Wireless Networks with Retrials and Heterogeneous Servers : Comparing Random Server and Fastest Free Server Disciplines

Nawel Gharbi
Computer Science Department
University of Sciences and Technology, USTHB
Algiers, Algeria
Email: ngharbi@wissal.dz

Leila Charabi
Computer Science Department
University of Sciences and Technology, USTHB
Algiers, Algeria
Email: leila.charabi@gmail.com

***Abstract*—This paper proposes an algorithmic approach based on Generalized Stochastic Petri Nets, for modeling and analyzing finite-source wireless networks with retrial phenomenon and two servers classes. The particularity of this approach is the direct computing of the infinitesimal generator of the proposed Generalized Stochastic Petri Net without generating neither the reachability graph nor the underlying Markov chain. Furthermore, we assume in this model that servers of one class are faster than those of the second one. In *Random Server* policy, customers requests are assigned randomly to free servers of both classes. The disadvantage of this policy is the increased response time when fast servers are free and requests are assigned (randomly) to slow ones. Hence, this paper aims at presenting another service policy, where priority is given to faster free servers. This policy is called the *Fastest Free Server* policy. Moreover, we compare through numerical examples, Random Service policy to Fastest Free Server one, by developing formulas of the main stationary performance indices of the network. We compare also these two policies to *Averaged Random* case, where the same global number of servers is assumed, but all homogeneous with the average service rate. We show that Fastest Free Server discipline gives better results than both Averaged Random case and Random Server discipline.**

***Keywords*-Wireless networks; Retrial phenomenon; Heterogeneous servers; Performance indices; Service disciplines.**

I. INTRODUCTION

Models with retrial phenomenon are characterized by the feature that a customer finding all servers busy or unavailable, is obliged to leave the service area, but he repeats his request after some random period of time. As we have seen in [1], these models play an important role in cellular mobile networks [5], [13], [14] and wireless sensor networks [15]. Significant references reveal the non-negligible impact of repeated calls, which arise due to a blocking in a system with limited capacity resources or are due to impatience of customers. For a systematic account of the fundamental methods and results on this topic, we refer the readers to [3], [4], [9].

Most studies on retrial models with finite source, assume that the service station consists of homogeneous (identical) servers. However, retrial models with heterogeneous servers arise in various practical areas as telecommunications and cellular mobile networks. In fact, heterogeneous models are far more difficult for mathematical analysis than models with homogeneous servers, and explicit results are available only in few special cases and almost all studies are investigated only by means of queueing theory. In fact, we have found in the literature, only the few papers of Efrosinin and Sztrik [7], [8], [11], [12], where heterogeneous servers case was considered using retrial queueing model, and the paper [10] where we have proposed the modeling and the analysis of multiclass retrial systems by means of colored generalized stochastic Petri nets.

From a modeling point of view, and compared to retrial queueing models, Generalized Stochastic Petri Nets (GSPNs) [2], [6] are a high-level graphical formalism, which allows an easier description of the behavior of complex retrial networks, and it has shown to be a very effective mathematical model. Moreover, from the GSPN model, a Continuous Time Markov Chain (CTMC) can be automatically derived for the performance analysis. However, generating the Markov chain from the GSPN and solving it, still require large storage space and long execution time, since the state space increases as a function of the customers source size and servers number. So, for real retrial networks, the corresponding models have a huge state space.

Hence, using the GSPN model as a support, we have proposed in [1] an algorithmic approach for analyzing performance of finite-source retrial networks with two servers classes, servers of one class are supposed to be faster than those of the second one. In fact, the proposed approach allows to compute directly the infinitesimal generator without generating the reachability graph nor the underlying Markov chain. In addition, we developed the formulas of the main stationary performance indices, as a function of the number of servers of each class, the size of the customers source, the stationary probabilities and independently of the reachability set markings. Nevertheless, the unique service policy we have employed in [1] was the *Random Service* policy, where the server to which a request is assigned is chosen randomly among all idle servers, in both classes. The inconvenience

of this discipline is that it can assign a customer's request to a slow server, while there is at least one fast server free.

In the present paper, we extend our idea by considering another service policy, that can improve performance by giving priority to fastest servers class, i.e., new requests are assigned to a server in the slowest class, only if all servers in the fastest class are busy, this policy is called *Fastest Free Server* policy. Moreover, using some numerical examples, we make a comparison between these two policies, with the *Averaged Random case*, where we suppose the same number of servers (in both classes), but all homogeneous with the average service rate.

This paper is organized as follows. In Section II, we describe the basic model of finite-source retrial networks with heterogeneous servers. In Section III, the basic notions of GSPNs are reviewed. Section IV presents the GSPN models describing retrial networks with heterogeneous servers for each policy namely, Random Server and Fastest Free Server. Next, the proposed stochastic analysis approach is detailed in Section V. The computational formulas for evaluating exact performance indices are derived in Section VI. Next, based on numerical examples, we validate the proposed approach, we discuss the effect of some network parameters on the main performance indices, as the mean response time and the blocking probability, and we compare the two service disciplines, in Section VII, finally, we provide a conclusion.

II. The basic model

We consider retrial networks with finite source (population) of customers of size L and a service station that consists of heterogeneous servers. Each customer can be in one of the following states: free, under service or in orbit at any time. The input stream of primary calls is the so called quasi-random input. The probability that any particular customer generates a primary request for service in any interval $(t, t+dt)$ is $\lambda dt + o(dt)$ as $dt \to 0$ if the customer is free at time t, and zero if the customer is being served or in orbit at time t.

The servers are partitioned in two classes: Class C_1 and Class C_2,where the servers of a given class have the same parameters. Each class C_j $(1 \leq j \leq 2)$ contains S_j identical and parallel servers. There are two possible states for a server: idle or busy (on service). If there is an idle server at the moment a customer request arrives, the service starts immediately. The customer becomes "*under service*" and the server becomes "*busy*". Service times are independent identically-distributed random variables, whose distribution is exponential with parameter μ_1 if a server of class C_1 is selected and μ_2 for servers of class C_2.

Each customer request must be served by one and only one server. Hence, we consider two service disciplines:

- the *Random Server discipline*, which means that, the server to which a request is assigned is chosen randomly among all idle servers, whatever their class.
- and the *Fastest Free Server discipline*, in which the request is affected randomly to an idle server of C_1 class (supposed to be the fastest), if at least one server is free, otherwise, it is assigned to a C_2 class server.

After service completion, the customer becomes free, so it can generate new primary calls, and the server becomes idle again. Otherwise, if all servers of the two classes are busy at the arrival of a request, the customer joins the orbit and starts generating a flow of repeated calls exponentially distributed with rate ν, until he finds one free server. We assume that all customers are persistent in the sense that they keep making retrials until they receive their requested service and that the total servers number $S_1 + S_2$ is smaller than the size of customers source L. Otherwise, the problem is not interesting (no customer in orbit at all).

As usual, we assume that the arrival, service and inter-retrial times are mutually independent of each other.

III. An overview of Generalized Stochastic Petri nets

A GSPN [2], [6] is a directed graph that consists of two kinds of nodes, called places and transitions that are partitioned into two different classes: timed and immediate transitions. Timed transitions describe the execution of time consuming activities and fire with an exponentially distributed delay. Immediate transitions, which fire in zero time once they are enabled, model logic activities, like synchronization, and they have priority over timed transitions.

The system state is described by means of markings. A marking is a mapping from P to $\mathbb{N}$, which gives the number of tokens in each place after each transition firing. A transition is said to be enabled in a given marking, if and only if each of its normal input places contains at least as many tokens as the multiplicity of the connecting arc, and each of its inhibitor input places contains fewer tokens than the multiplicity of the corresponding inhibitor arc.

The set of all markings reachable from initial marking M_0 is called the *reachability set*. The *reachability graph* is the associated graph obtained by representing each marking by a vertex and placing a directed edge from vertex M_i to vertex M_j, if marking M_j can be obtained by the firing of some transition enabled in marking M_i.

Markings enabling no immediate transitions are called *tangible markings*. In this case, one of the enabled timed transitions can fire next. Markings in which at least one immediate transition is enabled, are called *vanishing markings* and are passed through in zero time. In this case, only the enabled immediate transitions are allowed to fire. Since the process spends zero time in the vanishing markings, they do not contribute to the dynamic behavior of the system, so, they are eliminated from the reachability graph by merging them with their successor tangible markings. This elimination of vanishing markings results in a *tangible reachability graph*, which is isomorphic to a continuous time

Markov chain (CTMC) [2]. Hence, the states of the CTMC are the markings in the tangible reachability graph, and the state transition rates are the exponential firing rates of timed transitions in the GSPN.

The solution of this CTMC at steady-state (if the system is ergodic) is the stationary probability vector π, which is the solution of the linear system of equations:

$$\begin{cases} \pi.Q = 0 \\ \sum_i \pi_i = 1 \end{cases}$$

where π denotes the steady-state probability that the process is in state M_i and Q is the infinitesimal generator corresponding to the CTMC. Having the probability vector π, we can easily compute several stationary performance indices of the system, like the mean number of tokens in a place, the mean throughput of a transition, and the probability that an event occurs.

The process of generating stationary performance indices from the GSPN model is summarized in Figure 1.

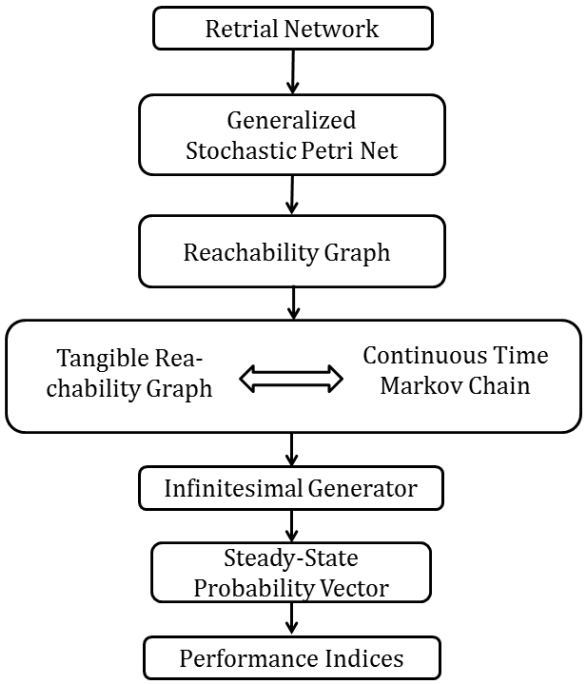

Figure 1. Steps of performance evaluation of retrial networks using GSPN formalism.

However, when modeling real retrial networks with an important customers source size and servers number, generating the GSPN, its reachability graph and then, the tangible reachability graph and the underlying CTMC, require a huge storage space and a very long execution time, since the state space increases as a function of the customers source size and servers number.

IV. GSPN MODEL OF RETRIAL NETWORKS WITH HETEROGENEOUS SERVERS

In the following, we present the GSPN model describing finite-source retrial systems with two servers classes, using Fastest Free Server policy. The detailed model corresponding to Random Server one is given in [1], only the scheme of the corresponding GSPN is given here (Figure 2).

We assume that servers of class C_1 are faster than those of class C_2. The flexibility of GSPN allows us to easily obtain Fastest Free Server policy model, which is depicted in Figure 3.

In this model, place *Cus_Free* represents the free customers, *Orbit* contains the customers waiting for the service, *Ser_Idle1* and *Ser_Idle2* indicate respectively the number of free servers of class C_1 and class C_2, while *Cus_Serv1* and *Cus_Serv2* model the busy servers of both classes.

The arrival of a primary call causes the firing of the transition *Arrival*, which firing rate is marking dependent and equals $\lambda.M(Cus\_Free)$ (*infinite service semantics*), which is represented by the symbol $\#$ placed next to transition, because all free customers are able to generate calls, independently of each other. The place *Choice* is then marked, at this moment, if at least one server in class C_1 is free (The place *Ser_Idle_1* is marked), it will serve the customer's request (firing of transition *Begin_Serv_1*). Otherwise, if place *Ser_Idle_1* is empty and place *Ser_Idle_2* contains at least one token (i.e., at least one C_2 server is idle), the transition *Begin_Serv_2* is enabled, and the request is assigned to a server of class C_2.

In case no server is available at the arrival moment of the primary call (neither in C_1 nor in C_2), the immediate transition *Go_Orbit* is enabled, and a token is put into place *Orbit*, which means that the customer asking for service joins the orbit, it starts generating a flow of repeated calls distributed exponentially with rate ν, as shown in transition *Retrial*.

The firing of the transition *Retrial* corresponds to the generation of a repeated call from a customer in orbit. This transition has infinite servers semantics, since all customers in orbit can trigger repeated calls independently.

By the end of a customer service under a server of class C_1 (C_2 respectively), the timed transition *Serv_End1* (*Serv_End2* respectively) fires. As several servers may be busy at the same time, the semantics of these two transitions is ∞-*servers* to allow modeling parallel services. After completion of service, the customer returns to free state (one token is added to place Cus_Free) and the server becomes

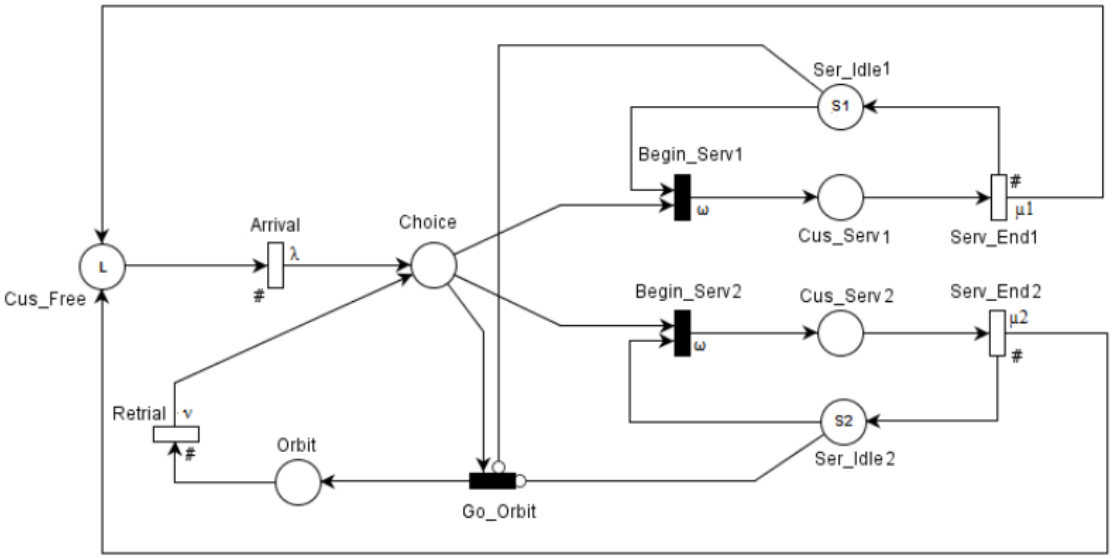

Figure 2. GSPN Model of finite-source retrial networks with two servers classes and Random server discipline.

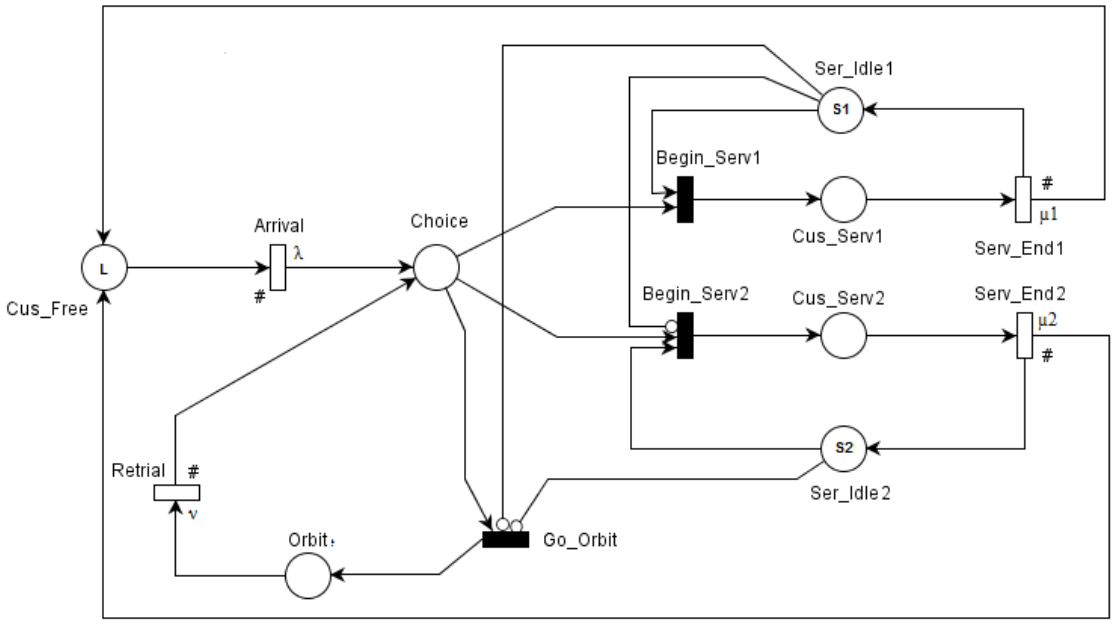

Figure 3. GSPN Model for finite-source retrial networks with two servers classes and Fastest Free Server discipline.

available (one token is put in place *Ser_Idle1* or *Ser_Idle2*, according to the server class).

V. Stochastic analysis

As it is shown in the end of Section III, the disadvantage of calculating performance indices of a retrial network using GSPN formalism was the increase of the state space as a function of customers source size and number of servers when generating the underlying CTMC. In order to overcome this problem, this paper aims to avoid these steps by designing *an algorithm* that computes directly the infinitesimal generator as a function of network parameters, without generating neither the reachability graphs nor the underlying CTMC, as to be shown in Figure 4.

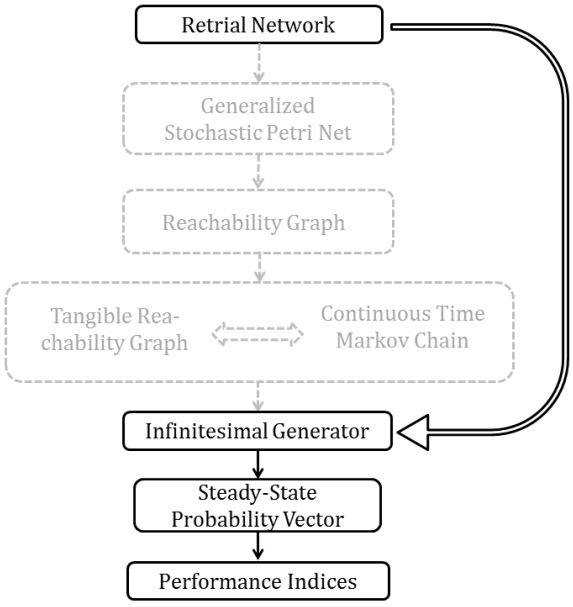

Figure 4. Our approach steps of retrial networks performance evaluation.

This section describes in detail, how to derive this algorithm [1]. We show that the discussion is the same for both service disciplines, whereas, CTMC we obtain for Random Service policy is different from Fastest Free Server policy's one. Consequently, the algorithms that generate the infinitesimal generator are different.

Initially, the orbit is empty, all customers are free and all servers are available. Thus, the initial marking can be expressed in this form:

$$\begin{aligned} M_0 &= \{M(Cus\_Free), M(Choice), M(Orbit), \\ &\quad M(Ser\_Idle1), M(Cus\_Serv1), M(Ser\_Idle2), \\ &\quad M(Cus\_Serv2)\} \\ &= \{L, 0, 0, S_1, 0, S_2, 0\} \end{aligned}$$

Whatever the values of L, S_1 and S_2, the conservation of the number of customers and servers of the two classes, gives the following equations:

$$\begin{cases} M(Ser\_Idle1) + M(Cus\_Serv1) = S_1 \\ M(Ser\_Idle2) + M(Cus\_Serv2) = S_2 \\ M(Cus\_Free) + M(Cus\_Serv1) \\ +M(Cus\_Serv2) + M(Orbit) = L \end{cases} \quad (1)$$

Observing these three equations, we note that the system state at steady-state can be described by means of three variables (i, j, k), which we call a *micro-state*, where:

- i represents the number of customers being served by servers of class C_1 (in place Cus_Serv1);
- j represents the number of customers being served by servers of class C_2 (in place Cus_Serv2);
- and k is the number of customers in orbit (in place $Orbit$).

Hence, having the micro-state (i, j, k), the markings of all places can be obtained, since

$$\begin{cases} M(Ser\_Idle1) = S_1 - i \\ M(Ser\_Idle2) = S_2 - j \\ M(Cus\_Free) = L - (i + j + k) \end{cases} \quad (2)$$

On the other hand, applying (1), we can deduce:

$$\begin{cases} 0 \leq i \leq S_1 \\ 0 \leq j \leq S_2 \\ 0 \leq k \leq L - (S_1 + S_2) \end{cases} \quad (3)$$

In fact, we introduce the concept of micro-state as a compact state description derived by the analysis of P-invariants of the model, so that it is always possible to define a one-to-one correspondence between the micro-states and the ordinary states of the classical approach.

The corresponding CTMC contains n micro-states corresponding to the accessible tangible markings, where n equals

$$n = (S_1 + 1) \cdot (S_2 + 1) \cdot (L + 1 - S) \text{ and } S = S_1 + S_2 \quad (4)$$

Figure 5 describes the CTMC corresponding to the Random Service policy model, while The CTMC resultant from the Fastest Free Server GSPN is given in Figure 6.

Thus, the corresponding infinitesimal generator Q is a $n \times n$ matrix, defined by:

$$\begin{cases} Q[(i,j,k),(x,y,z)] = \theta[(i,j,k),(x,y,z)] \\ Q[(i,j,k),(i,j,k)] = -\sum_{(l,m,n)\neq(i,j,k)} \theta[(i,j,k),(l,m,n)] \end{cases} \quad (5)$$

where $\theta[(i,j,k),(x,y,z)]$ is the transition rate from state (i,j,k) to state (x,y,z).

By analyzing the micro-states and the transition rates of each CTMC, we obtain the following rates, for the Random Service discipline :

- $[0 \leq i \leq S_1 - 1, 0 \leq j \leq S_2 - 1] : (i,j,k) \xrightarrow{\frac{1}{2}(L-i-j-k)\lambda} (i+1,j,k)$ and $(i,j,k) \xrightarrow{\frac{1}{2}(L-i-j-k)\lambda} (i,j+1,k)$
- $[0 \leq i \leq S_1 - 1] : (i,S_2,k) \xrightarrow{(L-i-S_2-k)\lambda} (i+1,S_2,k)$,
- $[0 \leq j \leq S_2 - 1] : (S_1,j,k) \xrightarrow{(L-S_1-j-k)\lambda} (S_1,j+1,k)$,
- $[0 \leq k < L - (S_1+S_2)] : (S_1,S_2,k) \xrightarrow{(L-S-k)\lambda} (S_1,S_2,k+1)$,
- $[i > 0] : (i,j,k) \xrightarrow{i\mu_1} (i-1,j,k)$,
- $[j > 0] : (i,j,k) \xrightarrow{j\mu_2} (i,j-1,k)$,
- $[0 \leq i \leq S_1 - 1, 0 \leq j \leq S_2 - 1, k > 0] : (i,j,k) \xrightarrow{\frac{1}{2}k\nu} (i+1,j,k-1)$ and $(i,j,k) \xrightarrow{\frac{1}{2}k\nu} (i,j+1,k-1)$,
- $[0 \leq i \leq S_1 - 1, k > 0] : (i,S_2,k) \xrightarrow{k\nu} (i+1,S_2,k-1)$,
- $[0 \leq j \leq S_2 - 1, k > 0] : (S_1,j,k) \xrightarrow{k\nu} (S_1,j+1,k-1)$,

As a consequence, the infinitesimal generator can be automatically calculated by means of Algorithm 1 given below.

In the same manner, rates $\theta[(i,j,k)(x,y,z)]$ of the Fastest Free Server discipline are given by :

- $[0 \leq i < S_1] : (i,j,k) \xrightarrow{(L-i-j-k)\lambda} (i+1,j,k)$,
- $[0 \leq j < S_2] : (S_1,j,k) \xrightarrow{(L-S_1-j-k)\lambda} (S_1,j+1,k)$,
- $[0 \leq k < L - S] : (S_1,S_2,k) \xrightarrow{(L-S-k)\lambda} (S_1,S_2,k+1)$,
- $[0 < i \leq S_1] : (i,j,k) \xrightarrow{i\mu_1} (i-1,j,k)$,
- $[0 < j \leq S_2] : (i,j,k) \xrightarrow{j\mu_2} (i,j-1,k)$,

Algorithm 1 Infinitesimal Generator Construction - Random Server Policy

```
for k ← 0, L − S do
    for i ← 0, S1 − 1 do
        for j ← 0, S2 − 1 do
            Q[(i,j,k),(i+1,j,k)] ← 1/2(L−i−j−k)λ
            Q[(i,j,k),(i,j+1,k)] ← 1/2(L−i−j−k)λ
            Q[(S1,j,k),(S1,j+1,k)] ← (L−S1−j−k)λ
        end for
        Q[(i,S2,k),(i+1,S2,k)] ← (L−i−S2−k)λ
    end for
end for
for k ← 0, L − S − 1 do
    Q[(S1,S2,k),(S1,S2,k+1)] ← (L − S − k)λ
end for
for k ← 0, L − S do
    for i ← 1, S1 do
        for j ← 0, S2 do
            Q[(i,j,k),(i−1,j,k)] ← iμ1
        end for
    end for
    for j ← 1, S2 do
        for i ← 0, S1 do
            Q[(i,j,k),(i,j−1,k)] ← jμ2
        end for
    end for
end for
for k ← 1, L − S do
    for i ← 0, S1 − 1 do
        for j ← 0, S2 − 1 do
            Q[(i,j,k),(i+1,j,k−1)] ← 1/2.kν
            Q[(i,j,k),(i,j+1,k−1)] ← 1/2.kν
        end for
        Q[(i,S2,k),(i+1,S2,k−1)] ← kν
    end for
    for j ← 0, S2 − 1 do
        Q[(S1,j,k),(S1,j+1,k)] ← kν
    end for
end for
```

- $[0 \leq i < S_1, 0 < k \leq L-S] : (i,j,k) \xrightarrow{k\nu} (i+1,j,k-1)$,
- $[0 \leq j < S_2, 0 < k \leq L - S)] : (S_1,j,k) \xrightarrow{k\nu} (S_1,j+1,k-1)$,

And the algorithm that generates the infinitesimal generator is given in Algorithm 2 (Fastest Free Server policy).

VI. Performance measures

The aim of this section is to derive the formulas of the most important stationary performance indices. As the

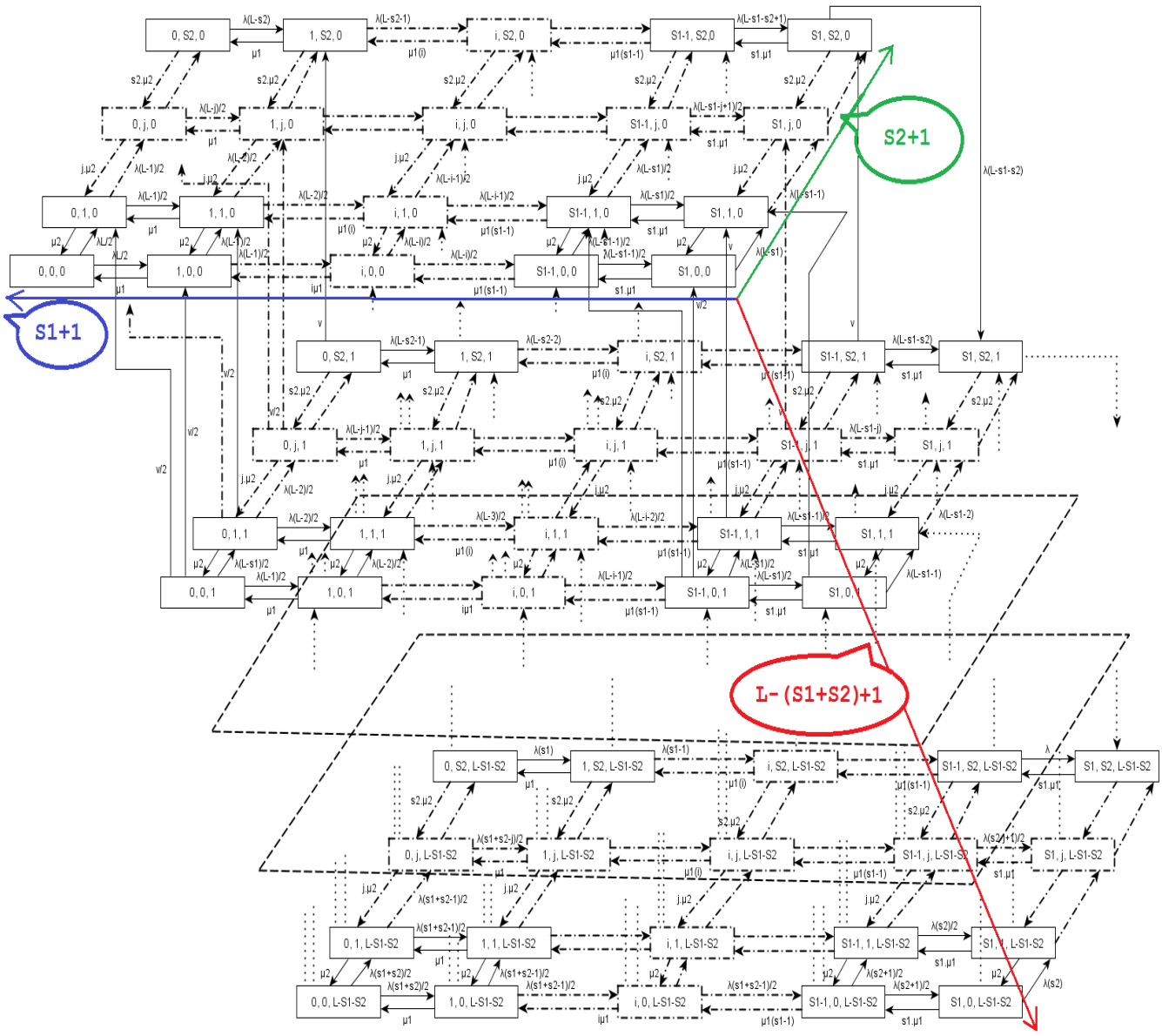

Figure 5. The CTMC describing finite-source retrial networks with two servers classes and Random Server discipline.

proposed models are bounded and the initial marking is a home state, the underlying process is ergodic. Hence, the steady-state solution exists and is unique. The infinitesimal generators Q corresponding to the proposed GSPN models can be obtained automatically by applying the above algorithms. Then, the steady-state probability vector π can be computed by solving the linear system of equations:

$$\begin{cases} \pi.Q = 0 \\ \sum_i \pi_i = 1 \end{cases} \quad (6)$$

where π_i denotes the steady-state probability that the process is in state M_i.

Having the probability distribution π, we can derive several exact stationary performance measures of finite-source retrial networks with two classes of servers, applying the formulas given below, which are based essentially on Equation (2) and the definition of the three variables i, j, and k given in Section V. In following, $M_i(p)$ indicates the number of tokens in place p in marking M_i, A is the set of reachable tangible markings, and $A(t)$ is the set of tangible markings reachable by transition t.

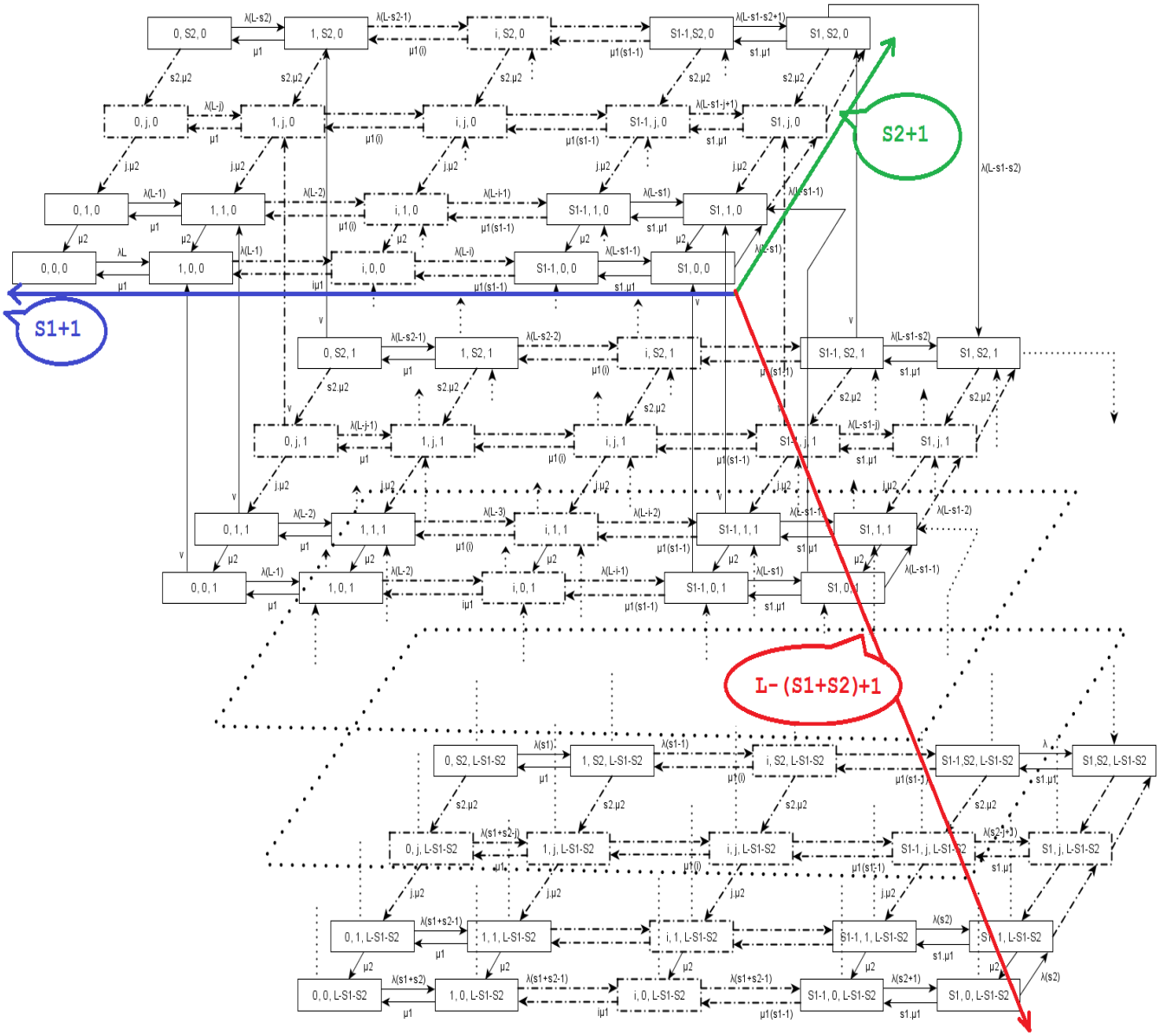

Figure 6. The CTMC describing finite-source retrial networks with two servers classes and Fastest Free Server discipline.

- `Mean number of free customers:` It corresponds to the mean number of tokens in place *Cus_Free*,

$$n_{CusFree} = \sum_{i:M_i \in A} M_i(Cus\_Free).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} (L-i-j-k).\pi_{i,j,k}$$

- `Mean number of customers in the orbit:` This corresponds to the mean number of tokens in *Orbit*,

$$n_{Orb} = \sum_{i:M_i \in A} M_i(Orbit).\pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0}^{S_2} k.\pi_{i,j,k}$$

- `Mean number of busy servers of class`

Algorithm 2 Infinitesimal Generator Construction - Fastest Free Server Policy

for $k \leftarrow 0, L-S$ **do**
  for $i \leftarrow 0, S1-1$ **do**
    for $j \leftarrow 0, S2$ **do**
      $Q[(i,j,k),(i+1,j,k)] \leftarrow (L-i-j-k)\lambda$
    end for
  end for
  for $j \leftarrow 0, S2-1$ **do**
    $Q[(S1,j,k),(S1,j+1,k)] \leftarrow (L-S1-j-k)\lambda$
  end for
end for
for $k \leftarrow 0, L-S-1$ **do**
  $Q[(S1,S2,k),(S1,S2,k+1)] \leftarrow (L-S-k)\lambda$
end for
for $k \leftarrow 0, L-S$ **do**
  for $i \leftarrow 1, S1$ **do**
    for $j \leftarrow 0, S2$ **do**
      $Q[(i,j,k),(i-1,j,k)] \leftarrow i\mu_1$
    end for
  end for
  for $j \leftarrow 1, S2$ **do**
    for $i \leftarrow 0, S1$ **do**
      $Q[(i,j,k),(i,j-1,k)] \leftarrow j\mu_2$
    end for
  end for
end for
for $k \leftarrow 1, L-S$ **do**
  for $i \leftarrow 0, S1-1$ **do**
    for $j \leftarrow 0, S2$ **do**
      $Q[(i,j,k),(i+1,j,k-1)] \leftarrow k\nu$
    end for
  end for
  for $j \leftarrow 0, S2-1$ **do**
    $Q[(S1,j,k),(S1,j+1,k-1)] \leftarrow k\nu$
  end for
end for

C_1: Note that this is also the mean number of customers under service by Class C_1, it corresponds to the mean number of tokens in place *Cus_Serv1*,

$$\begin{aligned} n_{busyC_1} &= \sum_{i:M_i \in A} M_i(Cus\_Serv1).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} i.\pi_{i,j,k} \end{aligned}$$

- `Mean number of busy servers of class` C_2: It is also the mean number of customers in service by Class C_2, and it represents the mean number of tokens in *Cus_Serv2*,

$$\begin{aligned} n_{busyC_2} &= \sum_{i:M_i \in A} M_i(Cus\_Serv2).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} j.\pi_{i,j,k} \end{aligned}$$

- `Mean number of busy servers:`

$$\begin{aligned} n_{busy} &= n_{busyC_1} + n_{busyC_2} \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}(i+j).\pi_{i,j,k} \end{aligned}$$

- `Mean number of customers in the system`: Which is the total number of the mean number of customers in the orbit and those under service (by C_1 and C_2),

$$\begin{aligned} n &= n_{Orb} + n_{busy} \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}(i+j+k).\pi_{i,j,k} \end{aligned}$$

- `Mean number of free servers of class` C_1: This represents the mean number of tokens in place *Ser_Idle1*,

$$\begin{aligned} n_{FreeC_1} &= \sum_{i:M_i \in A} M_i(Ser\_Idle1).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}(S_1-i).\pi_{i,j,k} \\ &= S_1 - n_{busyC_1} \end{aligned}$$

- `Mean number of free servers of class` C_2: This represents the mean number of tokens in place *Ser_Idle2*,

$$\begin{aligned} n_{FreeC_2} &= \sum_{i:M_i \in A} M_i(Ser\_Idle2).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2}(S_2-j).\pi_{i,j,k} \\ &= S_2 - n_{busyC_2} \end{aligned}$$

- `Mean number of free servers (of both classes):`

$$n_{Free} = n_{FreeC_1} + n_{FreeC_2} = S - n_{busy}$$

- `Effective customer arrival rate:` This represents the throughput of the transition *Arrival*,

$$\begin{aligned}\bar{\lambda} &= \sum_{i:M_i \in A(Arrival)} \lambda.M_i(Cus\_Free).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \lambda.(L-i-j-k).\pi_{i,j,k} \\ &= \lambda.n_{CusFree}\end{aligned}$$

- `Effective customer retrial rate:` It corresponds to the throughput of *Retrial* transition,

$$\begin{aligned}\bar{\nu} &= \sum_{i:M_i \in A(Retrial)} \nu.M_i(Orbit).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \nu.k.\pi_{i,j,k} \\ &= \nu.n_{Orb}\end{aligned}$$

- `Mean rate of` C_1 `service:` This corresponds to the throughput of the transition *Serv_End1*,

$$\begin{aligned}\bar{\mu_1} &= \sum_{i:M_i \in A(Serv\_End1)} \mu_1.M_i(Cus\_Serv1).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \mu_1.i.\pi_{i,j,k} \\ &= \mu_1.n_{busyC_1}\end{aligned}$$

- `Mean rate of` C_2 `service:` This corresponds to the throughput of *Serv_End2*,

$$\begin{aligned}\bar{\mu_2} &= \sum_{i:M_i \in A(Serv\_End2)} \mu_2.M_i(Cus\_Serv2).\pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2} \mu_2.j.\pi_{i,j,k} \\ &= \mu_2.n_{busyC_2}\end{aligned}$$

- `Total mean rate service:`

$$\bar{\mu} = \bar{\mu_1} + \bar{\mu_2}$$

- `Availability of` s `servers of class` C_1 $(1 \leq s \leq S_1)$`:` It's the probability that at least s servers of class C_1 are available

$$A_{sC_1} = \sum_{i:M_i(Ser\_Idle1) \geq s} \pi_i = \sum_{k=0}^{L-S}\sum_{i=0}^{S_1-s}\sum_{j=0}^{S_2} \pi_{i,j,k}$$

- `Availability of` s `servers of class` C_2 $(1 \leq s \leq S_2)$`:` It's the probability that at least s servers of class C_2 are available

$$A_{sC_2} = \sum_{i:M_i(Ser\_Idle2) \geq s} \pi_i = \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0}^{S_2-s} \pi_{i,j,k}$$

- `Availability of` s `servers in the system (among both classes):`

$$\begin{aligned}A_s &= \sum_{i:M_i(Ser\_Idle1)+M_i(Ser\_Idle2) \geq s} \pi_i \\ &= \sum_{k=0}^{L-S}\sum_{i=0}^{S_1}\sum_{j=0,i+j\leq S-s}^{S_2} \pi_{i,j,k}\end{aligned}$$

- `Utilization of at least` s `servers of the class` C_1`:` This corresponds to the probability that at least s servers of class C_1 are busy

$$U_{sC_1} = \sum_{i:M_i(Cus\_Serv1) \geq s} \pi_i = \sum_{k=0}^{L-S}\sum_{i=s}^{S_1}\sum_{j=0}^{S_2} \pi_{i,j,k}$$

- `Utilization of` s `servers at least, of the class` C_2`:` This corresponds to the probability that at least s servers of class C_2 are busy

$$U_{sC_2} = \sum_{i:M_i(Cus\_Serv2)\geq s} \pi_i = \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=s}^{S_2} \pi_{i,j,k}$$

- Utilization of s servers at least in the system (among the two classes):

$$\begin{aligned} U_s &= \sum_{i:M_i(Cus\_Serv1)+M_i(Cus\_Serv2)\geq s} \pi_i \\ &= \sum_{k=0}^{L-S} \sum_{i=0}^{S_1} \sum_{j=0,i+j\geq s}^{S_2} \pi_{i,j,k} \end{aligned}$$

- The blocking probability of a primary customer:

$$B_p = \frac{\sum_{k=0}^{L-S} (L-k-S).\lambda.\pi_{S_1,S_2,k}}{\bar{\lambda}}$$

- The blocking probability of a repeated call:

$$B_r = \frac{\sum_{k=1}^{L-S} k.\nu.\pi_{S_1,S_2,k}}{\bar{\nu}}$$

- The blocking probability:

$$B = B_p + B_r$$

- The mean waiting time: It's the mean period between the arrival of the customer and its service beginning. Using the Little's formula, the mean waiting time is given by :

$$\bar{W} = \frac{n_{Orb}}{\bar{\lambda}}$$

- The mean response time:

$$\bar{R} = \frac{n}{\bar{\lambda}}$$

- The mean service time:

$$\bar{S} = \bar{R} - \bar{W} = \frac{n_{busy}}{\bar{\lambda}}$$

VII. Validation and Numerical Examples

In order to test the feasibility of our approach, we developed a $C\#$ code to implement the above algorithms 1 and 2 as well as the performance indices formulas. Next, we tested it for a large number of examples. In particular, in the homogeneous case, by assuming $\mu_1 = \mu_2$, the results were validated by the Pascal program given in [9]. From Table I, we can see that both models give exactly the same results up to the sixth decimal digit.

Table I
Validation in the Homogeneous Case

	Homogeneous case	Two servers classes system
Number of servers	4	S_1= 1,S_2= 3
Size of source	20	20
Primary call generation rate	0.1	0.1
Service rate	1	μ_1= 1, μ_2= 1
retrial rate	1.2	1.2
Mean number of busy servers	1.800 748	C_1: 0.521 865 C_2: 1.278 883 Tot.: 1.800 748
Mean number of source in orbit	0.191 771	0.191 771
Mean primary call generation rate	1.800 748	1.800 748
Mean waiting time	0.106 495	0.106 495

In the following, we present sample numerical results to illustrate graphically the impact of some system parameters, namely, the primary call rate, retrial rate, and the servers number in both classes, on some performance indices, which are the mean response time and the blocking probability, in both policies cases; Random Server and Fastest Free Server. We also consider the *Averaged Random case*, where we assume the same number of servers in the network ($S = S_1 + S_2$), all homogeneous with the average service rate μ, whose formula is given by:

$$\mu = \frac{\mu_1.S_1 + \mu_2.S_2}{S_1 + S_2} \quad (7)$$

The customers requests are assigned to idle servers randomly.

The input parameters of Random and Fastest Free Server policies are collected in Table II, while those of the Averaged Random case are summarized in Table III.

Table II
INPUT NETWORK PARAMETERS

	L	S_1	S_2	λ	ν	μ_1	μ_2
Figure 7, 11	50	5	2	x axis	0.1	8	2
Figure 8, 12	50	4	2	0.5	x axis	8	2
Figure 9, 13	30	x axis	4	2	1	6	1
Figure 10, 14	30	4	x axis	2	1	6	1

Table III
INPUT NETWORK PARAMETERS FOR THE AVERAGED RANDOM CASE

	L	S	λ	ν	μ
Figure 7, 11	50	7	x axis	0.1	6.28
Figure 8, 12	50	6	0.5	x axis	6

In Figure 7, we study the primary call generation rate variation effect on the mean response time. As we can see, this latter increases with the intensity of the flow of primary calls. This is due to the increase of waiting time of customers in the orbit. Furthermore, the mean response time of Fastest Free Server discipline is always shorter than the one of Random Server discipline. The curve of the Averaged Random case is situated in the middle.

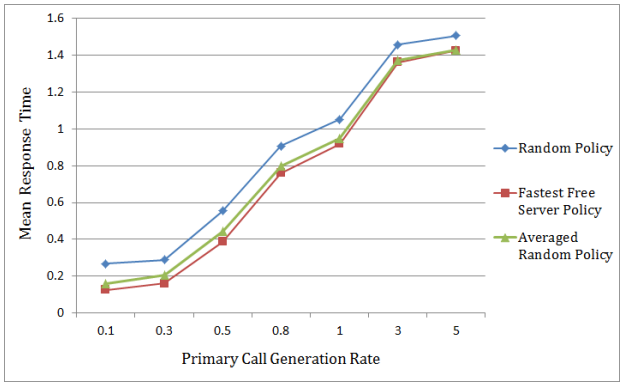

Figure 7. Mean response time versus primary call generation rate.

The Figure 8 shows the sensitivity of the mean response time to the retrial generation rate, for both service disciplines. Indeed, the response time decreases with the intensity of the flow of repeated calls, particularly when the retrial intensity is low (between 0.01 and 0.3), beyond the value 0.3, the influence becomes less significant. In addition, performance obtained with the discipline of the Fastest Free Server and Averaged Random case are always better than the one obtained by Random Server discipline.

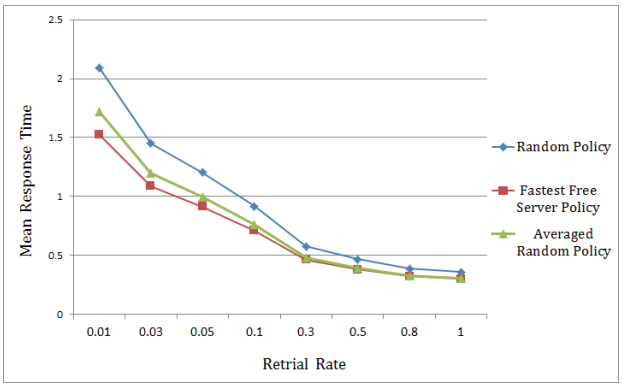

Figure 8. Mean response time versus retrial generation rate.

In Figure 9, (10 respectively), we show the influence of the number of servers of C_1 (C_2 respectively) class, on the mean response time.

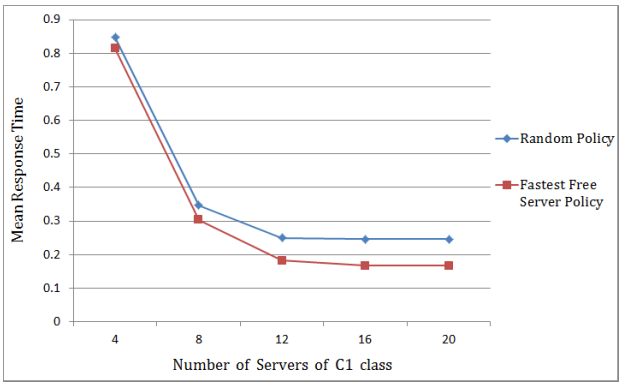

Figure 9. Mean response time versus C_1 class servers number.

We conclude that the mean response time decreases with the increase of the number of servers. However, the rate of influence of the number of servers in the C_1 class is faster than the influence due to increasing number of servers of C_2 class, because the former is faster ($\mu_1 = 6$ vs $\mu_2 = 1$). In Figure 9, the response time reached the optimum and stabilized after a certain time (number of servers = 12). Hence, it is not interesting to invest in new servers in the C_1 class.

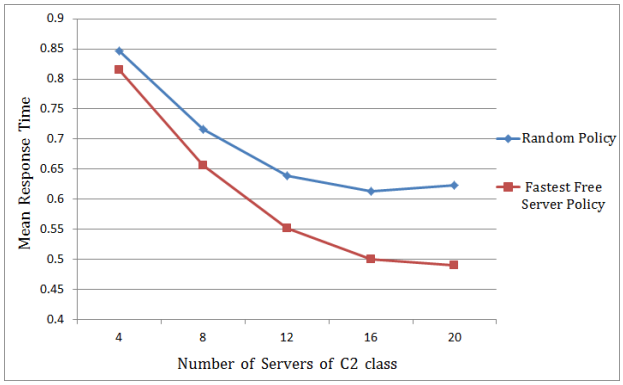

Figure 10. Mean response time versus C_2 class servers number.

In Figure 10, the surprising increase in the mean response time in Random Service policy case when having more C_2 class servers (from 16 to 20 servers) reveal another weakness of this policy. Actually, the slowest servers number becomes much greater than the fastest one, and as customers requests are assigned to servers randomly, fastest servers have less chance to catch a customer request (4 servers in C_1 class vs 20 of C_2 class), and C_2 class servers tend to take most of customers requests, this results in increasing the mean service time, and consequently the mean response time.

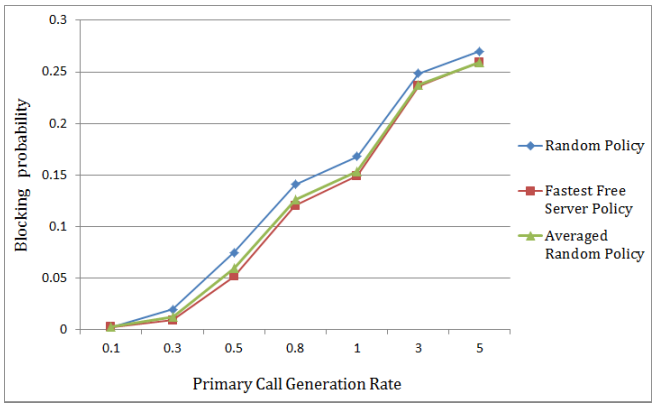

Figure 11. Blocking probability versus primary call generation rate.

As it can be seen in Figures 11 and 12, the blocking probability depends on both primary call generation rate and retrial rate, the increase of these latter involves the increase of the blocking probability in both Random Server and Fastest Free Server policies. But, as we can see on the curves, Fastest Free Server policy gives always values slightly better than those obtained in the Averaged Random case, and the results of this latter are better than those given by Random Server policy, which means that the Fastest Free server policy gives better performance.

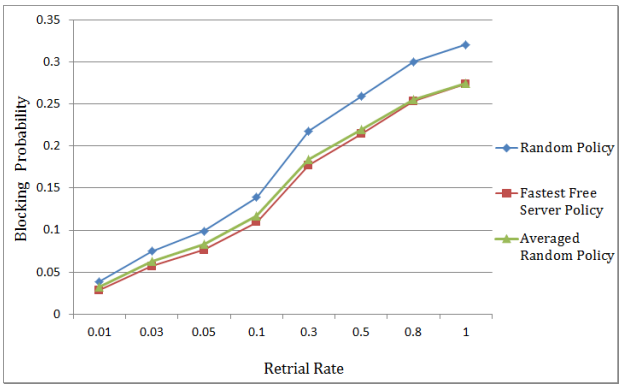

Figure 12. Blocking probability versus retrial generation rate.

In Figures 13 and 14, the blocking probability is displayed as a function of servers number in C_1 and C_2 classes respectively. As to be expected, the blocking probability is higher when having a few number of servers, and it decreases as the number of servers rises in the system, the decrease is more significant in case of C_1 class, because it is supposed to be faster. Moreover, the performance of Fastest Free Server discipline and Random Server discipline are almost the same.

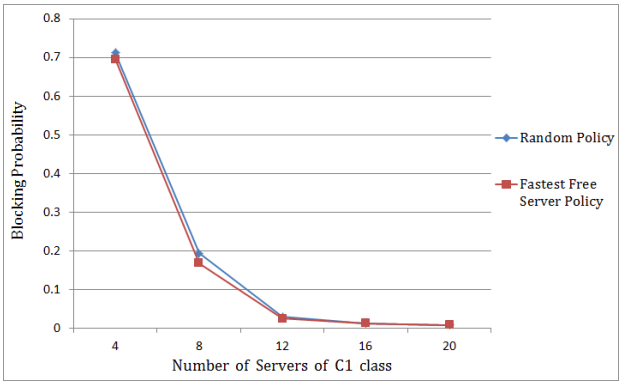

Figure 13. Blocking probability versus C_1 class servers number.

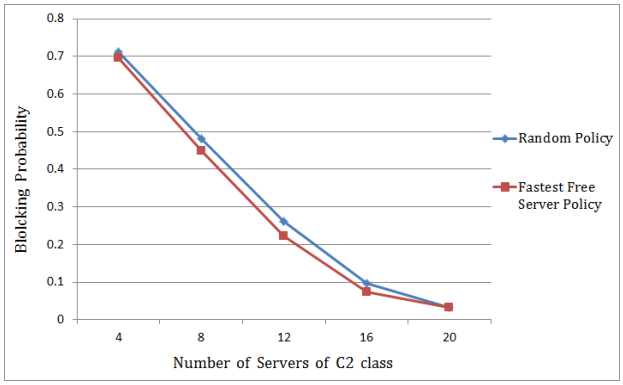

Figure 14. Blocking probability versus C_2 class servers number.

VIII. Conclusion and Future Work

In [1], we have proposed a technique based on GSPNs to analyze finite-source retrial networks with two servers classes, using Random Server discipline. In the current paper, we have extended this idea by considering another service discipline, which is the Fastest Free Server. Hence, we investigated and compared the two service disciplines.

The advantage of our approach is the automatic computation of the infinitesimal generator for both disciplines, applying the given algorithms, and without need to generate neither the reachability graph nor the underlying Markov chain. We have also developed formulas of the main stationary performance indices based on stationary probabilities and network parameters. Furthermore, we studied the effect of network parameters on performance indices, and proved through some numerical examples, that Fastest Free Server discipline gives more favorable system performance than both Random Server discipline and Averaged Random case, which is the equivalent homogeneous network with the average service rate.

References

[1] N. Gharbi and L. Charabi, *An Algorithmic Approach for Analyzing Wireless Networks with Retrials and Heterogeneous Servers*, The seventh International Conference on Wireless and Mobile Communications, ICWMC, 2011, Luxembourg, pp. 151-156, ISBN:978-1-61208-008-6.

[2] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, 1994, New York, NY, USA, John Wiley & Sons, Inc., ISBN-13: 9780471930594.

[3] J.R. Artalejo and A. Gómez-Corral, *Retrial Queueing Systems: A Computational Approach*, 2008, Berlin, Springer Berlin Heidelberg, ISBN-13: 978-3642097485.

[4] J.R. Artalejo, *Accessible bibliography on retrial queues: Progress in 2000-2009*, Mathematical and Computer Modelling, 2010, vol. 51, pp. 1071-1081.

[5] J.R. Artalejo and M.J. Lopez-Herrero, *Cellular mobile networks with repeated calls operating in random environment*, Computers & operations research, 2010, vol. 37, no. 7, pp. 1158-1166.

[6] M. Diaz, *Les réseaux de Petri - Modèles Fondamentaux*, 2001, Paris, Hermès Science Publications, ISBN-13: 978-2-7462-0250-4.

[7] D. Efrosinin and L. Breuer, *Threshold policies for controlled retrial queues with heterogeneous servers*, Annals of Operations Research, 2006, vol. 141, pp. 139-162.

[8] D. Efrosinin and J. Sztrik, *Performance Analysis of a Two-Server Heterogeneous Retrial Queue with Threshold Policy*, Quality Technology and Quantitative Management, 2011, vol. 8, no. 3, pp. 211-236.

[9] G.I. Falin and J.G.C. Templeton, *Retrial Queues*, 1997, London, Chapman and Hall, ISBN-13: 978-0412785504.

[10] N. Gharbi, C. Dutheillet, and M. Ioualalen, *Colored Stochastic Petri Nets for Modelling and Analysis of Multiclass Retrial Systems*, Mathematical and Computer Modelling, 2009, vol. 49, pp. 1436-1448.

[11] J. Roszik and J. Sztrik, *Performance analysis of finite-source retrial queues with nonreliable heterogeneous servers*, Journal of Mathematical Sciences, 2007, vol. 146, pp. 6033-6038.

[12] J. Sztrik, G. Bolch, H. De Meer, J. Roszik, and P. Wüchner, *Modeling finite-source retrial queueing systems with unreliable heterogeneous servers and different service policies using MOSEL*, Proc. of 14th Inter. Conf. on Analytical and Stochastic Modelling Techniques and Applications, ASMTA'07, 2007, Prague, Czech Republic, pp. 75-80.

[13] T. V. Do, *A new computational algorithm for retrial queues to cellular mobile systems with guard channels*, Computers & Industrial Engineering, 2010, vol. 59, pp. 865-872.

[14] P. Tran-Gia and M. Mandjes, *Modeling of customer retrial phenomenon in cellular mobile networks*, IEEE Journal on Selected Areas in Communications, 1997, vol. 15, pp. 1406-1414.

[15] P. Wüchner, J. Sztrik, and H. De Meer, *Modeling Wireless Sensor Networks Using Finite-Source Retrial Queues with Unreliable Orbit*, Proc. of the Workshop on Performance Evaluation of Computer and Communication Systems, PERFORM'2010, 2010, pp. 73-86.