The files

- pollingsystem2.prism and
- pollingsystem2.props

contain a PRISM model generated for a variant of the LPEP example program given in section II.C of the paper

"An Approach to Performance Evaluation Programming"
Author: Eneia Nicolae Todoran (SYNASC 2017 submission 68)

There is just one difference between the LPEP program discussed in section II.C of the paper and the LPEP program described in this file (and implemented in the Haskell file Lpep2.hs), namely, the LPEP command

```
[serve1] (vs1=1) & (vVal1>=4) -> ?vIndex1 : (vs1:=0) & (vVal1:=vVal1);
```

was replaced with

```
[serve1] (vs1=1) & (vVal1>=4) -> ?vIndex1 : (vs1:=0) & (vVal1:=1);
```

and a similar replacement was performed for serve2. Therefore vVal1 (vVal2) evolves as follows: 1,2,3,4,1,2,3,4,1,2,3,4,....

The implication is that in the long run variables vIndex1 and vIndex2 are more evenly distributed among the possible values 0,1,2,3. In the final part of this file we present an experiment that involves variable vIndex1. For further explanations see also the readme file available from the parent directory.

We expect that this modification should not affect the overall behavior of the system. Indeed, we will see that if we repeat the experiments included in the parent directory we get similar results.

For the experiments presented below in the PRISM file pollingsystem2.prism we have included the following declarations:

```
const int Wfile;
const int Idx;
```

The constant Wfile is used in the Server module as follows:

```
module Server
  wfile : [1..41] init Wfile;
  …
```

The following experiment (presented below, and also in Figure 1, section III of the paper) shows the probability that in the long run station 1 is awaiting service:

```
S=? [ vs1=1 & !(vs=1 & va=1) ]
```

In this experiment:

- N ranges from 1 to 20 with step 4
- sched ranges from 10 to 50 with step 20
- Wfile=4
- Idx=1

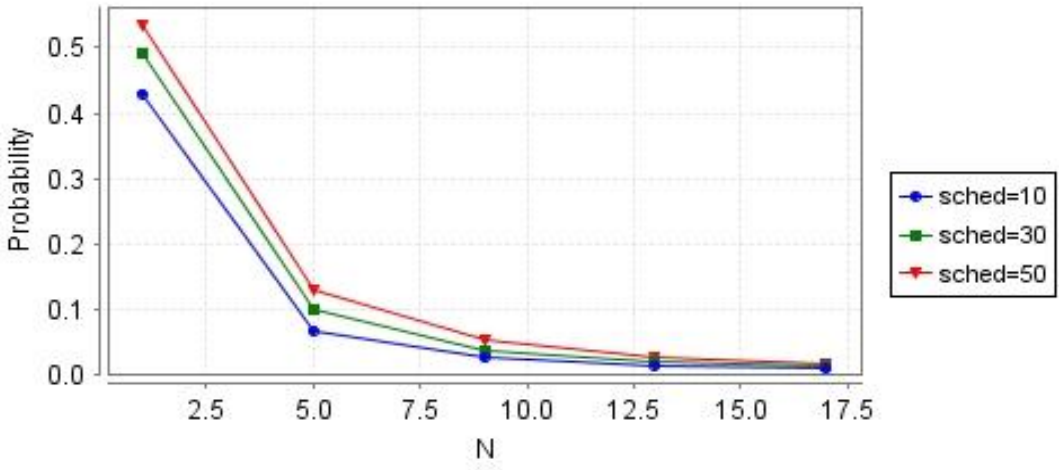

Figure 1. Probability that in the long run station 1 is awaiting service

For the experiment given below (and also in Figure 2, section III of the paper) we defined a rewards structure:

```
rewards "served" // expected number of times Station1 is served
        [serve1] true : 1;
endrewards
```

We compute the expected reward (number of times station 1 is served) accumulated by time T as follows:

```
R{"served"}=?[C<=T]
```

In this experiment

- we fixed sched=10
- N ranges from 1 to 20 with step 4
- T ranges from 1 to 15 with step 3
- Wfile=4
- Idx=1

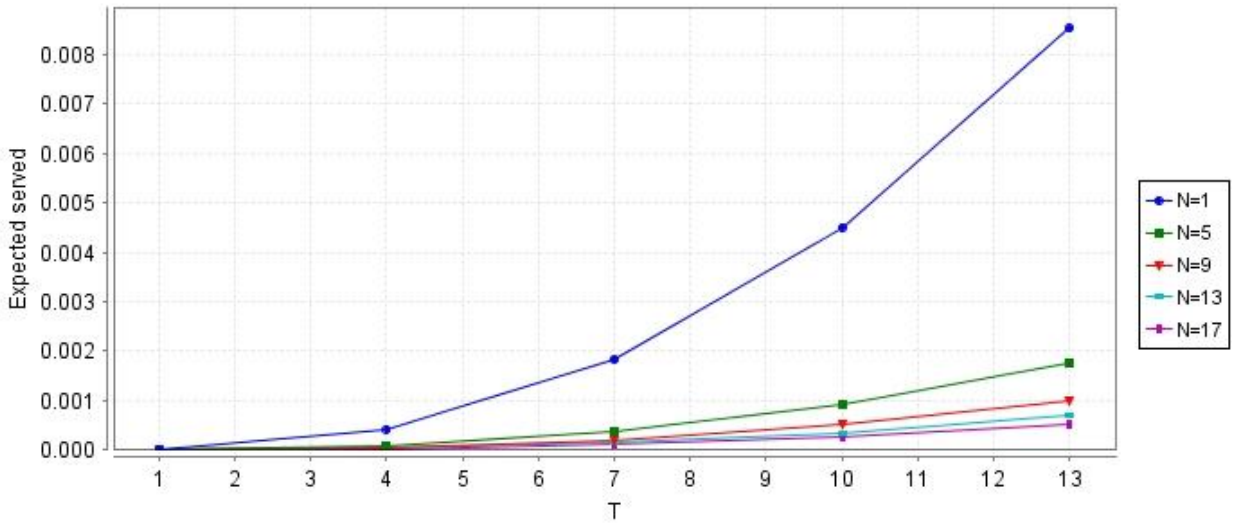

Figure 2. Number of times station 1 is served

The following experiment shows the probability that in the long run station 1 is idle:

S=? [vs1=0]

In this experiment:

- N ranges from 1 to 20 with step 4
- sched ranges from 10 to 50 with step 20
- Wfile=4
- Idx=1

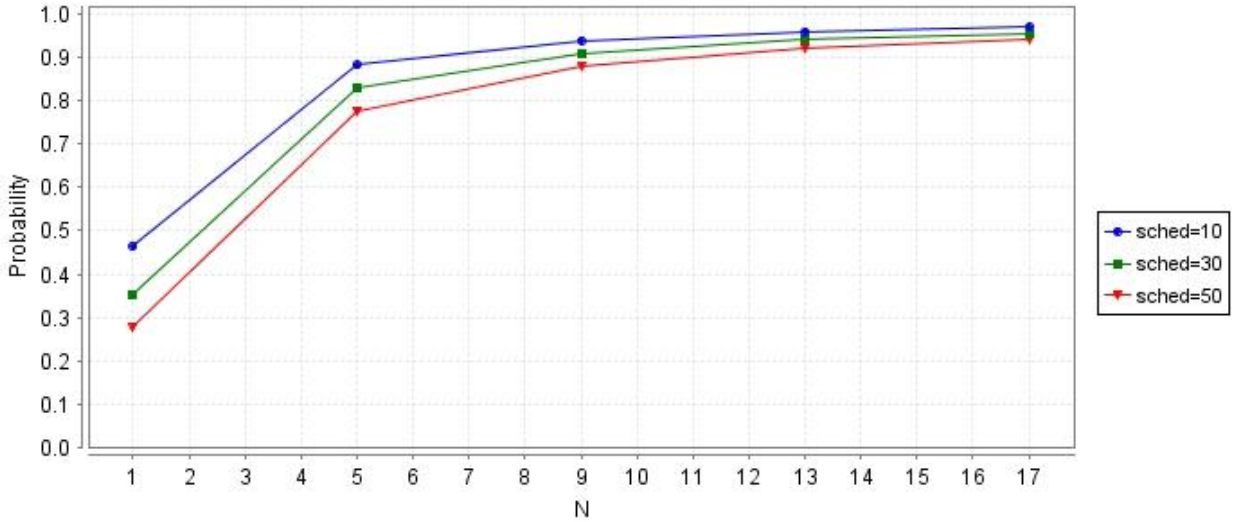

Figure 3. Probability that in the long run station 1 is idle

Next we investigate behavior of variables vIndex1 and vIndex2. Statistically, the behavior of these variables does not seem interesting. The experiment presented below shows no trends. The main purpose of this experiment is just to show that variables used in LPEP receive statements can be monitored just like any other variables.

The values for vIndex1 and vIndex2 are computed by the function findex in corresponding send activities. For example, assuming that vVal1=1, wfile=cons[Z] 3 (nil[Z]) which is encoded as 4 (see section III of the paper), and vmax=3, the send statement (!(((findex vVal1) 1) vmax) wfile) executed by the LPEP command

[serve1] (vs=1)&(va=1) -> !(((findex vVal1) 1) vmax) wfile : (va:=2)&(vVal1Cp:=vVal1);

yields 2, representing the index of (vVal1=1) in (cons[Z] 3 (cons[Z] 1 (nil[Z]))), which will be the value of wfile after the corresponding call to finsert. The following code is generated in the PRISM file for this command (the function ((((findex vVal1) 1) vmax) wfile) is evaluated in 23 basic or elementary steps):

[serve1] (vs=1)&(va=1)&(vVal1=1)&(wfile=4) -> 1/(23 + updsched) : (va'=2)&(vVal1Cp'=vVal1);

In order to monitor the behavior of vIndex1 in the PRISM file in module Station1 we generate the following command (which has a passive rate)

[serve1] (vs1=1) & (vVal1=1) & (wfile=4) -> 1 : (vIndex1’=2) & (vs1'=0) & (vVal1'=vVal1+1);

This value (namely 2) is computed by ((((findex vVal1) 1) vmax) wfile), transmitted to module Station1 and assigned to vIndex1.

In pollingsystem2.prism we have generated a different receive command (which synchronizes with a corresponding send command executed by module Server) for each different combination of values for vs1 and vVal1, (and similarly for vIndex2, vs2 and vVal2, which are used in synchronizations upon the action serve2).

vVal1 evolves as follows: 1,2,3,4,1,2,3,4,1,2,3,4,...., hence the value of vIndex1 in the long run should be evenly distributed among the possible values 0,1,2,3. However, the activities that are used to compute the values of vIndex1 proceed at different rates, therefore we cannot expect to have

(S=? [vIndex1=0])=(S=? [vIndex1=1])= (S=? [vIndex1=2])= (S=? [vIndex1=3])=0.25.

Indeed, in the experiment given below N=2, sched=10, Wfile ranges from 1 to 41 with step 1, and Idx ranges from 0 to 3 with step 1. The figure below shows the probability that in the long run vIndex1=Idx:

S=? [vIndex1=Idx]

In this experiment Idx ranges from 0 to 3 with step 1, Wfile ranges from 1 to 41 with step 1, N=2, and sched=10.

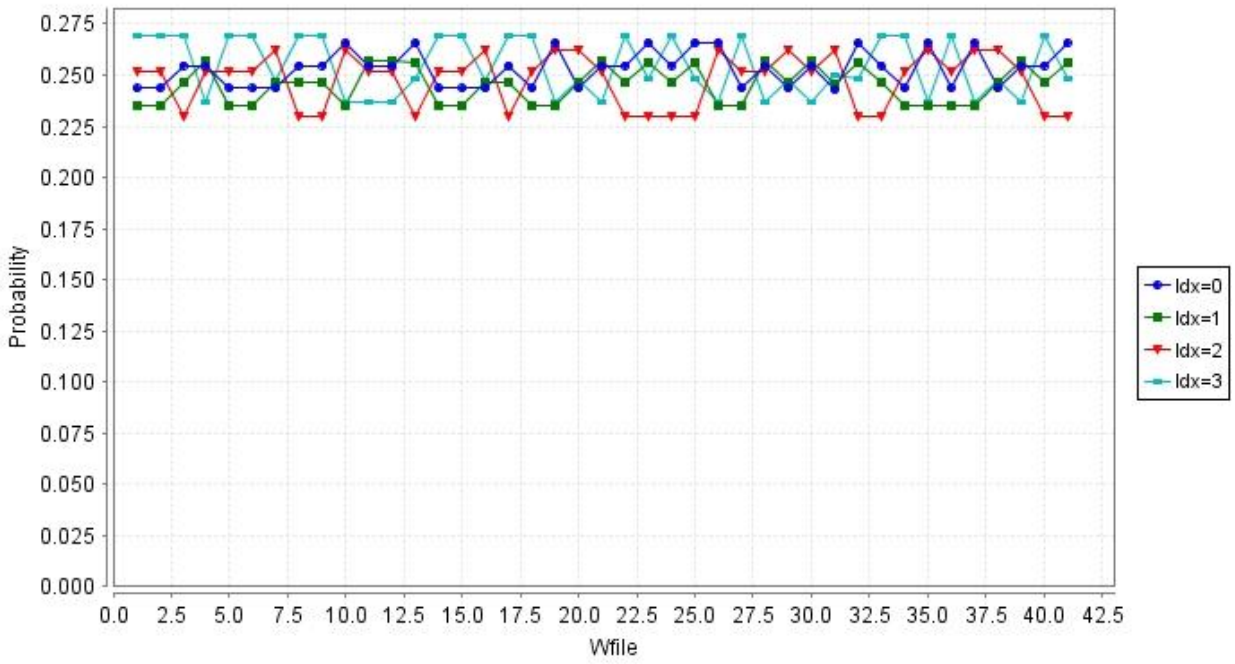

Figure 4. Probability that in the long run vIndex1=Idx, for Idx=0,1,2,3

Statistically, the behavior of variable vIndex1 does not seem very interesting. The experiment given in Figure 4 shows no trends. The main purpose of this experiment is just to show that variables used in LPEP receive statements can be monitored like any other variables. In other applications such variables could display a more interesting behavior.

Clearly, we always have

(S=? [vIndex1=0])+(S=? [vIndex1=1])+(S=? [vIndex1=2])+(S=? [vIndex1=3]) = 1

because variable vIndex1 is declared as follows in the module Station1

vIndex1 : [0..3] init 0;