Table 1: Tempi esecuzione sperimentali (espessi in μs) ottenuti da vari test

$\lambda_{controller}(1)$	Total mean arrival rate into *controller* of class 1 tasks
$\lambda_{controller}(2)$	Total mean arrival rate into *controller* of class 2 tasks
$\mu_{cloudlet}(1)$	Mean service rate at *cloudlet* of class 1 tasks
$\mu_{cloudlet}(2)$	Mean service rate at *cloudlet* of class 2 tasks
$\mu_{cloud}(1)$	Mean service rate at *cloud* of class 1 tasks
$\mu_{cloud}(2)$	Mean service rate at *cloud* of class 2 tasks

1 Notation and Modeling for Classed Jackson Networks

2 Analytical solution

In this last section we will develop an analytical solution to validate the results obtained previously through our simulations.

2.1 System based on access control Algorithm 1

Let's start with presentation of analytical solution of system based on access control algorithm 1.

In order to compute all parameters and metrics of interest associated with above-mentioned system, due to cloudlet's limited resources according to which it can accept jobs until their number does not exceed a given threshold N, is crucial compute first the **fraction of jobs that are forwarded to cloudlet and to cloud**; to do it, we must compute at first probability according to which the sum of job of each class in cloudlet system is equal to that threshold.

To determine this probability, we had modelled cloudlet with a **continuous-time Markov chain** (**CTMC**), of which you can see a graphical representationin Figure 1, where each chain's state, denoted with (n_1, n_2), is represented by the number of class 1 job, n_1, and class 2 job, n_2, present in system at a certain moment.

2.1.1 Balance equation computing

Obviously we can compute **limiting probabilities** $\pi_{(n_1,n_2)}$, namely the probability according to which the chain is in a certain state, say j, independently of the starting state, say i, via **balance** (or **stationary**) **equations**, in which we can equate the rate at which the system leaves state j with the rate at which the system enters state j [1], **remembering that limiting probabilities sum to 1** (i.e., $\sum_{j=0}^{\infty} \pi_j = 1$). These balance equations, shown in table 2, are been resolved using a very simple MATLAB script called `MATLAB_ALG1_CTMC_ResolverScript.m` in which each equation, previously generated through a Java script[2], is been resolved using a MATLAB function called `solve(eqn,var)`.[3].

[1] *Cfr.* Mor Harchol-Balter - *Performance Modeling and Design of Computer Systems* - Carnegie Mellon University, Pennsylvania, pag. 237

[2] Cfr. `CTMCResolverScriptGenerator.java` and `ResolverUsingRoutingAlgorithm1.java` files.

[3] *Cfr.* `https://www.mathworks.com/help/symbolic/solve.html`

Figure 1: Access control algorithm 1 based cloudlet's CTMC.

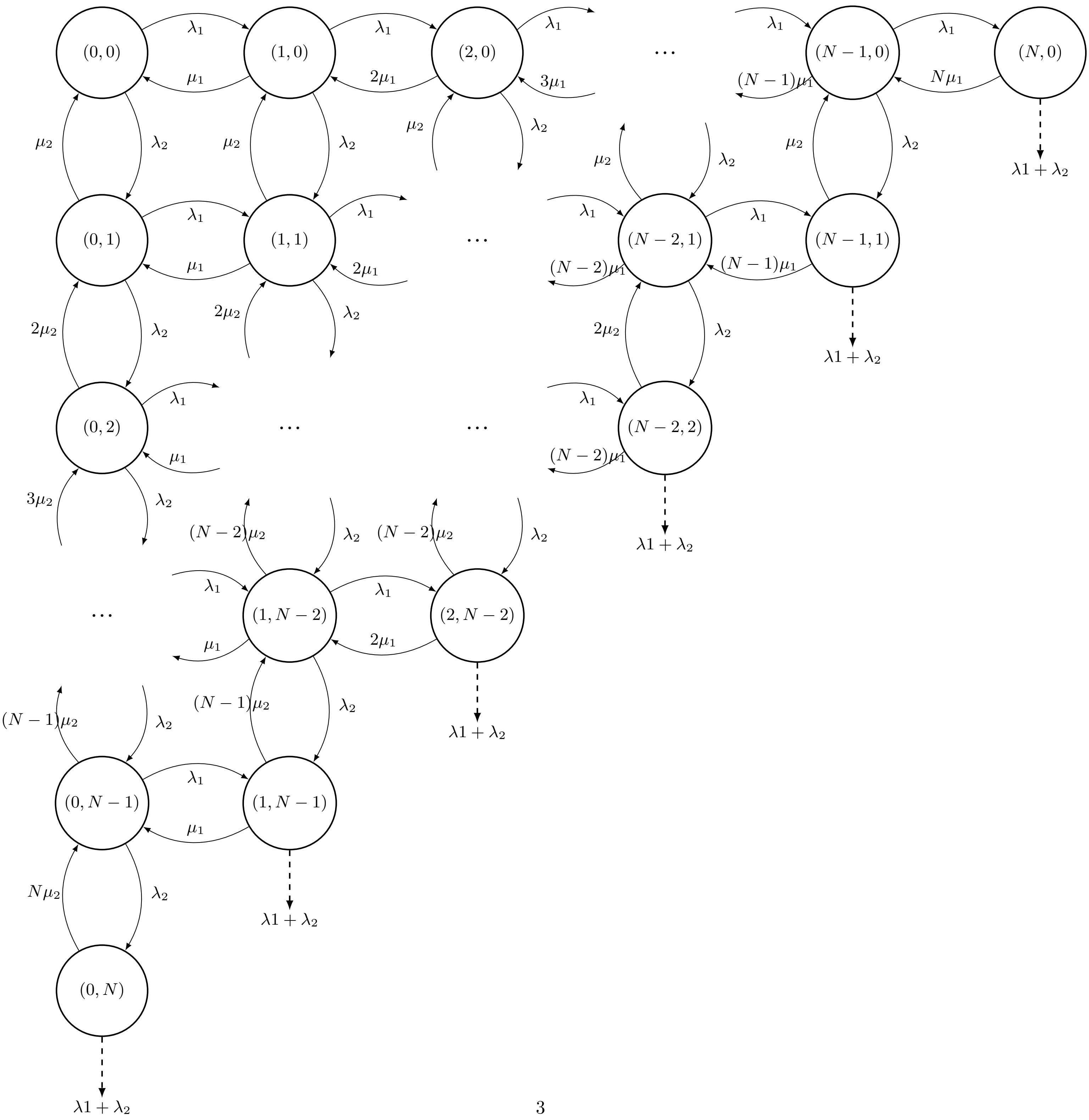


Table 2: Balance equations.

$(\lambda_1+\lambda_2)\pi_{(0,0)}$	$=$	$\mu_1\pi_{(1,0)}+\mu_2\pi_{(0,1)}$	
$(\lambda_1+\lambda_2+n_1\mu_1)\pi_{(n_1,0)}$	$=$	$\lambda_1\pi_{(n_1-1,0)}+\mu_1(n_1+1)\pi_{(n_1+1,0)}+\mu_2\pi_{(n_1,1)}$	$\forall n_1\in\mathbb{N}\cap[1,N-1]$
$(\lambda_1+\lambda_2+n_2\mu_2)\pi_{(0,n_2)}$	$=$	$\lambda_2\pi_{(0,n_2-1)}+\mu_1\pi_{(1,n_2)}+\mu_2(n_2+1)\pi_{(0,n_2+1)}$	$\forall n_2\in\mathbb{N}\cap[1,N-1]$
$\mu_1 N\pi_{(N,0)}$	$=$	$\lambda_1\pi_{(N-1,0)}$	
$\mu_2 N\pi_{(0,N)}$	$=$	$\lambda_2\pi_{(0,N-1)}$	
$(n_1\mu_1+n_2\mu_2)\pi_{(n_1,n_2)}$	$=$	$\lambda_1\pi_{(n_1-1,n_2)}+\lambda_2\pi_{(n_1,n_2-1)}$	$\forall n_1,n_2\in\mathbb{N}\cap[1,N-1]\mid n_1+n_2=N$
$(\lambda_1+\lambda_2+n_1\mu_1+n_2\mu_2)\pi_{(n_1,n_2)}$	$=$	$\lambda_1\pi_{(n_1-1,n_2)}+\lambda_2\pi_{(n_1,n_2-1)}+\mu_1(n_1+1)\pi_{(n_1+1,n_2)}$ $+\,\mu_2(n_2+1)\pi_{(n_1,n_2+1)}$	$\forall n_1,n_2\in\mathbb{N}\cap[1,N-1]\mid n_1+n_2<N$

2.1.2 Probabilities computing

Having found the stationary probabilities, we can now find $\Pi_{\mathbf{SendToCloud}}$, that is the **probability that an arriving job on controller has to be forwarded to cloud**. Observe that the class to which an arrival job belongs to is not important because, according to access control Algorithm 1, **jobs of both classes have same probability to be sent to cloud**. To be more precise, $\Pi_{\mathbf{SendToCloud}}$ is the probability that an arrival job find that the number of jobs present in cloudlet has exceeded threshold N, which occurs when $n_1+n_2=N$. Formally:

$$\begin{aligned}\Pi_{\text{SendToCloud}} &= P\{\text{An arrival job on controller sees } N \text{ jobs in cloudlet}\}\\ &= \text{Limiting probability that there are } N \text{ jobs in system}\\ &= \sum_{\substack{n_1,n_2\in\mathbb{N}\cap[0,N]\\ n1+n2=N}} \pi_{(n_1,n_2)}\end{aligned} \tag{1}$$

At this point we can easily compute $\Pi_{\mathbf{SendToCloudlet}}$, which instead represents the **probability according to which an arriving job on controller has to be accepted on cloudlet** and it is same for both job classes too.

$$\begin{aligned}\Pi_{\text{SendToCloudlet}} &= P\{\text{An arrival job on controller sees less than } N \text{ jobs in cloudlet}\}\\ &= 1-P\{\text{An arrival job on controller sees } N \text{ jobs in cloudlet}\}\\ &= 1-\Pi_{\text{SendToCloud}}\end{aligned} \tag{2}$$

2.1.3 Average arrival rates

Is $\lambda_i(c)$ the **total arrival rate into a system's component i of class c job**. Applying previous results, using an appropriate equation[4], we can now compute per-class average arrival rates as follow:

$$\begin{aligned} \lambda_{\text{cloud}}(1) &= \lambda_1 \cdot \Pi_{\text{SendToCloud}} \\ \lambda_{\text{cloud}}(2) &= \lambda_2 \cdot \Pi_{\text{SendToCloud}} \\ \lambda_{\text{cloudlet}}(1) &= \lambda_1 \cdot \Pi_{\text{SendToCloudlet}} \\ \lambda_{\text{cloudlet}}(2) &= \lambda_2 \cdot \Pi_{\text{SendToCloudlet}} \end{aligned} \tag{3}$$

Then we get λ_i, that is **total arrival rate to system's component i**, by summing the per-class rates as follows:

$$\begin{aligned} \lambda_{\text{cloud}} &= \lambda_{\text{cloud}}(1) + \lambda_{\text{cloud}}(2) \\ \lambda_{\text{cloudlet}} &= \lambda_{\text{cloudlet}}(1) + \lambda_{\text{cloudlet}}(2) \end{aligned} \tag{4}$$

2.1.4 Average population

Is $E[N_i](c)$ the **average number of class c jobs into a component i.**

We can use previously computed stationary probabilities to get average population for cloudlet; it's enough to sum each state's limiting probability multiplied by corresponding number of job as follows:

$$\begin{aligned} E[N_{\text{cloudlet}}](1) &= \sum_{(n_1,n_2)\in M} n_1 \cdot \pi(n_1,n_2) \\ E[N_{\text{cloudlet}}](2) &= \sum_{(n_1,n_2)\in M} n_2 \cdot \pi(n_1,n_2) \\ E[N_{\text{cloudlet}}] &= E[N_{\text{cloudlet}}](1) + E[N_{\text{cloudlet}}](2) \\ &= \sum_{(n_1,n_2)\in M} (n_1+n_2) \cdot \pi(n_1,n_2) \end{aligned} \tag{5}$$

Since we have modelled cloud component as a $M/M/\infty$ system, in which there is no job's waiting time due to presence of an infinite number of servers, we can simply to apply **Little's Law**[5] in order to get cloud's average population as shown in Equation 6.

$$\begin{aligned} E[N_{\text{cloud}}](1) &= \lambda_{\text{cloud}}(1) \cdot E[T_{\text{cloud}}](1) \\ &= \lambda_{\text{cloud}}(1) \cdot (E[T_{Q_{\text{cloud}}}](1) + E[S_{\text{cloud}}](1)) \\ &= \lambda_{\text{cloud}}(1) \cdot (\tfrac{1}{\mu_{\text{cloud}}(1)}) \\ &= \frac{\lambda_{\text{cloud}}(1)}{\mu_{\text{cloud}}(1)} \end{aligned} \tag{6}$$

[4] *Cfr. Ivi*, pag. 315, equation (18.1)
[5] *Cfr. Ivi*, pag. 95, theorem (6.1)

Similarly:

$$E[N_{\text{cloud}}](2) = \frac{\lambda_{\text{cloud}}(2)}{\mu_{\text{cloud}}(2)} \tag{7}$$

$$E[N_{\text{cloud}}] = E[N_{\text{cloud}}](1) + E[N_{\text{cloud}}](2) \tag{8}$$

Finally we can get global average job populations as follows:

$$\begin{aligned} E[N](1) &= E[N_{\text{cloudlet}}](1) + E[N_{\text{cloud}}](1) \\ E[N](2) &= E[N_{\text{cloudlet}}](2) + E[N_{\text{cloud}}](2) \\ E[N] &= E[N](1) + E[N](2) \end{aligned} \tag{9}$$

2.1.5 Average response time

Is $E[T_i](c)$ the **mean response time experienced by a class c jobs into a component i**.

In order to properly compute said metric for each system's component observe that:

- Knowing per-class average job population and per-class mean arrival rates, we can easily compute $E[T_i](c)$ using Little's Law.
- Since our system haven't queues, because of there is no waiting time experienced by jobs, is true that $E[T_i](c)$ is also equal to $E[S_i](c)$, that is **mean service time experienced by a class c jobs into a component** i, which is equal to $1/\mu_i(c)$, where $\mu_i(c)$ **means average service rate at which a class c jobs into a component i is served**.

Therefore we can get these metric as follows.

$$\begin{aligned} E[T_{\text{cloudlet}}](1) &= \frac{E[N_{\text{cloudlet}}](1)}{\lambda_{\text{cloudlet}}(1)} \\ &= E[S_{\text{cloudlet}}](1) \\ &= \frac{1}{\mu_{\text{cloudlet}}(1)} \end{aligned} \tag{10}$$

$$\begin{aligned} E[T_{\text{cloudlet}}](2) &= \frac{E[N_{\text{cloudlet}}](2)}{\lambda_{\text{cloudlet}}(1)} \\ &= E[S_{\text{cloudlet}}](2) \\ &= \frac{1}{\mu_{\text{cloudlet}}(2)} \end{aligned} \tag{11}$$

$$
\begin{aligned}
E[T_{\text{cloud}}](1) &= \frac{E[N_{\text{cloud}}](1)}{\lambda_{\text{cloud}}(1)} \\
&= E[S_{\text{cloud}}](1) \\
&= \frac{1}{\mu_{\text{cloud}}(1)}
\end{aligned} \tag{12}
$$

$$
\begin{aligned}
E[T_{\text{cloud}}](2) &= \frac{E[N_{\text{cloud}}](2)}{\lambda_{\text{cloud}}(2)} \\
&= E[S_{\text{cloud}}](2) \\
&= \frac{1}{\mu_{\text{cloud}}(2)}
\end{aligned} \tag{13}
$$

At this point we can get global per-class mean response times as follows:

$$
\begin{aligned}
E[T](1) &= E[T_{\text{cloudlet}}](1) \cdot P\{\text{An arrival class 1 job is sent to cloudlet}\} \\
&\quad + E[T_{\text{cloud}}](1) \cdot P\{\text{An arrival class 1 job is sent to cloud}\} \\
&= E[T_{\text{cloudlet}}](1) \cdot \Pi_{\text{SendToCloudlet}} + E[T_{\text{cloud}}](1) \cdot \Pi_{\text{SendToCloud}}
\end{aligned} \tag{14}
$$

Similarly:

$$
\begin{aligned}
E[T](2) &= E[T_{\text{cloudlet}}](2) \cdot P\{\text{An arrival class 2 job is sent to cloudlet}\} \\
&\quad + E[T_{\text{cloud}}](2) \cdot P\{\text{An arrival class 2 job is sent to cloud}\} \\
&= E[T_{\text{cloudlet}}](2) \cdot \Pi_{\text{SendToCloudlet}} + E[T_{\text{cloud}}](2) \cdot \Pi_{\text{SendToCloud}}
\end{aligned} \tag{15}
$$

Finally, using obtained results, we can get global mean response times as shown below:

$$
E[T] = E[T](1) \cdot \frac{\lambda_1}{\lambda_1 + \lambda_2} + E[T](2) \cdot \frac{\lambda_2}{\lambda_1 + \lambda_2} \tag{16}
$$

2.1.6 Throughput

To determine system's throughput let's prove if our system is stable. As we know, every queueing system in which its mean arrival rate is less than its mean service rate is known as a stable system; formally, are λ and μ our system's mean arrival rate and mean service rate respectively, if $\lambda \leq \mu$, given system is stable. Cloudlet subsystem is clearly *not* stable due to its limited resources compared to its workload; we have already seen that exists a not null probability according to which an arriving job on controller sees N jobs in cloudlet which implying its forward to cloud subsystem. Since it haven't a queue, cloudlet stability condition is achieved when $\Pi_{\text{SendToCloud}}$ is null but, based on current system parameters, is not the case. Regarding cloud subsystem, no matter how high we make λ_{cloud} because it is made up of an infinite number of server for which completion rate is still bounded by the arrival rate. Accordingly to previous considerations we can conclude that **our system is stable** due to stability or its cloud subsystem therefore we can get system throughput as follows:

$$X = \lambda = \lambda_1 + \lambda_2 \tag{17}$$

To get per-subsystems throughput we proceed as shown below:

$$X_{\text{cloud}} = \lambda_{\text{cloud}} \tag{18}$$

$$X_{\text{cloudlet}} = X - X_{\text{cloud}} \tag{19}$$

2.1.7 Summary of analytical results

	Class 1 Jobs			Class 2 Jobs			Cloudlet	Cloud	Global
	Cloudlet	Cloud	System	Cloudlet	Cloud	System			
$\Pi_{\text{SendToCloudlet}}$									0.5861
$\Pi_{\text{SendToCloud}}$									0.4139
X, λ (jobs/s)	2.3444	1.6556	4	3.663125	2.586875	6.25	6.007525	4.242475	10.25
$E[N]$ (jobs)	5.2040	6.6224	11.8264	13.5521	11.75852273	25.31062273	18.7561	18.38092273	37.13702273
$E[S], E[T]$ (s)	20/9	4	2.835407407	100/27	50/11	4.225763636	3460/1107	1954/451	3.683185595
ρ	0.2604888889			0.6783564815			0.9388453704		

2.2 System based on access control Algorithm 2

$$
\begin{cases}
(\lambda_1+\lambda_2)\pi_{(0,0)} = \mu_1\pi_{(1,0)} + \mu_2\pi_{(0,1)} & \\
(\lambda_1+\lambda_2+i\mu_1)\pi_{(i,0)} = \lambda_1\pi_{(i-1,0)} + \mu_1(i+1)\pi_{(i+1,0)} + \mu_2\pi_{(i,1)} & \forall i \in \mathbb{N} \mid 1 \le i \le S-1 \\
(\lambda_1+\lambda_2+i\mu_2)\pi_{(0,i)} = \lambda_2\pi_{(0,i-1)} + \mu_1\pi_{(1,i)} + \mu_2(i+1)\pi_{(0,i+1)} & \forall i \in \mathbb{N} \mid 1 \le i \le S-1 \\
(S\mu_1+\lambda_1)\pi_{(S,0)} = \lambda_1\pi_{(S-1,0)} + \lambda_1\pi_{(S-1,1)} + (S+1)\mu_1\pi_{(S+1,0)} & \\
(\lambda_1+S\mu_2)\pi_{(0,S)} = \lambda_2\pi_{(0,S-1)} & \\
(i\mu_1+j\mu_2+\lambda_1)\pi_{(i,j)} = \lambda_1\pi_{(i-1,j)} + \lambda_2\pi_{(i,j-1)} & \forall i,j \in \mathbb{N} \mid \begin{matrix} 1 \le i \le S-1 \\ 1 \le j \le S-1 \end{matrix} \mid i+j=S \\
(\lambda_1+\lambda_2+i\mu_1+j\mu_2)\pi_{(i,j)} = \lambda_1\pi_{(i-1,j)} + \lambda_2\pi_{(i,j-1)} + \mu_1(i+1)\pi_{(i+1,j)} + \mu_2(j+1)\pi_{(i,j+1)} & \forall i,j \in \mathbb{N} \mid \begin{matrix} 1 \le i \le S-1 \\ 1 \le j \le S-1 \end{matrix} \mid i+j<S \\
(i\mu_1+\lambda_1)\pi_{(i,0)} = \mu_1(i+1)\pi_{(i+1,0)} + \lambda_1\pi_{(i-1,0)} & \forall i \in \mathbb{N} \mid S+1 \le i \le N-1 \\
N\mu_1\pi_{(N,0)} = \lambda_1\pi_{(N-1,0)} & \\
\sum \pi_{(i,j)} = 1 & \forall i,j \in \mathbb{N}_0
\end{cases}
$$

2.2.1 Probabilities computing

although... To properly analyse this system, we need do compute some useful probabilities.

Similarly to the previous case we need to now following probabilities:

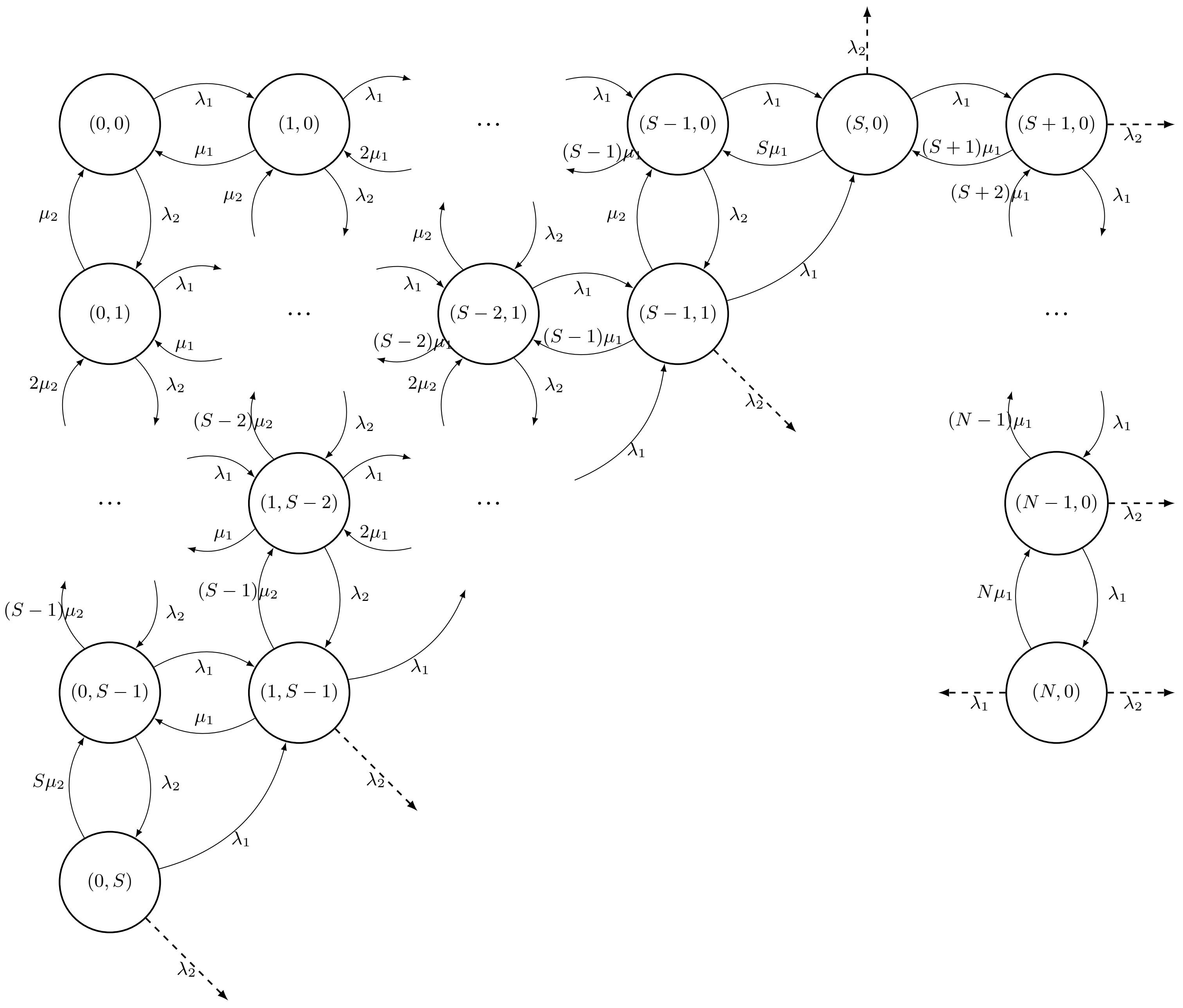
Table 3: Lista dei file del malware FASTCash

$\Pi_{\text{SendToCloudlet}}(k)$	Probability that an arriving job of class k on controller has to be forwarded to cloudlet
$\Pi_{\text{SendToCloud}}(k)$	Probability that an arriving job of class k on controller has to be forwarded to cloud
$\Pi_{\text{Class2JobInterruption}}(k)$	Probability that a job of class 2 running on cloudlet has to be interrupted and forwarded to cloud due of arriving class 1 job on cloudlet.

Differently from the previous case, owing to the use of a different access control algorithm, jobs of different classes have different probability according to which an arriving job on controller has to be send to cloud. From CTMC analysis, we can easily understand that:

$$
\begin{aligned}
\Pi_{\text{SendToCloud}}(1) &= P\{\text{An arrival class 1 job on controller sees } N \text{ class 1 jobs in cloudlet}\} \\
&= P\{n_1 = N\} \\
&= \pi(N,0)
\end{aligned}
\tag{20}
$$

Figure 2: Cloudlet system component modeled using a CTMC



$$
\begin{aligned}
\Pi_{\text{SendToCloud}}(2) &= P\{\text{An arrival class 2 job on controller sees that number of jobs in cloudlet exceed or}\\
&= P\{n_1 + n_2 \geq N\}\\
&= \sum_{\substack{n_1, n_2 = 0 \\ n_1 + n_2 = S}}^{S} \pi(n_1, n_2) + \sum_{n_1 = S+1}^{N} \pi(n_1, 0)\\
&= \sum_{\substack{0 \leq n_1 \leq N \\ 0 \leq n_2 \leq S \\ n_1 + n_2 \geq S}} \pi(n_1, n_2)
\end{aligned} \tag{21}
$$