Analytic Models

Assignment 3

Mina Mousavifar - 11279515 - sem311

Question 1.

For *small packets* we have:

$S_{small} = 0.01ms$, $P_{small} = 0.1$.
$\lambda_{small} = P_{small} * \lambda = (0.1) * (8000\frac{packet}{second} * \frac{1second}{1000ms}) = 0.8\frac{packet}{ms}$
So based on Little's law:
$U_{small} = \lambda_{small} * S_{small} = (0.8\frac{packets}{ms}) * (0.01ms) = 0.008$
$Q_{small} = \lambda_{small} * R_{small} = 0.8R_{small}$
Since we have a deterministic service time distribution for each class, then we have:
$S_{rem_{small}} = \frac{S_{small}}{2} = \frac{0.01ms}{2} = 0.005ms$

For *big packets* we have:

$S_{big} = 0.11ms$, $P_{big} = 0.9$.
$\lambda_{big} = P_{big} * \lambda = (0.9) * (8000\frac{packet}{second} * \frac{1second}{1000ms}) = 7.2\frac{packet}{ms}$
So based on Little's law:
$U_{big} = \lambda_{big} * S_{big} = (7.2\frac{packets}{ms}) * (0.11ms) = 0.792$
$Q_{big} = \lambda_{big} * R_{big} = 7.2R_{big}$
Since we have a deterministic service time distribution for each class, then we have:
$S_{rem_{big}} = \frac{S_{big}}{2} = \frac{0.11ms}{2} = 0.055ms$

Finally for each class based on small class non-preemptive priority we have:

$$R_{small} = S_{small} + S_{small} * (Q_{small} - U_{small}) + U_{small} * S_{rem_{small}} + U_{big} * S_{rem_{big}}$$

$$R_{small} = 0.01 + 0.01 * (0.8R_{small} - 0.008) + 0.008 * 0.005 + 0.792 * 0.055$$

```
Rs = (0.01 - (0.01*0.008) + (0.008*0.005) + (0.792*0.055))/(1 - 0.008)
cat("R small: ", Rs, 'milliseconds')
```

```
## R small:  0.05395161 milliseconds
```

$R_{big} = S_{big}+S_{small}*(Q_{small}-U_{small})+S_{big}*(Q_{big}-U_{big})+U_{small}*S_{rem_{small}}+U_{big}*S_{rem_{big}}+(R_{big}-S_{big})\lambda_{small}S_{small}$

$R_{big} = 0.11+0.01*(0.8*R_{small}-0.008)+0.11*(7.2R_{big}-0.792)+0.008*0.005+0.792*0.055+(R_{big}-0.11)*0.8*0.01$

```
Rb = (0.11+(0.01*(0.8*Rs-0.008))-(0.11*0.792)+(0.008*0.005)+(0.792*0.055)-(0.11*0.8*0.01))/(1-0.8)
cat("R big: ", Rb, 'milliseconds')
```

```
## R big:  0.3297581 milliseconds
```

Question 2.

```
# initialization
Q = seq(0,0,length.out=2)
R = seq(0,0,length.out=2)
maxpreR = seq(0,0,length.out=2)
N = 1:30
X = 0
Z = 30
f_val = (0:100)*0.01
for(i in N){
  optimum_x = -1
  optimum_f = -1  # finding best f for minimum delay
  R_tot = 0
  # checking each f value
  for(f in f_val){
    # find max troughput based on MVA
    for(k in 1:2){
      if(k==1){
        # fast server
        R[k] = (f)*(1.5)*(1+(f*Q[k]))
      } else{
        # slow server
         R[k] = (1-f)*(3)*(1+((1-f)*Q[k]))
      }
    }
    R_tot = sum(R)
    X = (i)/(Z + R_tot)
    if(X > optimum_x){
      optimum_x = X
      optimum_f = f
      maxpreR = R
    }
  }
  for(k in 1:2){
    Q[k] = maxpreR[k]*optimum_x
  }
  cat("Number of customers: ", i, " Optimal f based on MVA: ", optimum_f, "\n")
}
```

```
## Number of customers:  1  Optimal f based on MVA:  1
## Number of customers:  2  Optimal f based on MVA:  1
## Number of customers:  3  Optimal f based on MVA:  1
## Number of customers:  4  Optimal f based on MVA:  1
## Number of customers:  5  Optimal f based on MVA:  1
## Number of customers:  6  Optimal f based on MVA:  1
## Number of customers:  7  Optimal f based on MVA:  1
## Number of customers:  8  Optimal f based on MVA:  1
## Number of customers:  9  Optimal f based on MVA:  0.93
## Number of customers:  10  Optimal f based on MVA:  0.88
## Number of customers:  11  Optimal f based on MVA:  0.86
## Number of customers:  12  Optimal f based on MVA:  0.82
```

```
## Number of customers:  13  Optimal f based on MVA:  0.82
## Number of customers:  14  Optimal f based on MVA:  0.77
## Number of customers:  15  Optimal f based on MVA:  0.79
## Number of customers:  16  Optimal f based on MVA:  0.74
## Number of customers:  17  Optimal f based on MVA:  0.77
## Number of customers:  18  Optimal f based on MVA:  0.71
## Number of customers:  19  Optimal f based on MVA:  0.76
## Number of customers:  20  Optimal f based on MVA:  0.68
## Number of customers:  21  Optimal f based on MVA:  0.75
## Number of customers:  22  Optimal f based on MVA:  0.67
## Number of customers:  23  Optimal f based on MVA:  0.73
## Number of customers:  24  Optimal f based on MVA:  0.66
## Number of customers:  25  Optimal f based on MVA:  0.72
## Number of customers:  26  Optimal f based on MVA:  0.65
## Number of customers:  27  Optimal f based on MVA:  0.71
## Number of customers:  28  Optimal f based on MVA:  0.64
## Number of customers:  29  Optimal f based on MVA:  0.7
## Number of customers:  30  Optimal f based on MVA:  0.63
```

So based on these results, again we see that the optimal f converges to $f = \frac{S_{slow}}{S_{slow}+S_{fast}} = \frac{2}{3} = 0.69$

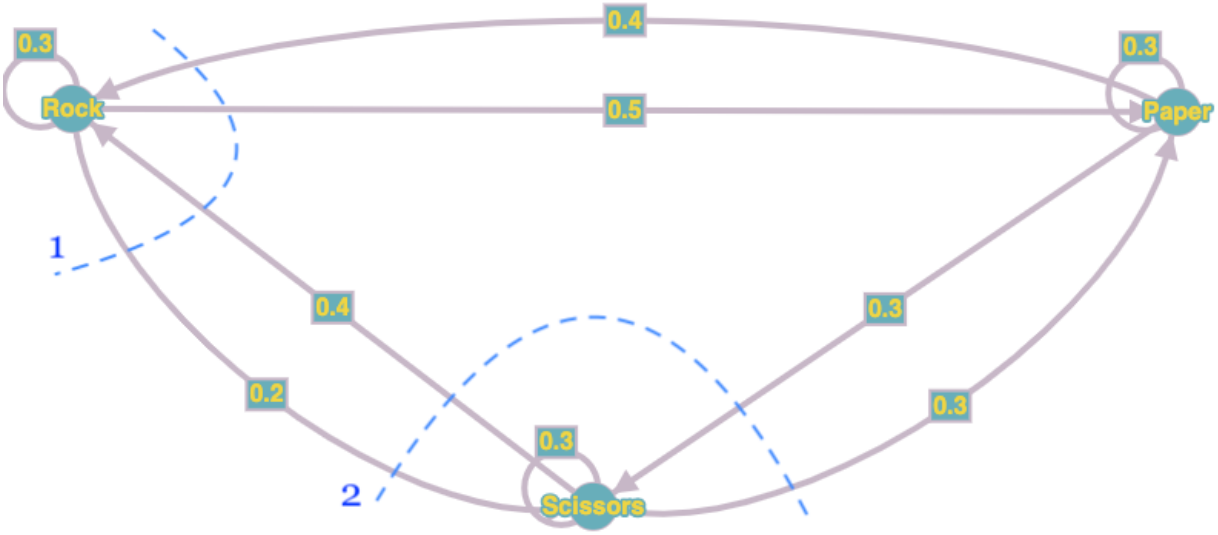

Figure 1: rps state-transition model

Question 3.

a)

This model is provided in figure 1.

b)

We can obtain the following equations by dividing the graph as figure 1.

$$\begin{cases} P_{rock} + P_{paper} + P_{scissors} = 1 \\ 1)0.4P_{paper} + 0.4P_{scissors} = 0.5P_{rock} + 0.2P_{rock} \\ 2)0.2P_{rock} + 0.3P_{paper} = 0.4P_{scissors} + 0.3P_{scissors} \end{cases}$$

The results are $P_{rock} = \frac{4}{11}$, $P_{paper} = \frac{41}{110}$, $P_{scissors} = \frac{29}{110}$

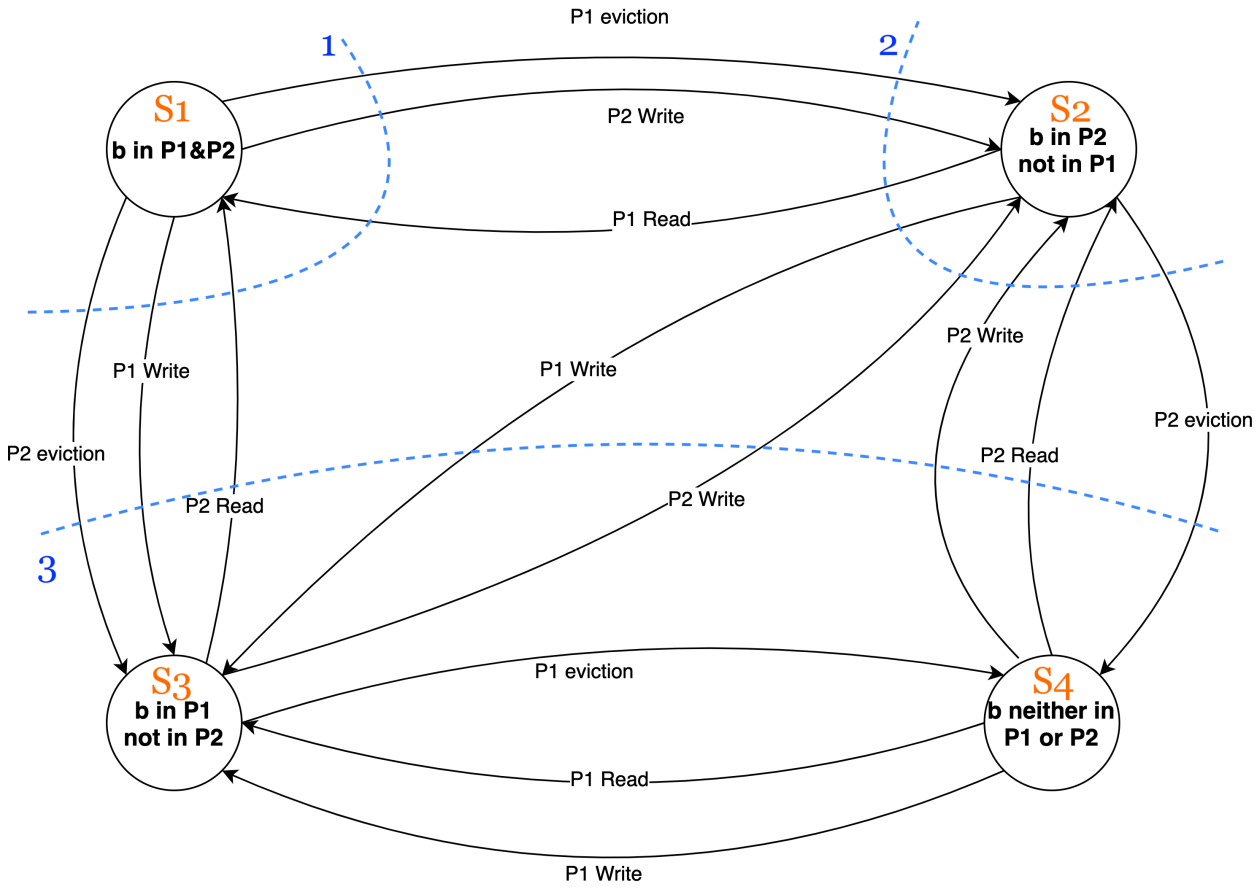

Figure 2: system state-transition model

Question 4.

a)

This model is provided in figure 2.

b)

We can obtain the following equantions by dividing the graph as figure 2. Read rate for each processor is $\frac{1}{R}$ and Write rate for each processor is $\frac{1}{W}$.

$$
\begin{cases}
S_1 + S_2 + S_3 + S_4 = 1 \\
\\
1) P_1Eviction * S_1 + P_2Eviction * S_1 + P_1Write * S_1 + P_2Write * S_1 = P_1Read * S_2 + P_2Read * S_3 \\
\rightarrow \lambda * S_1 + \lambda * S_1 + \frac{1}{W} * S_1 + \frac{1}{W} * S_1 = \frac{1}{R} * S_2 + \frac{1}{R} * S_3 \\
\\
2) P_1Read * S_2 + P_2Write * S_2 + P_2Eviction * S_2 = \\
P_1Eviction * S_1 + P_2Write * S_1 + P_2Write * S_3 + P_2Write * S_4 + P_2Read * S_4 \\
\rightarrow \frac{1}{R} * S_2 + \frac{1}{W} * S_2 + \lambda * S_2 = \lambda * S_1 + \frac{1}{W} * S_1 + \frac{1}{W} * S_3 + \frac{1}{W} * S_4 + \frac{1}{R} * S_4 \\
\\
3) P_2Eviction * S_1 + P_1Write * S_1 + P_1Write * S_2 + P_2Eviction * S_2 = \\
P_2Read * S_3 + P_2Write * S_3 + P_2Write * S_4 + P_2Read * S_4 \\
\rightarrow \lambda * S_3 + \frac{1}{W} * S_1 + \frac{1}{W} * S_2 + \lambda * S_2 = \frac{1}{R} * S_3 + \frac{1}{W} * S_3 + \frac{1}{W} * S_4 + \frac{1}{R} * S_4
\end{cases}
$$

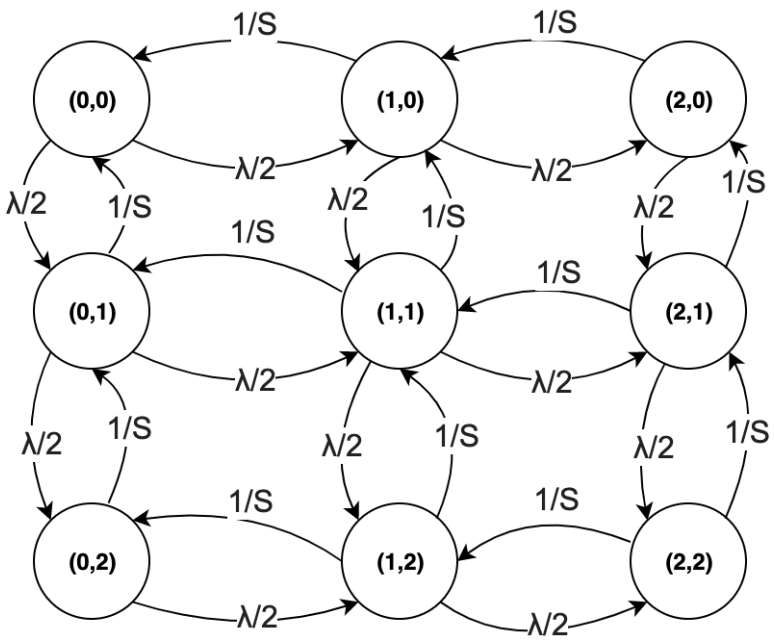

Figure 3: two servers state-transition model

Finally:

$$\begin{cases} S_1 + S_2 + S_3 + S_4 = 1 \\ 1) 2S_1\lambda + \frac{2S_1}{W} = \frac{S_2+S_3}{R} \\ 2) \frac{S_2}{R} + \frac{S_2}{W} + S_2\lambda = S_1\lambda + \frac{S_1+S_3+S_4}{W} + \frac{S_4}{R} \\ 3) S_3\lambda + S_2\lambda + \frac{S_1+S_2}{W} = \frac{S_3+S_4}{W} + \frac{S_3+S_4}{R} \end{cases}$$

Question 5.

a)

We define states as a set of two values, which shows requests in server 1(consider m) & server 2(consider n). Starting at (0,0) which means there is no work in both servers.

Then by rate of $\frac{\lambda}{2}$ we will have a transition to (1,0) or (0,1). Then by rate of $\frac{1}{S}$ will have a transition to (0,0) from (1,0) or (0,1). From (1,0), we might go to (2,0), (1,1) by rate of $\frac{\lambda}{2}$. From (0,1), we might go to (0,2), (1,1) by rate of $\frac{\lambda}{2}$.

So the formulation can be as following:

- State(m,n) $\rightarrow$ State(m+1,n) by rate $\frac{\lambda}{2}$
- State(m,n) $\rightarrow$ State(m,n+1) by rate $\frac{\lambda}{2}$
- State(m,n) $\rightarrow$ State(m-1,n) by rate $\frac{1}{S}$
- State(m,n) $\rightarrow$ State(m,n-1) by rate $\frac{1}{S}$

Sample diagram is shown in figure 3.

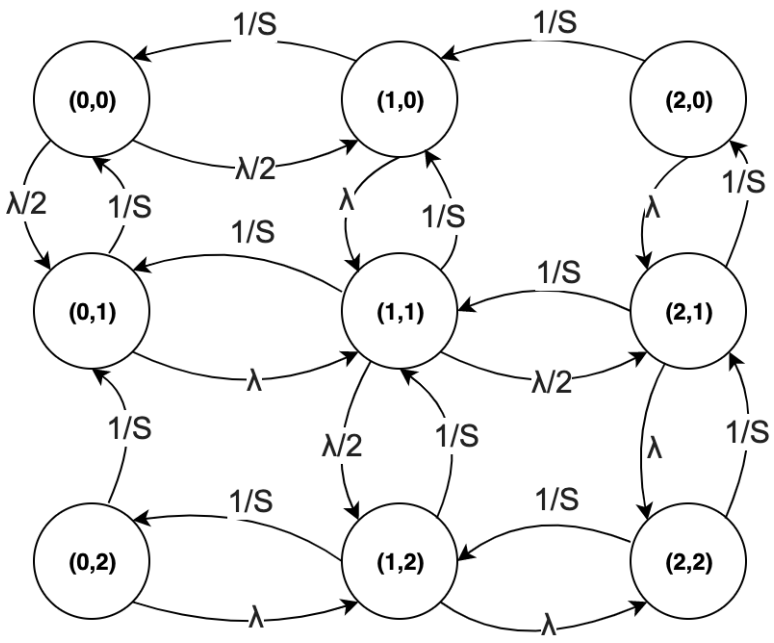

Figure 4: two servers state-transition model

b)

Again we consider the same satate definition as part a. Starting at (0,0) which means there is no work in both servers.

Then by rate of $\frac{\lambda}{2}$ we will have a transition to (1,0) or (0,1). Then by rate of $\frac{1}{S}$ will have a transition to (0,0) from (1,0) or (0,1). From (1,0), we can't go to (2,0) in contrast with part a, however we might go to (1,1) by rate of λ. From (0,1), we can't go to (0,2), but we might go (1,1) by rate of λ.

So the formulation can be as following:

- State(m,n) $\rightarrow$ State(m+1,n), if m = n by rate $\frac{\lambda}{2}$
- State(m,n) $\rightarrow$ State(m+1,n), if m $<$ n by rate λ
- State(m,n) $\rightarrow$ State(m,n+1), if m = n by rate $\frac{\lambda}{2}$
- State(m,n) $\rightarrow$ State(m,n+1), if m $>$ n by rate λ
- State(m,n) $\rightarrow$ State(m-1,n) by rate $\frac{1}{S}$
- State(m,n) $\rightarrow$ State(m,n-1) by rate $\frac{1}{S}$

Sample diagram is shown in figure 4. We should note that we can reach node (2,0) by moving to (2,1) from (1,1) by rate of $\frac{\lambda}{2}$ and then server 2 serving request by rate of $\frac{1}{S}$

c)

Here we change states so that we can find previous served server (mentioned as A & B) in addition to the number of requests in each server. So our states have three values. We assume that the round robin routing starts with routing request to server 1.

Starting at (0,0,A) which means there is no work in both servers. Then by rate of λ we will have a transition to (1,0,A). Then by rate of λ we will have a transition to (1,1,B). From (1,1,B) we might go to (0,1,B) or (1,0,B) by rate of $\frac{1}{S}$.

So the formulation can be as following:

- State(m,n,A) $\rightarrow$ State(m,n+1,B) by rate λ

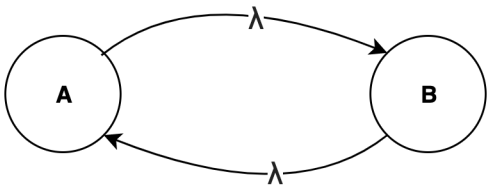

Figure 5: simple round robin state-transition model

- State(m,n,B) $\rightarrow$ State(m+1,n,A) by rate λ
- State(m,n,A) $\rightarrow$ State(m-1,n,A) by rate $\frac{1}{S}$
- State(m,n,B) $\rightarrow$ State(m,n-1,B) by rate $\frac{1}{S}$

However I think in this problem servers serving doesn't change the state and we can simply define our state space as a one dimentional value of routed server. Then by rate of λ we move from Server A to Server B and vice-versa. Sample diagram for 1 dimentional state is shown in figure 5.

Sample diagram for 3 dimentional state is also shown in figure 6.

d)

In my opinion, second routing algorithm might have better performance, because it tries to keep the utilization of the less crowded server by preventing inactivity, and prevent overflowing of the more crowded server.

Then I think third routing algorithm might have better performance than first one, because worst case of part a algorithm might end in overflowing one server severly and keeping the other server idle in this time.

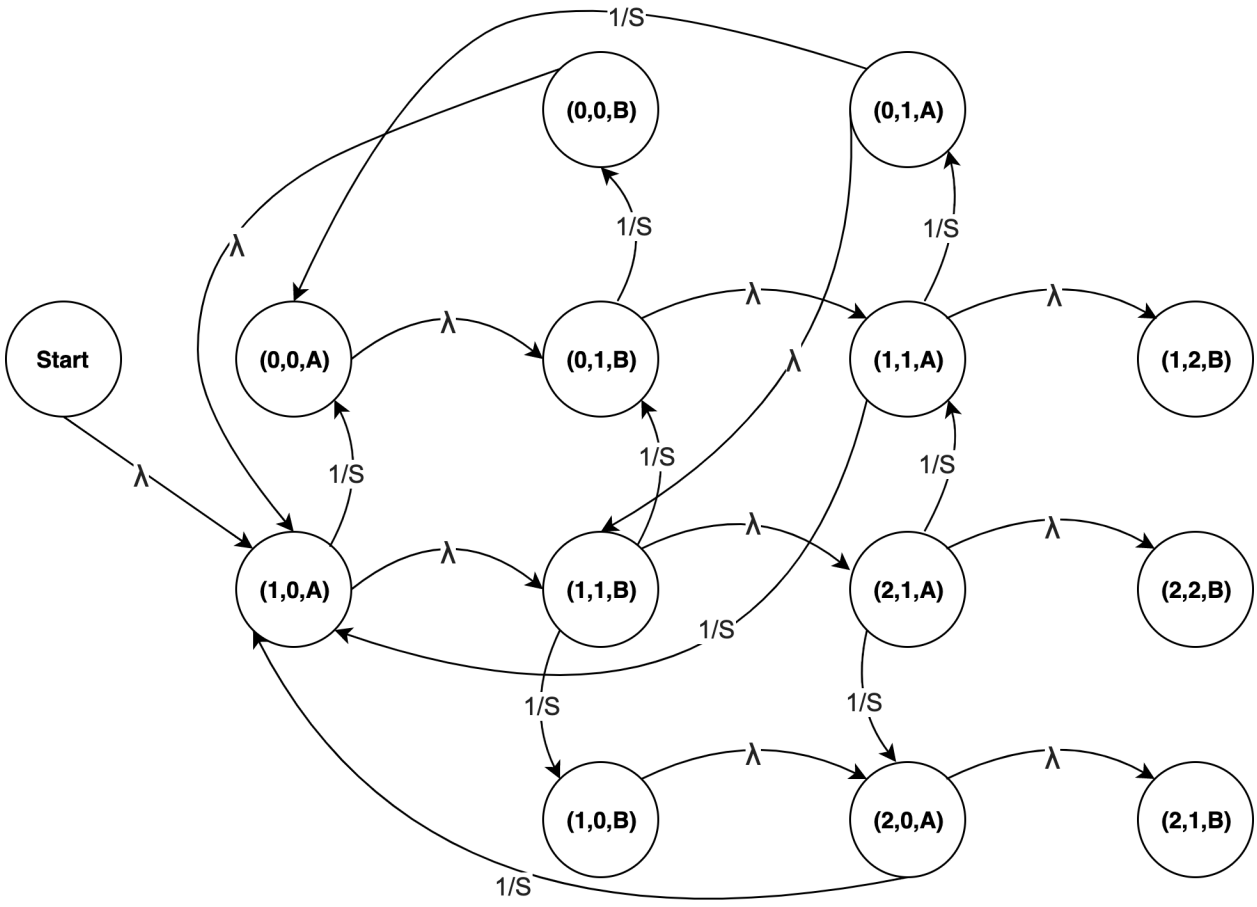

Figure 6: round robin state-transition model