Quantitative Evaluation of Embedded Systems (IN4390)
Practical assignments 2020/2021

1 Assignment 3

Learning Objectives. In this assignment you will analyse ROS 2 networks and use both Petri nets, Markov chains and queuing theory in order to determine whether a system breaks down or not.

Deadline. The due date for mandatory questions is **Thursday 21.01.2021 at 23:59** and the due date for optional questions is **Thursday 04.02.2021 at 23:59**. These deadlines are firm. Submissions that are uploaded after the deadline will not be graded.

Deliverables. Answer the following questions in a single report and upload it on Brightspace. Please do not forget to add your names, student numbers, team number and the date of submission to the file you upload.

Assessment Instructions. You must answer all questions. The grade is on a scale from 0-10. You will *pass* this mandatory assignment only if you gain at least 6 points. If answers are incorrect, you will receive feedback from TA's and you will have one more chance to submit the assignment.

2 ROS instructions

In this lab assignment you are given a couple of systems that run in ROS2 to simulate different systems. These systems comprise ROS-nodes that send and/or transmit messages to other nodes. The python source code of the nodes can be found in the folder `lab3/src/Petrinet/Petrinet`, where question 1 uses `node.py` and question 2 uses `node2.py`. This is tested on the Ubuntu distributed for lab 1, though should work on any form of Linux with ROS2 installed. The networks can be simulated by opening the terminal in the lab3 folder and running

```
$ bash run.bash
```

or

```
$ bash run2.bash
```

You will see information coming from all the simulated nodes.
In order to stop the simulation use a second terminal and run

```
$ bash stop.bash
```

3 Mandatory questions

3.1 Q1 (2 points)

In this question we want you to analyse the source code of the two types of nodes (in `node.py` and `node2.py`) and the files for running the ROS network (`run.bash` and `run2.bash`). Go through these files and explain within x words for each of the two node types how they work. We are specifically interested in: how these nodes are setup, what their inputs are and what they resemble, how the sending of messages is handled, how the receiving of messages is handled and how the queue length influences the operation of the queue(s).

3.2 Q2 (4 points)

The system for this question can be found in `run.bash`. In this system a single node is sending messages to multiple other nodes that listen for the data. Think of a user sending data to servers for the data to be processed. If the queue of a listener overflows, a bug causes the server to crash, rendering it useless. The talking node then redirects its messages to the remaining servers.

1. Use queuing theory to determine the steady state behaviour of this system.
Hint: consider basic variables like utilization, throughput, and response time.

2. Draw a Petri net and corresponding reachability graph for this system, and use this to determine if the system has a deadlock. If yes, explain the scenario that leads to the deadlock; if no, prove there is no deadlock using the reachability graph of the Petri net. In the Petri net you do not need to worry about the listening and talking rates of the nodes. (For the reachability graph you can work wit a maximum queue size of 2 to reduce the size of the graph.)

3. Discuss the dependability of the system, how likely is it that problems occur, and, comment on how it can be improved (without buying new expensive servers).

3.3 Q3 (4 points)

The system for this question can be found in `run2.bash`. These nodes continually forward messages to other nodes. A node can only forward messages that are inside its queue and if the queue overflows the node will break.

1. Use continues time Markov chains to determine the steady state behaviour of the system and comment on the system's dependability.

2. These systems can not be, fully, modeled with standard Petri nets. What aspect in these systems causes what problems? Discus if it is a problem that you can't fully model these systems with a normal Petri net?

3. Several extensions have been proposed to remedy shortcomings of the basic model. Discuss wether or not coloured Petri nets, stochastic Petri nets and timed Petri nets would be able to model the network setup by the `run2.bash` script.

4 Optional questions

Deadline. The due date for optional questions is on **Thursday 04.02.2021 at 23:59**. The deadline is firm. Submissions that are uploaded after the deadline will not be graded.

Deliverables. Upload a single report for all of the optional questions that you would like to answer on BrightSpace. Please do not forget to add your names, student numbers, and the date of submission to the file you upload.

Assessment Instruction. If your grade for an optional question is *below* 50% of the points of that question, you will not receive any point for that question. For example, if a question has 20 points and you scored 9, then you will not receive any point for that question. Consult a TA if you are not sure if your answer will qualify for grading.

One way of preventing the problems that arise in Q1 is to have the listeners tell the talker that their queue is (nearly) full, which would result in the talker only talking to the other listeners.

4.1 Q4 (20 points)

Modify your Petri net from Q1 to include the fix and show that this solves the deadlock that can appear in the original system.

4.2 Q5 (50+10 points)

1. Modify the source code to implement this fix. You are allowed to alter both `run.bash` and `node.py` to implement this fix, but you are not allowed to alter the listening/talking rates.

2. Then compare the steady state behaviour of the system with the expected behaviour determined in Q1.

Deliverables. If you do this optional question, please hand in the modified files in a zip file with the name `lab3_Customisable_group_<group number>.zip`, together with the rapport.