Fault Containment Algorithms in Distributed System

By: David Quoc Truong
Faculty Sponsor: Dr. Anurag Dasgupta

Background:

- A distributed system is self-stabilizing, when starting from an arbitrary initial configuration, the system returns to a legitimate configuration in a bounded number of steps and remains in that configuration thereafter. Such arbitrary configurations may be caused by transient failures that can corrupt the system state.
- In most well-designed systems, the possibility of a massive failure is miniscule, and single failures are much more likely to occur.
- To increase the efficiency of fault-tolerance, it is important to guarantee a much faster recovery from all single failures, while also guaranteeing eventual recovery from more major failures.
- The problem of containing the effect of minor failures is becoming important not only because they are more likely to occur, but also due to the dramatic growth of network sizes.
- In most self-stabilizing systems, a single transient failure can potentially contaminate a large portion of the system.
- The tight containment of the effect of single failures depends on the context: containment in time implies that all observable variables of the system recover to their legitimate values in O(1) time, whereas containment in space means that the processes at O(1) distance from the faulty process make observable changes. For optimal performance, both of these properties should hold.

Abstract:

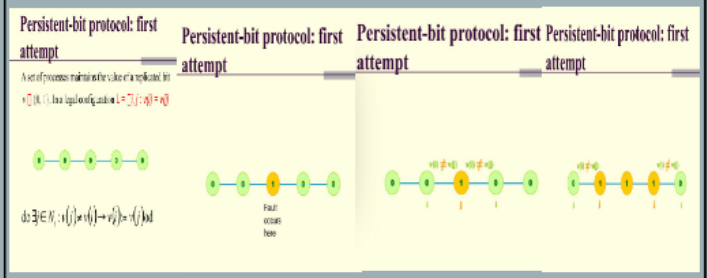
In distributed systems, fault containment is an important feature. Transient failures can lead to random configurations that can disrupt the system state. Fault containment algorithms restore the legitimate states of the system from any given configuration in a finite number of moves, which is called the convergence property of the system. The probability of a massive failure is considered low and single fault is much more likely to occur in most modern-day systems because of system reliability. We explored such single fault scenarios using a randomized scheduler. Currently we are running simulation experiments and implementing fault containment algorithms. The simulation results will provide insight regarding the efficiency of the algorithms. The experiments will take several factors into account, for example- variation in the number of nodes, the position of the faulty node, different network topologies etc.

Conclusion:

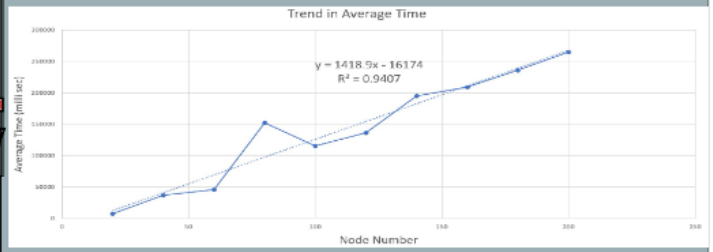
Our algorithm restricts the single fault to only its 1-distance neighbors with high probability [1]. We ran simulation experiments to evaluate the effectiveness of our algorithm. We used different parameters to see how those with variations affect the stabilization time of the network. We ran simulations with variable network size, different degrees of the nodes (sparse and dense graphs), changing the faulty node's location etc. Along with creating an algorithm, we created a random graph generator and a random scheduler to record the time within the fault gap and to see how long it took the fault to recovered thus achieving stabilization.

Persistent-bit protocol Solution:

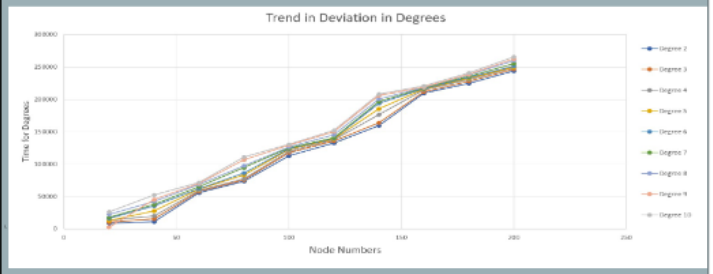
Persistent bit problem is not stabilizing with a single variable. That is why we introduced a secondary variable x in our solution.



Results:



- Figure 1: Trend in average time of selective node numbers.



- Figure 2: Trend in deviation in degrees of selective node numbers.

Algorithm and Examples:

Process i updates v(i) when the following three conditions hold -

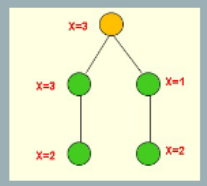
a) The randomized scheduler chooses i

b) $\exists j \in N_i : v(j) \neq v(i)$

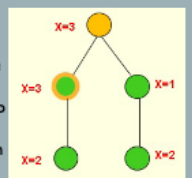
c) $\forall j \in N_i : x(i) \geq x(j)$

The corresponding action also increases x(i) to max {x(j) : j ∈ N_i } + **m**, (m>0)

1. Green node x=3 is being triggered

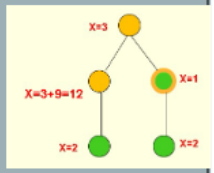


2. All three conditions are true, so it changes from green to orange

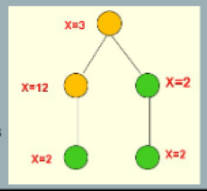


3. And we assume m=9 in this example, so x becomes 12 for that node (x= x+m)

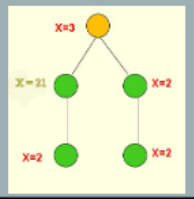
4. If the node with x = 1 is chosen, there the first two conditions hold. So, x value will be increased to 2 but its color stays the same



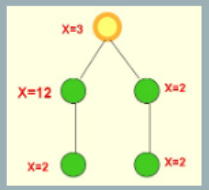
5. Now if the scheduler chooses the node with x= 12, again all three conditions are true for the node.



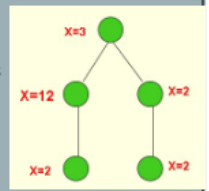
6. So, it changes to green and x becomes 21



7. Recovery completes when ∀j ∈ N_i : v(i) ≠ v(j) and the scheduler selects i. Node i flips v(i) regardless of the value of x(i)



8. So, at this point if the top most node is chosen, it turns to green and the system stabilizes.



Future Work:

The above algorithm is an extension of the persistent bit protocol solution [2, 3]. There are other fault containment algorithms [4] of similar type and in the future, we want to conduct simulation experiments for those too. We will also explore the conditions in different topologies like ring, tree, star etc. This research was supported by the Blazer Summer Research Institute at Valdosta State University.

References:

1. Probabilistic Fault-Containment. A. Dasgupta, S. Ghosh, X. Xiao, SSS 2007: 189-203.
2. Fault-local distributed mending. S. Kutten, D. Peleg. in: Proc. 14th Annu. ACM Symp. on Principles of Distributed Computing, August 1995.
3. Stabilizing time-adaptive protocols. S. Kutten, B. Patt-Shamir. Theor. Comput. Sci. 220, 93-111 (1999).
4. Fault Containment in weakly stabilizing systms. A. Dasgupta, S. Gosh, X. Xiao. Theor. Computer. Sci. 412(33): 4297-4311 (2011).