ogni processo sceglie casualmente lo slot tra 0 e 2^m-1 da cui ritrasmettere

Diagramma UML delle classi



$$\forall x \forall y \ \mathrm{Bus(x)} \wedge \mathrm{Bus(y)} \Longrightarrow \mathrm{x} = \mathrm{y}$$

Associazioni:

- Connesso

  Proprietà: *occupato, energia*

Due componenti:

- Peer
  - Proprietà: *posizione*
  - Metodi: *spedire(msg), CD()*
- Bus

Diagramma UML Stati & Transizioni - Peer

Peer

- rappresenta il processo che invia il messaggio

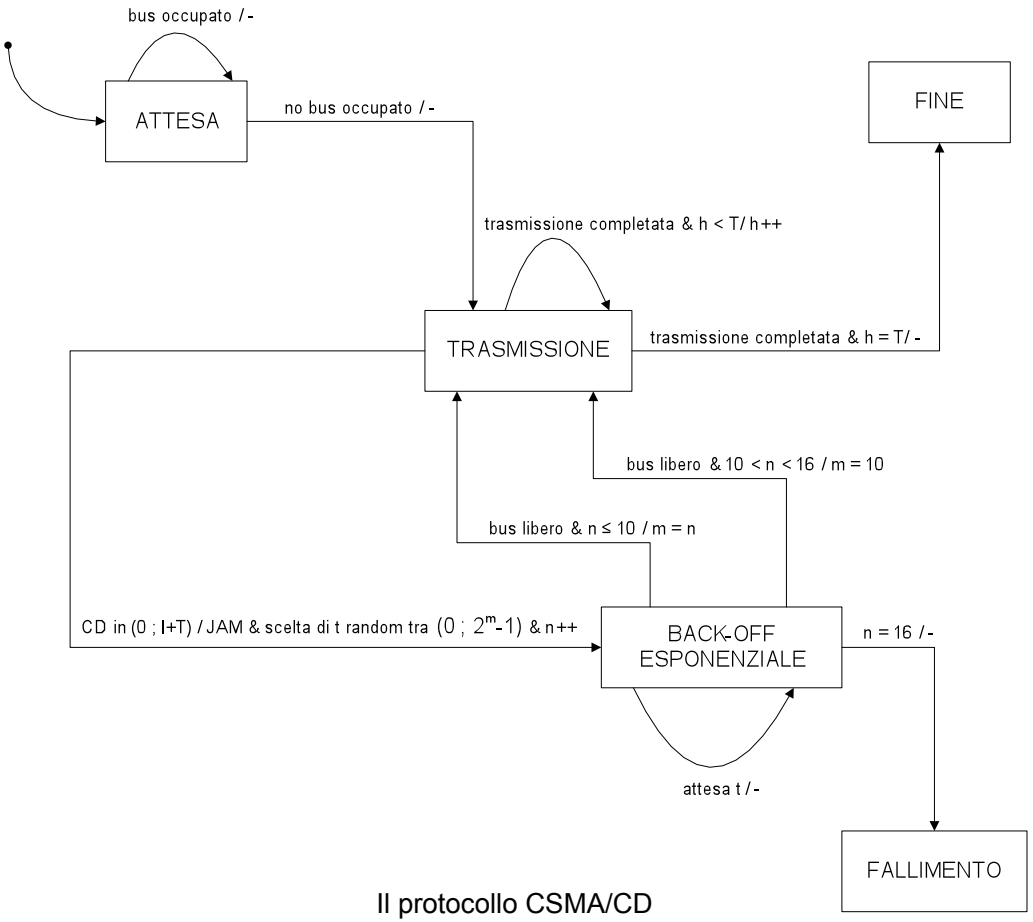
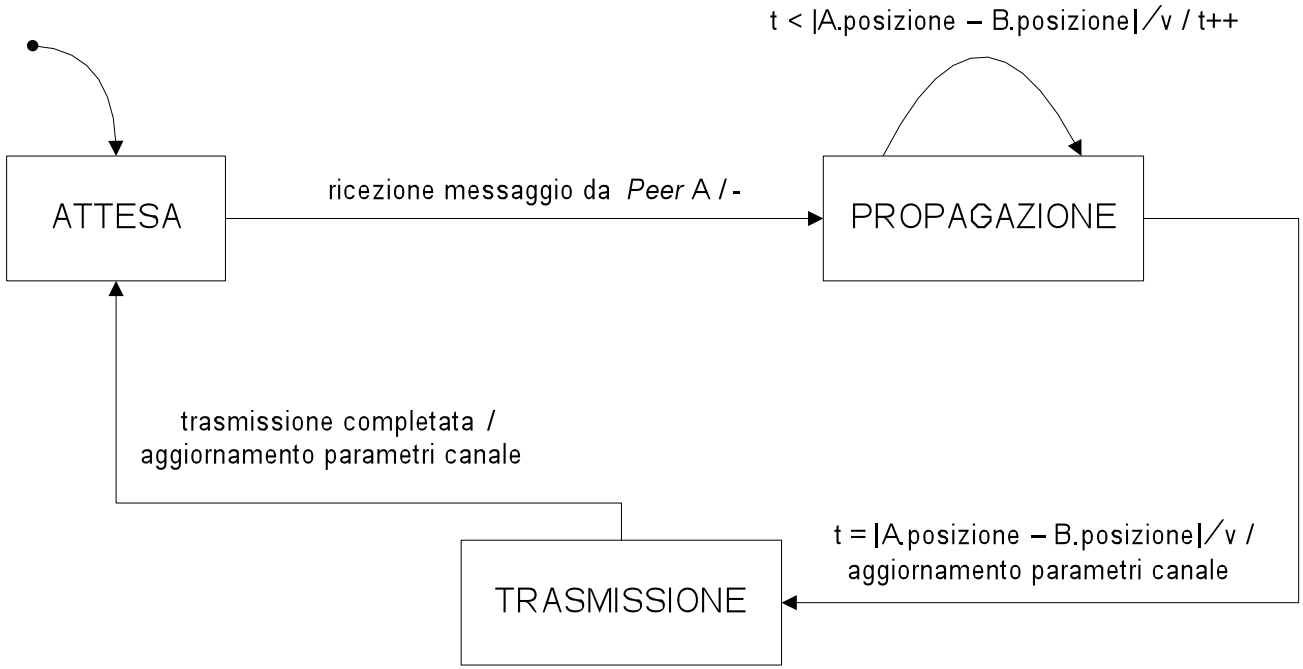


Diagramma UML Stati & Transizioni - Bus

Bus

- sistema multithread
- per ogni *Peer* connesso al *Bus* e per ogni nuova trasmissione, viene istanziato un nuovo thread
- n processi, k trasmissioni → #thread = k(n-1)



SPIN : Studio del modello (1/7)

Vantaggi:

- Sistema descritto mediante stati e transizioni
- Proprietà esprimibili in logica temporale lineare
- Processi che comunicano mediante lo scambio di messaggi

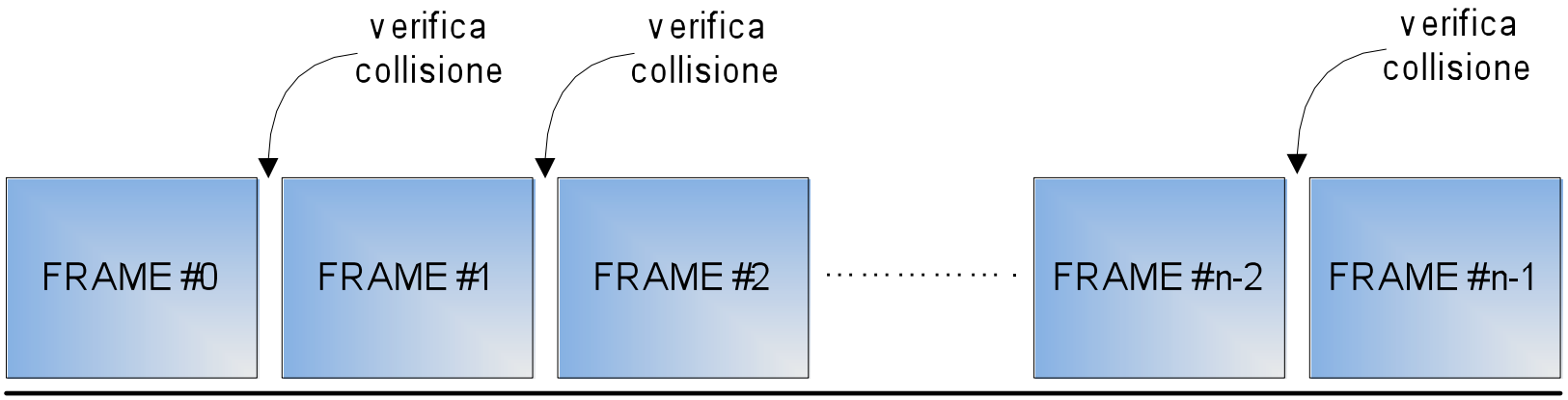
SPIN : Studio del modello (2/7)

Svantaggi:

- Trasmissione VS messaggio
- Assenza Randomize()
- Assenza Timer()

SPIN : Studio del modello (3/7)

Trasmissione intesa come una raffica di messaggi



SPIN : Studio del modello (4/7)

Funzione random

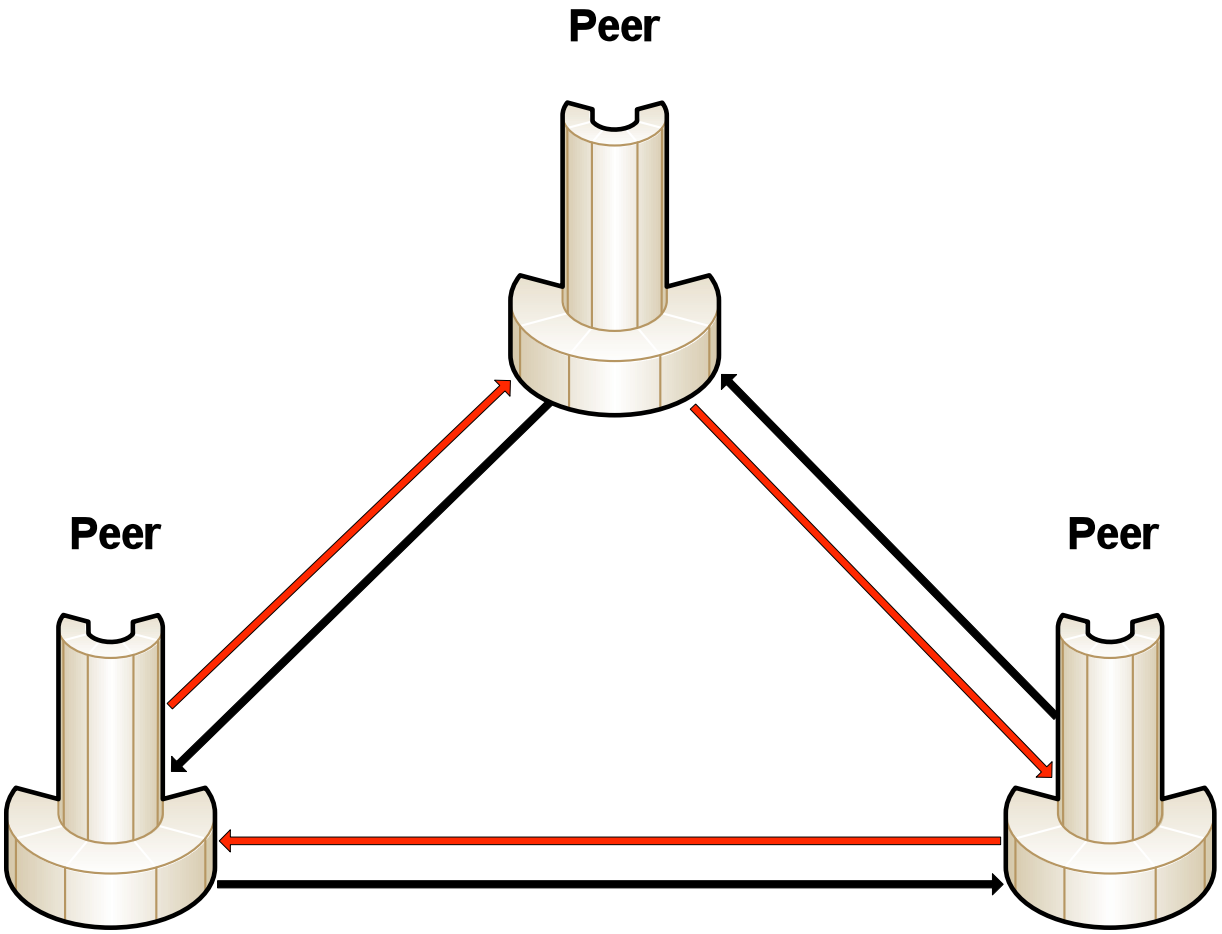
```
RANDOM:
   do
     :: num = 1; goto NEXT;
        :: num = 2; goto NEXT;
     :: num = 3; goto NEXT;
     :: num = 4; goto NEXT:
   od;
```

SPIN : Studio del modello (5/7)

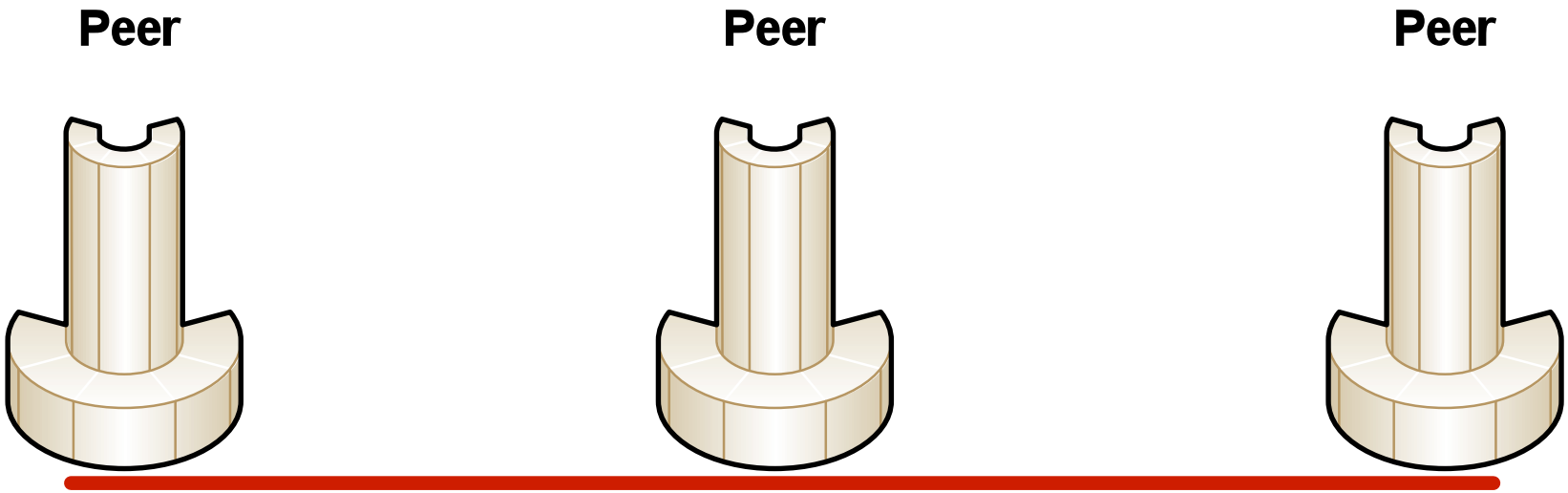
Funzione di attesa

```
WAIT:
   do
     :: nr < slot -> timeout; nr = nr + 1;
     :: nr == slot -> nr = 0; goto NEXT;
   od;
```

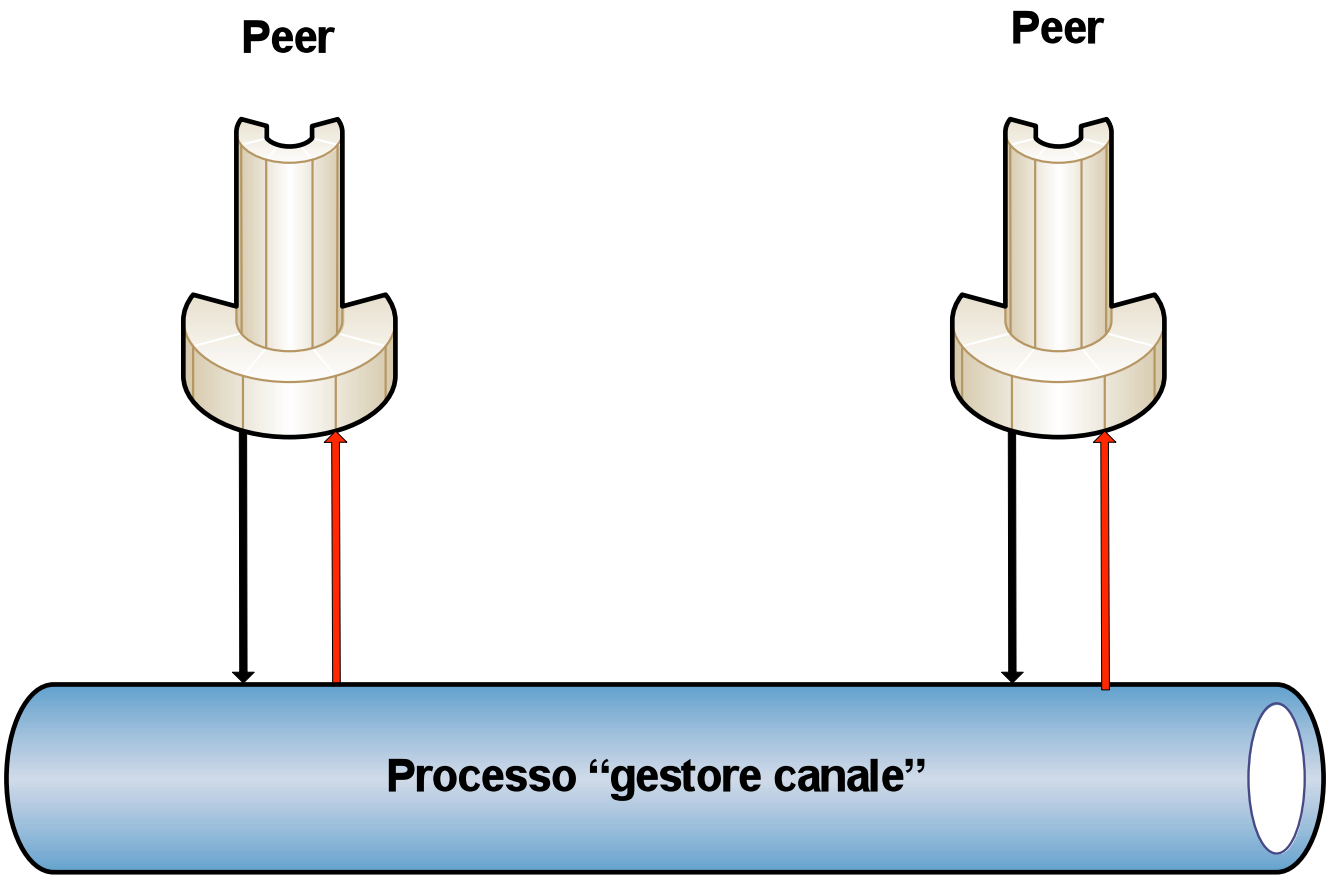
SPIN : Studio del modello (6/7)



SPIN : Studio del modello (7/7)



SPIN : Descrizione del modello (2/2)



SPIN : Realizzazione del CSMA/CD

- Assenza puntatori -> processi comunicano tramite canali o variabili globali
- 2 canali aggiuntivi per ogni processo: *state* e *collision*

SPIN : Realizzazione del CSMA/CD – Carrier Sense

Si verifica la dimensione del canale state

```
CARRIER_SENSE:
   do
     :: len(state) == 0 -> goto TRANSMISSION;
     :: len(state) != 0 -> break;
   od;
```

SPIN : Realizzazione del CSMA/CD – Collision Detection

Accesso al canale *collision*

```
CD:
   do
      :: if
         :: len(collision) == 0 -> goto TRANSMISSION;
         :: else -> collision?_,_; jc!JAM; ind = 0;
           goto EXP_BACKOFF;
         fi;
  od;
```

SPIN : Realizzazione del Backoff esponenziale (1/2)

- *Round Trip Time:* differenze
- Un messaggio impiega n loop per attraversare n posizioni
- Ottimizzazione
- No generazione dinamica di uno stato
- *Round_x* composto da $2^{indice} \times RTT$ e

$\text{indice} \in [0\ ;\ \text{x}]$

SPIN : Realizzazione del Backoff esponenziale (1/2)

Esempio per *Round_7* e RTT/2 = 5

```
ROUND_7:
   do
     :: slot = 10 -> goto BO_WAIT;
     :: slot = 20 -> goto BO_WAIT;
     :: slot = 40 -> goto BO_WAIT;
     :: slot = 80 -> goto BO_WAIT;
     :: slot = 160 -> goto BO_WAIT;
     :: slot = 320 -> goto BO_WAIT;
     :: slot = 640 -> goto BO_WAIT;
     :: slot = 1280 -> goto BO_WAIT;
   od;
```

SPIN : Implementazione

Peer:

```
proctype peer(chan out, in, jc, state, collision) {
...
}
```

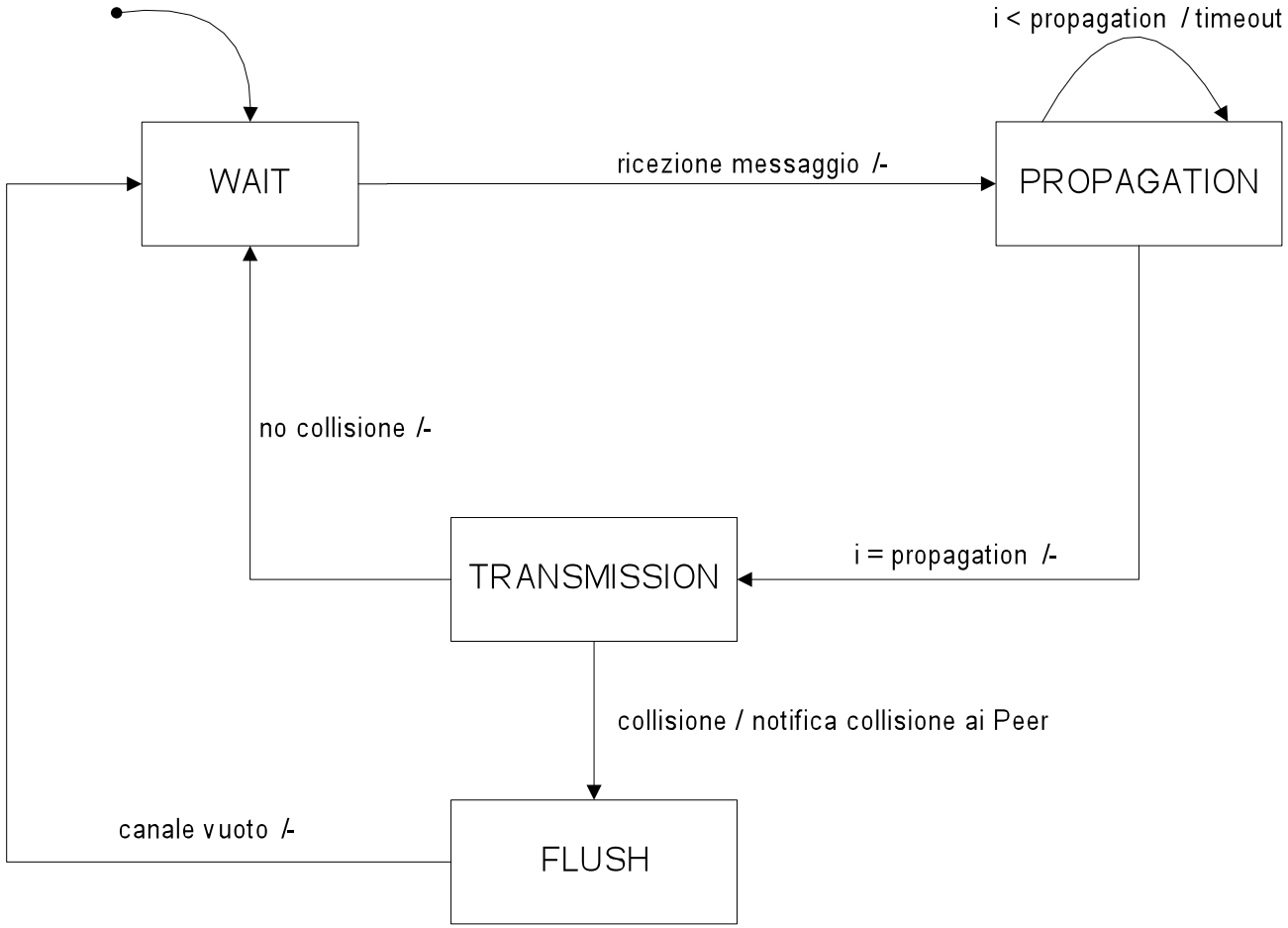
Bus:

```
proctype bus(chan in1, out1, in2, out2,
   state1,collision1, state2, collision2; int
   propagation) {
...
}
```

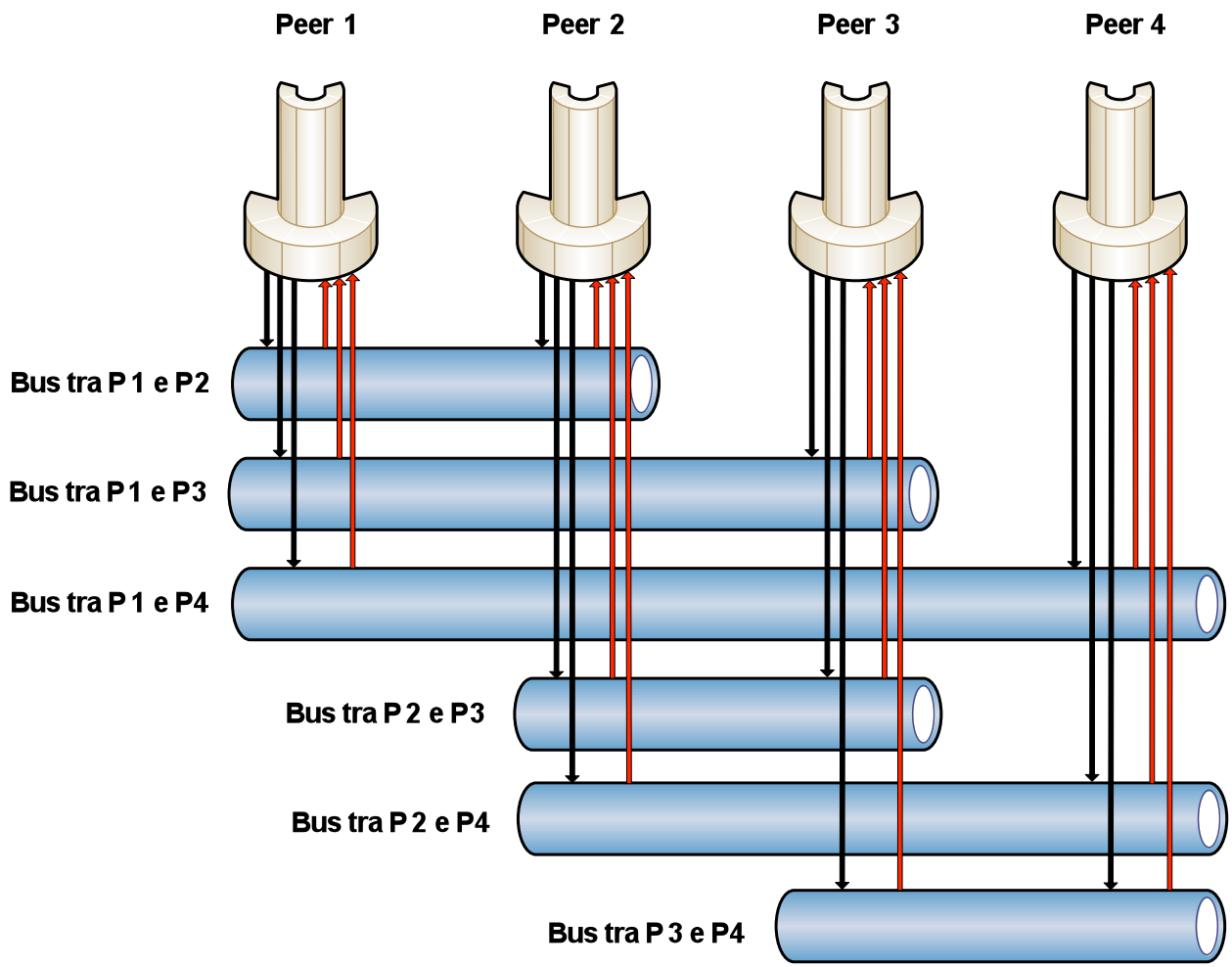
SPIN : Implementazione - Peer

Diagramma stati peer

SPIN : Implementazione - Bus



SPIN : Modello per numero arbitrario di Peer (1/2)



SPIN : Modello per numero arbitrario di Peer (2/2)

Numero di processi Bus dati dalla seguente formula:

$$\mathrm{m} = \frac{(n(n-1))}{2}$$

SPIN : Verifica delle proprietà - Terminazione

Proprietà di liveness

$$
\begin{aligned}
&(\text{Peer_1.state} = \text{Start} \longrightarrow \\
&\text{F}(\text{Peer_1.state} = \text{Complete} \bigvee \text{Peer_1.state} = \text{Fail})) \\
&\bigwedge \\
&(\text{Peer_2.state} = \text{Start} \longrightarrow \\
&\text{F}(\text{Peer_2.state} = \text{Complete} \bigvee \text{Peer_2.state} = \text{Fail}))
\end{aligned}
$$

SPIN : Verifica delle proprietà – Carrier sense

$$((\text{peer_1.state} = \text{Carrier sense} \bigwedge \text{len}(\text{state_p1}) = 0) \longrightarrow \text{X peer_1.state} = \text{Transmission}) \bigwedge$$
$$((\text{peer_1.state} = \text{Carrier sense} \bigwedge \text{len}(\text{state_p1}) = 1) \longrightarrow \text{X peer_1.state} = \text{Carrier sense}) \bigwedge$$
$$((\text{peer_2.state} = \text{Carrier sense} \bigwedge \text{len}(\text{state_p2}) = 0) \longrightarrow \text{X peer_2.state} = \text{Transmission}) \bigwedge$$
$$((\text{peer_2.state} = \text{Carrier sense} \bigwedge \text{len}(\text{state_p2}) = 1) \longrightarrow \text{X peer_2.state} = \text{Carrier sense})$$

SPIN : Verifica delle proprietà – Collision detection

$$((\text{Peer_1.state} = \text{TRANSMISSION} \bigwedge$$
$$\text{F}(\ \text{Peer_2.state} = \text{TRANSMISSION})) \longrightarrow$$
$$\text{F}(\text{Peer_1.state} = \text{Exp_backoff} \bigwedge \text{Peer_2.state} = \text{Exp_backoff})$$

SPIN : Verifica delle proprietà – Attesa di propagazione

$$((\text{Peer1_state} = \text{Propagation} \bigwedge \text{len(collision)} = 0 \bigwedge \text{i} < \text{RTT}/2) \longrightarrow$$
$$\text{X Peer_1.state} = \text{Propagation}) \bigwedge$$
$$((\text{Peer1_state} = \text{Propagation} \bigwedge \text{len(collision)} = 0 \bigwedge \text{i} = \text{RTT}/2) \longrightarrow$$
$$\text{X Peer_1.state} = \text{Complete}) \bigwedge$$
$$((\text{Peer1_state} = \text{Propagation} \bigwedge \text{len(collision} = 1)\ ) \longrightarrow$$
$$\text{X Peer_1.state} = \text{Exp_backoff}) \bigwedge$$
$$((\text{Peer2_state} = \text{Propagation} \bigwedge \text{len(collision)} = 0 \bigwedge \text{i} < \text{RTT}/2) \longrightarrow$$
$$\text{X Peer_2.state} = \text{Propagation}) \bigwedge$$
$$((\text{Peer2_state} = \text{Propagation} \bigwedge \text{len(collision)} = 0 \bigwedge \text{i} = \text{RTT}/2) \longrightarrow$$
$$\text{X Peer_2.state} = \text{Complete}) \bigwedge$$
$$((\text{Peer2_state} = \text{Propagation} \bigwedge \text{len(collision)} = 1\ ) \longrightarrow$$
$$\text{X Peer_2.state} = \text{Exp_backoff})$$