ISPCS Brescia '09

Collision avoidance for Delay_Req messages in broadcast media

Augusto Ciuffoletti
augusto@di.unipi.it

Università degli Studi di Pisa
Dipartimento di Informatica

This presentation also available as slidecast from www.slideshare.com

Outline of the presentation

- Motivation and problem description
- The algorithm
- Implementation issues
- Validation by analytical model and simulation

System model

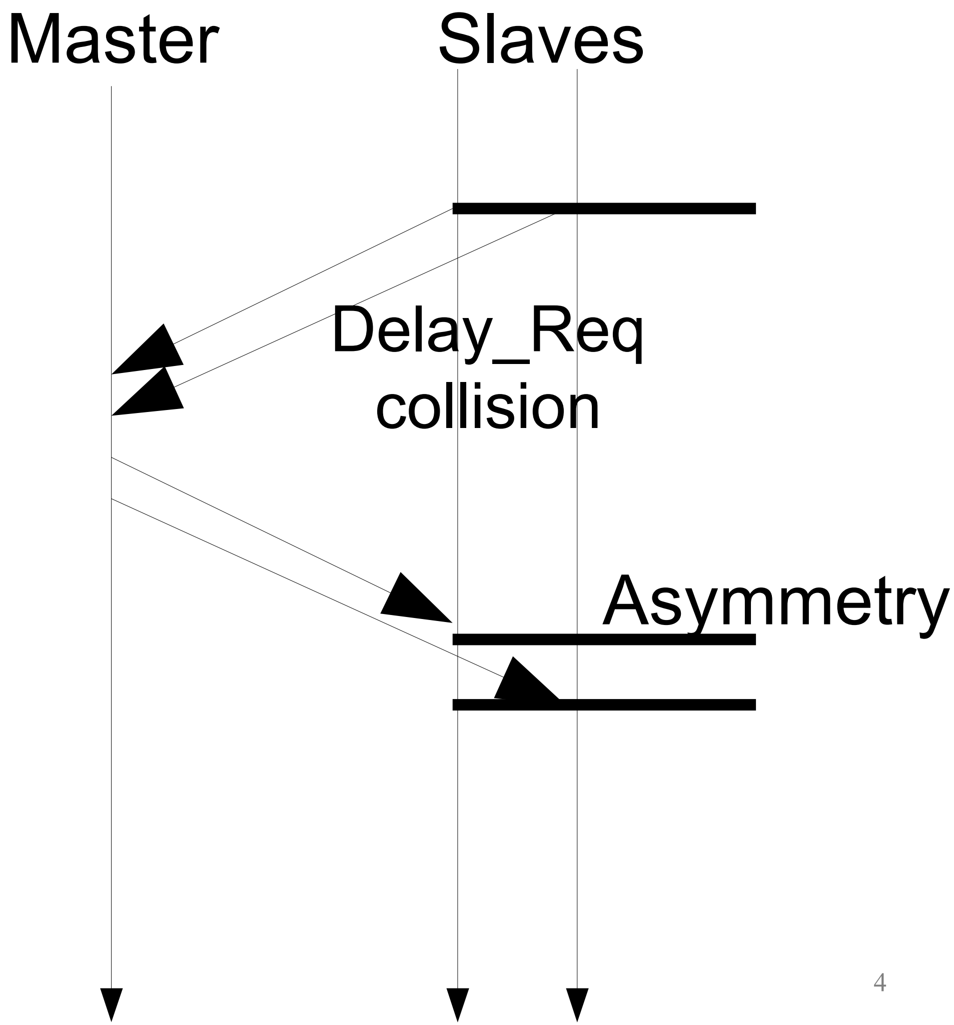
- The system is composed by a (large) number of slave clocks
- There is one master clock in charge of maintaining clock synchronized
- Network is a broadcast medium with unique collision domain
- Slaves require to be adjusted with Delay_Req/ Delay_Resp with bounded periodicity

Idea:

Exploit broadcast to avoid collision

Collision events

- Two Delay_Req msgs sent at approx the same time
- The (broadcast) network transports them in different times
- They are serviced with some delay
- Asymmetry and jitter are introduced



How frequently this event occurs?

- R: It depends on the distribution of Delay_Req events
- We assume that all slaves want to refresh their clock with the same frequency.
- IEEE1588 avoids clustering of Delay_Req messages using random delays (clause 9.5.11.2)
- This has the effect of keeping the density of Delay_Req events constant in time

Probability of collision

- We introduce two system constants:

  n: the number of slaves

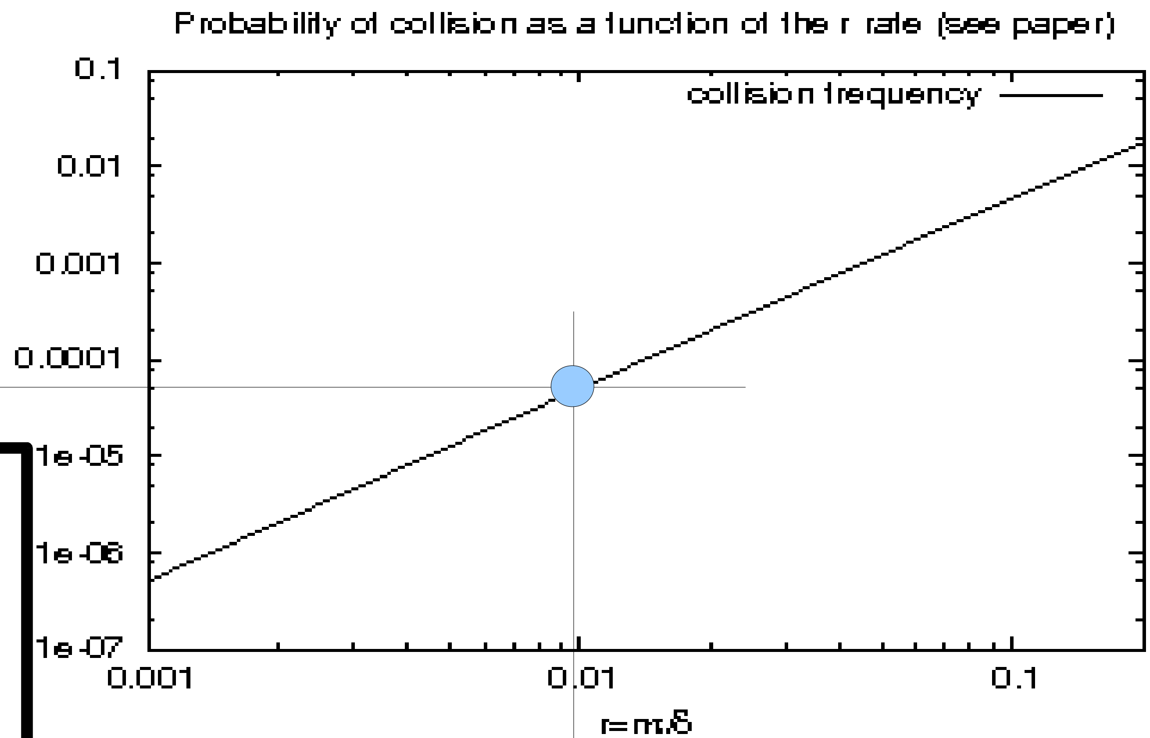
  δ: the period between successive Delay_Req on one slave
- Expected number of Delay_Req per time unit

$$\lambda = n/\delta$$

- To evaluate the probability of collision, we need to introduce the duration of the event τ

$$1-(1+r)e^{-r} \text{ with } r=\lambda\tau$$

Probability of collision



p=5 10^{-5}

τ=10 µsec
N=1000
δ=1 sec

1 collision
every 2000 secs

Requirements
for a collision avoidance algorithm

- Bounded timing of Delay_Req (from slave perspective);
- Lower probability of collision events with respect to random scheduling
- Low traffic overhead
- Embedded into existing protocol

The basic idea: token scheduling

ISPCS Brescia '09

- A token is circulated: the slave holding the token sends the Delay_Req

however

- Additional control to implement the overlay ring
- Additional bandwidth to implement token exchange
- Uncertain roundtrip time
- Compensated by (apparent) deterministic collision avoidance

Introducing random walks

- Randomized control of the overlay network (token destination selected at random in a random set of neighbors)
- Token carried by the same Delay_Req/Delay_Resp pair
- Global view to enforce bounded return time

Token circulation algorithm
slave part

ISPCS Brescia '09

- Maintain a dynamic, random list of neighbors observing the traffic on the broadcast medium
- Wait to receive a token
- Send the Delay_Req upon receiving a token
- Deliver the token to a neighbor at random

OK, but what about bounded return time?

Token circulation algorithm
master part

- The master maintains the timeouts of all slaves
- When one of the slaves is about to exceed the return time bound

  the master reroutes the token

  to feed the starving slave
- The data structure needed for the task is not discussed in the paper

  How does an IP device de-route a token...

  ...and what is a token, after all?

What is a token, anyway?

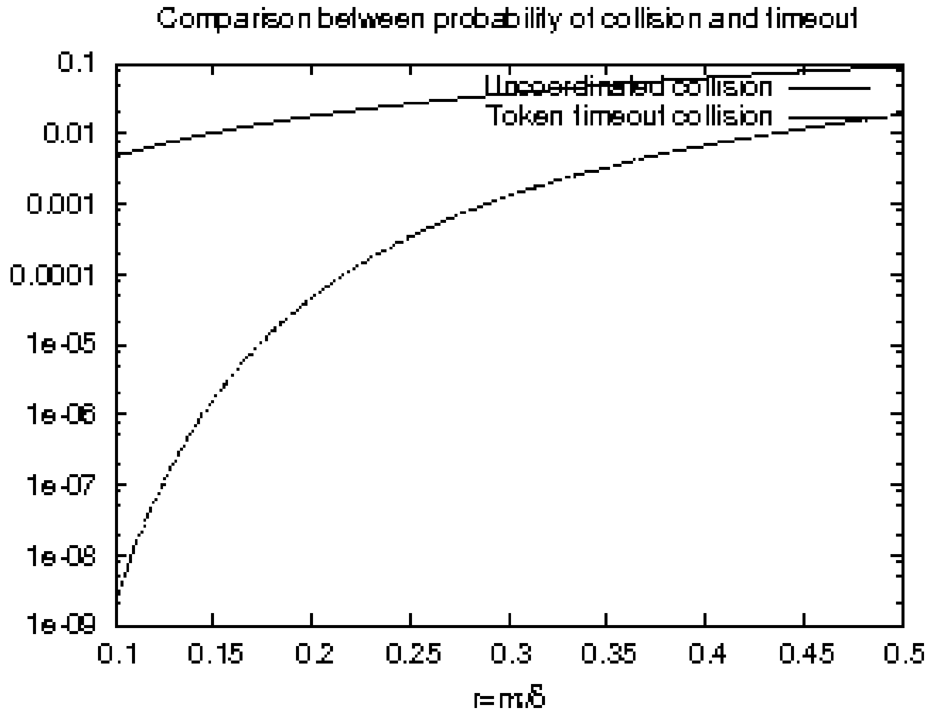
- There is no token in fact: it is just an abstraction
- **"Holding the token"** is a flag in the state of the slave
- The slave holding the token indicates, in the Delay_Req packet, the MAC of the next holder
- The master may **reroute** the token by indicating a different token holder in the Delay_Resp
- Remember that we are in a broadcast network: everybody sees every packet

Implementation issues

- Not enough room in a Delay_Req Delay_Req (5 octets + 4 bits available) frame to hold a MAC address (6 octets)
- Unless MAC are locally administered
- Our solution:
  – Use 3 octets in both Delay_Req and Delay_Resp
  – In case of ambiguity, the master disambiguates completing the MAC
  – In case of rerouting and ambiguity, two "Delay_Resp" are required (extra network load)

Evaluating the solution

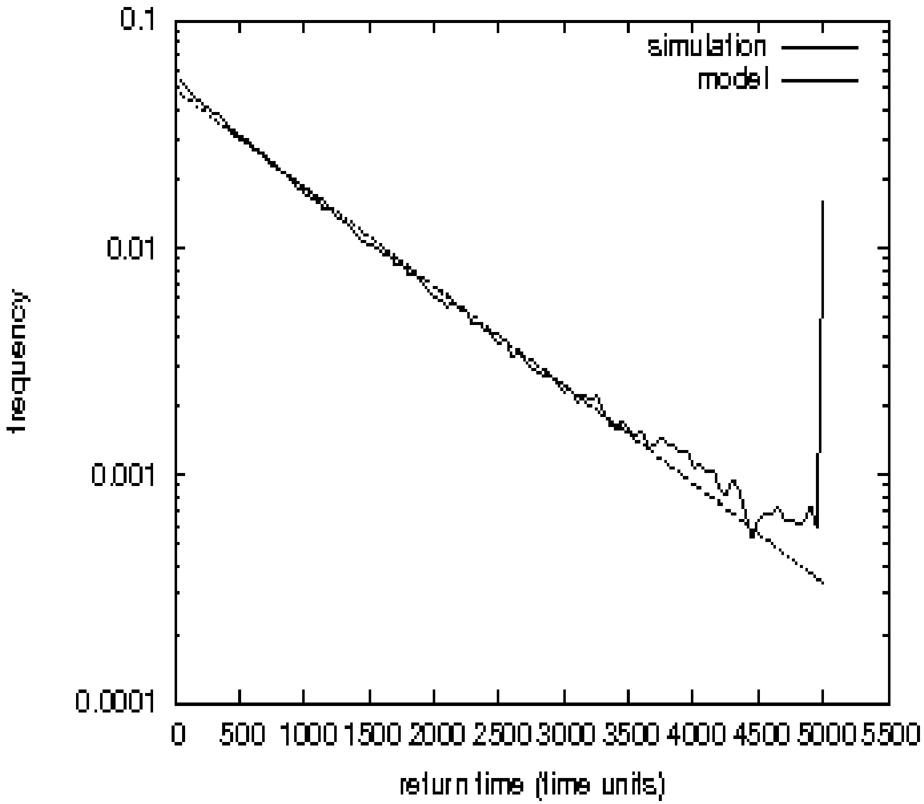
- Residual collision event: two timeout are generated at the same time
- The master selects one of them to receive the token



- The figure compares native random scheduling with token scheduling
- The model considers a full mesh overlay
- Colliding events are discarded

Evaluating the solution

- To evaluate the impact of the two approximations, we used simulation.



r=0.2

- The right spike is motivated by the slaves that receive the token as a consequence of a timeout
- The deviation is due to bounded degree approximation of the network

Conclusions

- In a broadcast network with many slaves Delay_Req messages may collide, and deteriorate synchronization
- We use the power of broadcast (the reason of the problem) to reduce the risk of collision
- Timing of Delay_Req is bounded (as required)
- No network overhead (as required)
- No change in message format (as required)