Exploiting Wireless Broadcast Property to Improve Performance of Mutual Exclusion

Ghazale Hosseinabadi and Nitin Vaidya

Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign

Introduction

- Mutual Exclusion (MUTEX) : a group of processors require to enter critical section exclusively in order to perform some critical operations.
- MUTEX algorithms : permission based, token based.
- Wireless channel : shared medium
- Messages might be overheard by nearby nodes due to broadcast nature of the channel.
- Goal : design MUTEX algorithms that exploit wireless broadcast property to improve performance.

Correctness

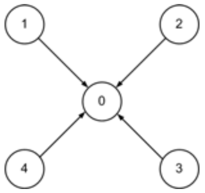
- Mutual Exclusion (safety) : At most one node is in CS at any time.
- Deadlock free (live ness) : If any node is waiting for CS, then in a finite time some node enters CS.
- Starvation free (fairness) : If a node is waiting for CS, then in a finite time it enters CS.

Performance metric

- Number of messages sent per critical section entry.

Algorithm 1

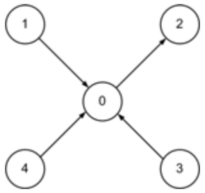
- Based on [Raymond], "A Tree-Based Algorithm for Distributed Mutual Exclusion" :
  - Messeges are sent on a spanning tree.
  - Single directed path from each node to the node holding token.
  - Spanning tree : fixed.
- Our algorithm :
  - Spanning tree : dynamic, changes in time.
  - Token is sent from A to B : any C that overhears the message, changes its parent in the tree.
  - If B is a neighbor of C, C chooses B as its parent. Otherwise, C chooses A as its parent.



Single-hop.

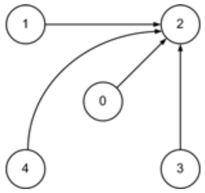
A message may be overheard by all nodes.

Initially token is held by 0.



Token is sent from 0 to 2.

Raymond : Tree after token is sent.

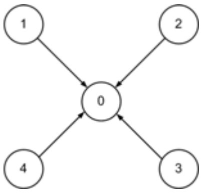


Token is sent from 0 to 2.

Our algorithm: tree after token is sent.

Algorithm 2

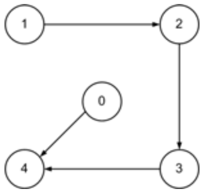
- Based on [Trehel/Naimi], "A Distributed Algorithm for Mutual Exclusion Based on Data Structures and Fault Tolerance":
  - Each node i has a variable *last*, which is the initiator of the last request message that is received at node i.
  - When a node initiates request for token, it sends its request to *last*.
- Our algorithm :
  - Multi hop: messages are sent on the shortest path between end points.
  - Id of the node initiating request for token and time of initiation is written in the request message.
  - *last* changes either by regular reception or by overhearing.



Single-hop.

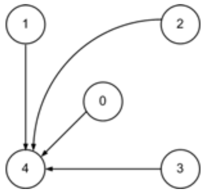
A message may be overheard by all nodes.

Initially last = 0.



1, 2, 3, 4 initiate request for token, respectively.

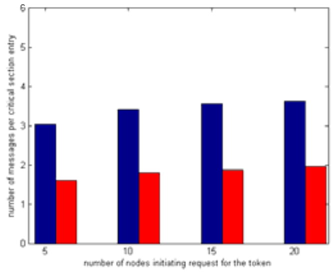
Trehel/Naimi algorithm: tree after requests are received.



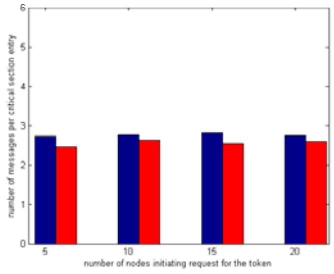
Our algorithm: tree after all requests are received.

Simulation Results

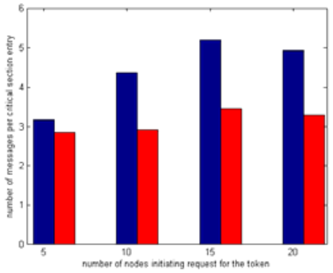
- NS-2
- 20 nodes randomly placed in the area
- area = 100m x 100m, 500m x 500m
- Each node makes next request for token *t* seconds after it exits CS.
- *t* : exponential random variable with mean λ
- Low demand : λ = 100 sec., high demand : λ = 0.005 sec.



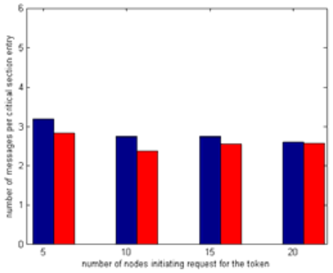
area = 100m x 100m, low demand, λ = 100 seconds.



area = 100m x 100m, high demand, λ = 0.005 second.

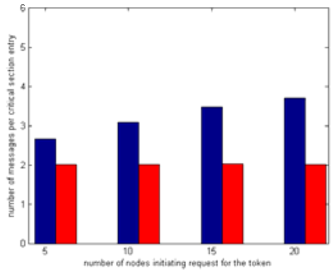


area = 500m x 500m, low demand, λ = 100 seconds.

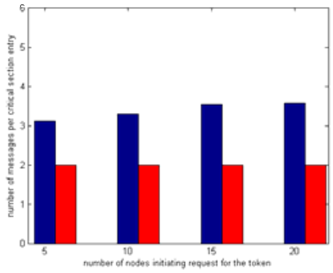


area = 500m x 500m, high demand, λ = 0.005 second.

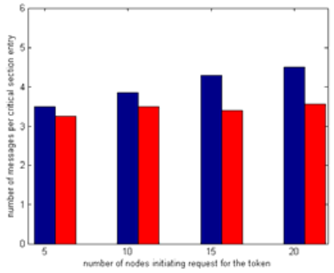
Simulation Results



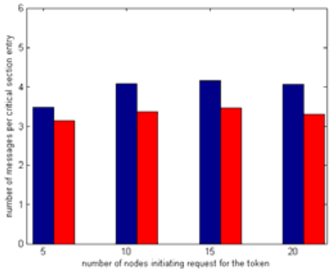
area = 100m x 100m, low demand, λ = 100 seconds.



area = 100m x 100m, high demand, λ = 0.005 second.



area = 500m x 500m, low demand, λ = 100 seconds.



area = 500m x 500m, high demand, λ = 0.005 second.

Funded in part by Boeing