Universität Heidelberg
Wintersemester 18/19

Group Parallel and Distributed Systems (PVS)
Artur Andrzejak, Diego Costa

Problem Set 8 for lecture Distributed Systems I (IVS1)

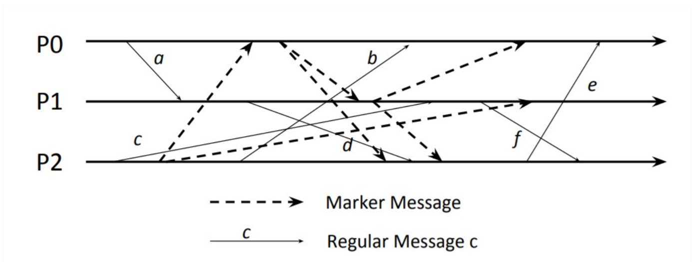
Due: 18.12.2018, 14:00 Uhr

Exercise 1 (1 Point)

The Chandy-Lamport global snapshot algorithm works under certain number of assumptions of the underlying distributed systems. List all assumptions and explain which are unlikely to hold on a real distributed system's network.

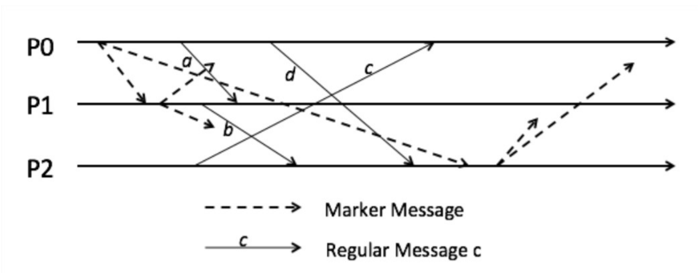
Exercise 2 (2 Points)

Using the system described below, present the global state recorded when the Chandy-Lamport snapshot algorithm has terminated.



Exercise 3 (1 Point)

In the system below, $P0$ initiates the snapshot algorithm, but Chandy-Lamport algorithm can not create a consistent snapshot of the system. Find the reason and explain why a consistent global state cannot be found with Chandy-Lamport algorithm.



Exercise 4 (2 Points)

Two processes P and Q are connected in a ring using two channels, and they constantly rotate a message m. At any one time, there is only one copy of m in the system. Each process's state consists of the number of times it has received m, and P sends m first. At a certain point, P has the message and its state is 101. Immediately after sending m, P initiates the snapshot algorithm. Explain the operation of the algorithm in this case, giving the possible global state(s) reported by it.

Exercise 5 **(6 Points)**

The Dijkstra mutual exclusion algorithm proposed in 1974[1] is one of the first self-stabilizing algorithms, responsible of boostraping the self-stabilizing research subfield of fault-tolerance algorithms. In this exercise, you will implement a simulation of the Dijsktra token ring for an arbitrary number N nodes and K states, and evaluate how long it takes for the algorithm to stabilize.

- To simulate the nodes, you can use an array/list with N positions, each containing a valid state $s = [0, .., K-1]$ with $K > N$.
- Initialize each node of the system with a unique and randomized state. This is the worst case scenario where all nodes have a token.
- On each step of your program, select a random node to perform an action. The action must follow Dijkstra's algorithm, that is, a node can only change its state if it has the privilege, and the state transition needs to comply with Dijkstra rules.
- Record both the number of steps your program take until the stabilization and the amount of states changed in the process.

Test your code with multiple configurations $N = [2, .., 20]$ and $K = N + 1$, and run each configuration at least 10 times. For this exercise, submit your code into Moodle and the answers of the following assignments:

1. Plot a chart with the average number of steps needed to reach a stable state per N. How many steps in average are necessary to stabilize the algorithm?
2. Plot a chart with the average number of states changed per N, until a stable state is reached. On average, how many times a node changes until the algorithm has stabilized?
3. What happens to the stabilization property of Dijkstra's algorithm if $K \leq N$? **Hint:** Run your code with high values of $N = [100, .., 1000]$ and low values such as $K = [2, 3, 4]$.

[1] Available at https://www.cs.utexas.edu/users/EWD/ewd04xx/EWD426.PDF