**Self-stabilizing α-maximal-partitioning**

**Group 43**

**Shivneshwar Velayutham and Madhumitha Venkatesan**

**Pseudocode**

**processor\_logic()**

do forever:

temp\_set = my\_set

if i am not in my set:

temp\_set = me

if size(temp\_set) > alpha // ensures that size is never greater than alpha

temp\_set = me

if check\_if\_set\_is\_a\_connected\_tree(me, temp\_set) is false

temp\_set = me

for every neighbor:

if neighbor in temp\_set and I am not in neighbor.set and size(neighbor.set) >= size(temp\_set)

// ensures that there is no intersection between partitions

temp\_set = me

// assigns my set to me since neighbor’s set is larger

if neighbor.set has me and size(neighbor.set) > size(temp\_set)

temp\_set = neighbor.set

if neighbor is not my temp\_set and neighbor.set does not contain me and size(temp\_set) + size(neighbor.set) <= alpha

// unions two sets to ensure that this is a maximal paritition

temp\_set = temp\_set + neighbor.set

end

my\_set = temp\_set

end

As you can see from the algorithm that no partition is empty will at least have one element. Other properties of the partitioning are reached and explained with the help of the comments in the pseudocode above.

**check\_if\_set\_is\_a\_connected\_tree(root, set) // bfs algorithm**

for each p in set:

explored[p] = false

end

explored[root] = true

q = empty queue

q.enqueue(root)

while q is not null:

tmp = q.dequeue()

for each neighbor n of tmp:

if n belongs to set and explored[n] = false

explored[n] = true

q.enqueue(n)

end

end

if explored contains one false

return false

else

return true

The above functions helps when the configuration loaded is bad one has elements in the set that do not form a tree. The above is done using bfs algorithm. Thus the algorithm will stabilize with any starting system configuration.