Chapter 6

Importance Sampling
in rate-sharing networks

In Part I we focused on bandwidth sharing as a result of explicit scheduling in network nodes. In Part II we turn to the case that bandwidth sharing is a consequence of the end-to-end rate control by end-users. The difference between these two cases is that Part I deals with bandwidth sharing among applications on small time scales, whereas Part II considers sharing among routes on somewhat larger time scales. As mentioned in Chapter 1, the latter scenario is well represented by flow-level models, as we have to take the random nature of flows into account, as opposed to Part I, where the number of flows was assumed to be fixed.

Over the past several years the Processor-Sharing (PS) discipline has been widely used for evaluating the flow-level performance of elastic data transfers competing for bandwidth on a single bottleneck link. In a multi-link setting, bandwidth-sharing networks as considered in [135] provide a natural extension for modeling the dynamic interaction among competing elastic flows.

It is well-known that the queue length distribution in a single-server PS system with Poisson arrivals has a simple geometric distribution that only depends on the service requirement distribution through its mean. In contrast, the distribution of the number of active users in bandwidth-sharing networks with several nodes has remained generally intractable, even for exponentially distributed service requirements. The crucial result that the wide family of so-called Alpha-Fair Sharing (AFS) policies, as introduced in [140], achieve stability under the simple condition that no individual link is overloaded, was established in [23]. The family of AFS policies covers several common notions of fairness as special cases, such as max-min fairness ($\alpha \to \infty$), proportional fairness ($\alpha \to 1$), and maximum throughput ($\alpha \downarrow 0$). In [146] it has also been shown that the case $\alpha = 2$, with additional class weights set inversely proportional to the respective RTTs, provides a reasonable modeling abstraction for the bandwidth sharing realized by TCP in the Internet. We refer to Chapter 1 for more
In this chapter we consider a network operating under an AFS policy. Since the service rate allocated to a flow is restricted in practice [28], we impose class-dependent access-link rate limitations, similar as in [14]. Assuming Poisson arrivals and exponentially distributed service requirements for each class, the dynamics of the user population may be described by a Markov process.

An essential requirement of modern bandwidth-sharing networks is their capability of providing a variety of Quality-of-Service (QoS) guarantees, where QoS is usually expressed in terms of constraints on a set of performance measures, such as mean transfer delays, but also the probability that there are many flows (per class) active in the network. Typically, such a probability is required to be below some small threshold, as this prevents flows from experiencing large delays. Motivated by this, we analyze in this chapter the probability that, given that the network is in some specific state $n_0$ at time 0, the network is in some set of states $A$ after some predefined time $T$. In particular, we assume that the underlying event is rare, i.e., this probability is small. As in general no explicit expressions are known for the probability of interest, an attractive approach may be to resort to Monte-Carlo (MC) simulation. In general, the number of runs needed to obtain an estimate with predefined accuracy and confidence, is inversely proportional to the probability to be estimated [80], implying that MC simulation is impractical due to the rarity of the event under consideration. A natural method to accelerate the simulation is to use Importance Sampling (IS). The idea underlying IS is to simulate the system with a new set of input probability distributions, i.e., new interarrival and service time distributions, such that the rare event becomes more likely, and then to correct the simulation output with appropriate likelihood ratios, in order to obtain an unbiased estimate.

To obtain appropriate new input distributions we first identify the most probable path (MPP) for the event to occur. Informally speaking, given that this rare event occurs, with overwhelming probability it will happen by a path close to this MPP. For the M/M/1-PS queue the MPP is already known [161], whereas this is not the case for a general AFS network topology. We develop an approach for finding the MPP, which exploits the large deviations results of [161]. The underlying idea is that locally the flow-level dynamics of a particular class in the network can be approximated as a M/M/1-PS queue. It is noted that, in contrast to the M/M/1-PS queue where the most likely path has a linear shape, the MPP has a non-linear shape in case of a general AFS network topology. The path is then subsequently translated into new input distributions, that are such that the event under consideration occurs by realizations close to this MPP.

Extensive numerical experiments indicate that the above approach is quite effective: we are able to estimate probabilities as small as $10^{-13}$ quickly, whereas $10^{-8}$ up to $10^{-4}$ is typically the range of interest. It is emphasized that we do not prove that our IS technique is asymptotically optimal or asymptotically efficient [41].
6.1 Preliminaries

The remainder of the chapter is organized as follows. In Section 6.1 we first provide a detailed model description, discuss the use of IS, and present a key large deviations theorem. Section 6.2 deals with the M/M/1-PS queue, which is in fact a special case of our network. In Section 6.3 we derive (that is, approximate, but the approximation can be made arbitrarily close) the MPP for a rare event to occur in a general AFS network topology, by exploiting the results for the M/M/1-PS queue. Section 6.4 shows how one can translate this MPP into new input distributions that can be incorporated in an IS algorithm. The pseudocode of the IS algorithm is presented in the Appendix. Section 6.5 examines the performance of the IS algorithm for two special networks, and shows that the IS scheme performs well. Finally, Section 6.6 concludes with some final observations.

6.1 Preliminaries

In this section we first describe our queueing model. Next we discuss IS, a simulation technique designed for estimating rare event probabilities. Finally, we briefly discuss some large deviations results, which are needed in the analysis.

6.1.1 Queueing model

We consider a network consisting of \( L \) nodes, where node \( j \) has capacity \( c_j \), \( j = 1, \ldots, L \). There are \( M \) classes of users in the network, where each class corresponds to a specific route in the network. We assume that class-\( i \) users arrive according to a Poisson process of rate \( \lambda_i \), and have independent and exponentially distributed service requirements with mean \( \mu_i^{-1} \), \( i = 1, \ldots, M \). The traffic load of class \( i \) is then \( \rho_i = \lambda_i / \mu_i \), \( i = 1, \ldots, M \). The arrival processes and service requirements are all assumed to be independent. If a user requires service at multiple nodes, then it must be served at all nodes simultaneously. Let \( S(j) \) denote the set of classes that require service at node \( j \), \( j = 1, \ldots, L \). Finally, let \( N(t) = (N_1(t), \ldots, N_M(t)) \in \mathbb{N}_0^M \) be a vector denoting the state of the network at time \( t \geq 0 \), with \( N_i(t) \) representing the number of class-\( i \) users at time \( t \geq 0 \).

The network operates under the AFS policy, as introduced in [140]. When the network is in state \( n = (n_1, \ldots, n_M) \in \mathbb{N}_0^M \setminus \{0\} \), the service rate \( x_i^* \) allocated to each of the class-\( i \) users is obtained by solving the optimization problem presented in Section 1.4.3.

Let \( s_i(n) := n_i x_i^* \) denote the total service rate allocated to class \( i \). Since the rate allocated to single flows is often restricted in practice, we assume that the effective total rate allocated to class-\( i \) users is [14]

\[
d_i(n) := \min \{ s_i(n), n_i x_i \},
\]
where $r_i$ can be thought of as the access-link rate limitation for a class-$i$ flow, $i = 1, \ldots, M$.

**Remark:** Above we first determined the AFS allocation, and then truncated the resulting rates at the access-link rates $r_i$, $i = 1, \ldots, M$. Note that if $n_i r_i < s_i(n)$ for some class $i$, then the excess rate $s_i(n) - n_i r_i$ does not become available to the other classes. A way to circumvent this problem is to incorporate the restrictions $x_i \leq r_i$, $i = 1, \ldots, M$, directly in the optimization problem (1.2). In this chapter, however, we do not choose to use this latter approach; it is not in the scope of this chapter to verify which of these alternatives is closest to reality. It turns out that in general our approach allows fairly explicit analysis, whereas this is considerably harder under the alternative method, see also [14].

It is easily verified that $N(t)$ is a Markov process with transition rates:

$$q(n, n + e_i) = \lambda_i; \quad q(n, n - e_i) = \nu_i(n), \quad i = 1, \ldots, M,$$

where $\nu_i(n) := \mu_i d_i(n)$. We note that $N(t)$ is in fact an $M$-dimensional birth-death process. Given that $r_i \geq c_i$, $i = 1, \ldots, M$, i.e., given that there are no access-link rate limitations, in [23] the appealing result was shown that $N(t)$ is an ergodic Markov process if

$$\sum_{i \in S(j)} \rho_i < c_j, \quad j = 1, \ldots, L. \tag{6.1}$$

Since the ‘down’ rates of our system differ only for a finite number of states from those in a similar system without rate limitations, it follows from Proposition 1 in [111] that $N(t)$ is ergodic for all values of $r_i > 0$, $i = 1, \ldots, M$, given that (6.1) holds. We emphasize that in general no explicit expressions are known for the steady-state distribution of $N(t)$.

In this chapter our goal is to estimate

$$P := P(N(T) \in A|N(0) = n_0),$$

i.e., the probability that, given that network is in state $n_0$ at time 0, the state of the network at time $T > 0$ is contained in set $A$. For example, here $n_0$ might be a state around which the network operates most of the time, and $A$ might be an ‘overflow set’:

$$\left\{ (x_1, \ldots, x_M) \in \mathbb{N}_0^M \mid \sum_{i=1}^M x_i > b \right\},$$

where $b \geq 0$ is a scalar.
6.1 Preliminaries

6.1.2 Variance-reduction technique

As in general no analytical expression for $P$ is known, a natural approach to obtain an estimate of $P$ is to perform simulation experiments. Let $\Omega = \{f_i, i = 1, 2, \ldots \}$ be the set of all paths $f$ in the evolution of the system, given that the system is in state $n_0$ at time $t = 0$, i.e., $f(0) = n_0$. Let $1_E$ be an indicator of the event $E$, and $p(f)$ the probability measure of the sample path $f$. Then we obtain that

$$
P = \int_{\Omega} 1_{f(T) \in A} p(f) df = E_p \left( 1_{f(T) \in A} \right),
$$

(6.2)

where the subscript $p$ indicates sampling from the measure $p$. An unbiased estimate of (6.2) can be obtained by performing MC simulation, i.e., we run $R$ independent simulations, with the system starting in state $n_0$, and we determine

$$
P_{MC} := \frac{1}{R} \sum_{i=1}^{R} 1_{f_i(T) \in A},
$$

where $f_i$ is the path obtained in the $i$th run. In case $n_0$ and $A$ are such that $f(T) \in A$ occurs relatively often, we can accurately estimate $P$ in a relatively small amount of time by $P_{MC}$. The number of runs needed to obtain an estimate with predefined accuracy and confidence, is in general inversely proportional to the probability to be estimated, see e.g. [80].

If $n_0$ and $A$ are such that $f(T) \in A$ is a rare event, then the above properties entail that we need a large number of simulations to provide an accurate statistical estimate of $P$. In this case the simulation can be accelerated by using IS. The idea underlying IS is to simulate the system with a new set of input probability distributions, such that the rare event becomes more likely. To this end, let us consider a new probability measure $p'$. Then, (6.2) is equivalent to

$$
P = \int_{\Omega} 1_{f(T) \in A} \frac{p(f)}{p'(f)} p'(f) df = \int_{\Omega} 1_{f(T) \in A} L(f) p'(f) df = E_{p'} \left( 1_{f(T) \in A} L(f) \right),
$$

(6.3)

where $L(f) := p(f)/p'(f)$ is called the likelihood ratio. Note that (6.3) is valid for any measure $p'($·$)$, given that $p'(f) > 0$ for all $f$ that are such that $f(T) \in A$. Hence, an unbiased IS estimator is given by

$$
P_{IS} := \frac{1}{R} \sum_{i=1}^{R} 1_{f_i(T) \in A} L(f_i),
$$

where $f_i$ is now simulated under the measure $p'$, with $f_i(0) = n_0, i = 1, \ldots, R$. 

Assuming that \( L(f) \) can be found, the simulation can be accelerated considerably if \( p' \) is properly chosen, in the sense that the number of runs needed to obtain an accurate statistical estimate of \( P \) using \( P_{\text{IS}} \), is less than the number of runs required in case of MC simulation. Hence, IS can be seen as a variance-reduction technique. We note, however, that not every choice of \( p' \) will reduce the variance. In fact, if \( p' \) is badly chosen, then this may increase the variance, or even make it infinite.

In this chapter we assume that \( n_0 \) and \( A \) are such that \( f(T) \in A \) is a rare event. As mentioned above, in this case MC simulation is inefficient, and one may resort to IS to obtain an estimate of \( P \). We derive an IS scheme that considerably speeds up the simulation. This scheme is based on sample-path large deviations results, see e.g. [161].

### 6.1.3 Large deviations

In this subsection we present large deviations results of [161], which will be needed in the next sections.

Let \( X(t) \) be a Markovian jump process with state space \( \mathbb{R}^d \) and with transition rates:

\[
q(x, x + v_i) = \psi_i(x),
\]

where \( v_i \) is a vector in \( \mathbb{R}^d \) and \( \psi_i(x) \) is the rate of the jump in that direction when the state is \( x \), \( i = 1, \ldots, l \). Also, let \( \overline{X}^k(t) := X(kt)/k, \ t \geq 0, \ k \geq 1, \) be the fluid scaled process, which is obtained by making the jumps smaller, but faster. Define the ‘local’ rate function

\[
\ell(x, y) := \sup_{\theta} \left( \langle \theta, y \rangle - \sum_{i=1}^{l} \psi_i(x) \left( e^{\langle \theta, v_i \rangle} - 1 \right) \right),
\]

where \( x, y \) and \( \theta \) are in \( \mathbb{R}^d \), and \( \langle \cdot, \cdot \rangle \) denotes the usual inner product: \( \langle a, b \rangle := \sum_{i=1}^{d} a_i b_i \). Finally, define the rate function

\[
I_T(f) := \begin{cases} 
\int_{0}^{T} \ell(f(s), f'(s))ds & \text{if } f \text{ is absolutely continuous}; \\
\infty & \text{otherwise},
\end{cases}
\]

where \( f \) is in \( \mathbb{R}^d \), and \( f' \) is the derivative of \( f \). The following sample-path large deviations principle (LDP) now holds (see Theorem 5.1 in [161]).

**Theorem 6.1.1** For any well-defined \( x_0 \) and set \( F \),

\[
- \lim_{k \to \infty} \frac{1}{k} \log \mathbb{P} \left( \overline{X}^k(\cdot) \in F | \overline{X}^k(0) = x_0 \right) = \inf_{f: f \in F, f(0) = x_0} I_T(f).
\]
Remark: Intentionally, Theorem 6.1.1 has been formulated in a slightly imprecise manner. In fact, the LDP consists of an upper and lower bound, which apply to closed and open sets, respectively, see Theorem 5.1 in [161]. However, for the purpose of this chapter, it is sufficient to state the theorem as above. For more details we refer to Chapter 5 of [161].

Let us write $g(x) \sim h(x)$ when $g(x)/h(x) \to 1$ if $x \to \infty$. Then it follows from the above that

$$
P\left(\mathbf{X}^k(\cdot) \in F | \mathbf{X}^k(0) = x_0\right) \sim g(k, F, x_0)e^{-kI_f(f^*)}, \quad k \to \infty,$$

where $f^*$ is the optimizing path in Theorem 6.1.1, and $g(k, F, x_0)$ is a subexponential function, i.e.,

$$
\lim_{k \to \infty} \frac{\log g(k, F, x_0)}{k} = 0.
$$

From the above it follows that Theorem 6.1.1 only gives the logarithmic asymptotics. Therefore, in general Theorem 6.1.1 does not provide any information on the function $g(k, F, x_0)$, which implies that we can only use it to obtain a rough estimate of

$$
P\left(\mathbf{X}^k(\cdot) \in F | \mathbf{X}^k(0) = x_0\right).
$$

In the next section we apply Theorem 6.1.1 to the so-called free M/M/1-PS process.

### 6.2 Free M/M/1-PS process

We first assume that $X(t)$ corresponds to the free M/M/1-PS process, i.e., the M/M/1-PS queue that is not reflected at 0, meaning that the state space of $X(t)$ is $\mathbb{N}$ (whereas the state space of a M/M/1-PS queue is $\mathbb{N}_0$). We note that the queue length dynamics of the M/M/1-PS queue coincide with those of the M/M/1-First-In First-Out (FIFO) queue, implying that both have the same steady-state queue length distribution. Hence, the results derived in this section in fact hold for the free M/M/1-FIFO process as well.

In this section we treat the free M/M/1-PS process, because this plays a key role in the analysis of a general AFS network topology, as we will see in Section 6.3. This may sound surprising, as the down rates corresponding to free M/M/1-PS process are constant, whereas the down rates corresponding to a general AFS network topology are state-dependent. The idea underlying this analysis is that we can locally approximate the flow-level dynamics of a particular class in a general AFS network topology by a free M/M/1-PS process with class-specific arrival and service rates, which will be exploited in the next sections to obtain an estimate of $P$.

Since $X(t)$ corresponds to the free M/M/1-PS process, we have that $X(t) = X_{\text{up}}(t) - X_{\text{down}}(t)$, where $X_{\text{up}}(t)$ is a Poisson process of rate $\lambda$ and $X_{\text{down}}(t)$ is an
importance sampling in rate-sharing networks

Assume that \( \lambda < \mu \), so that \( X(t) \) has a negative drift. The transition structure of \( X(t) \) is then, in the terminology of Section 6.1.3,

\[
\begin{align*}
& v_1 = +1; \quad \psi_1(x) = \lambda; \\
& v_2 = -1; \quad \psi_2(x) = \mu,
\end{align*}
\]

with \( x \in \mathbb{Z} \). Then,

\[
\ell(x, y) = \ell(y) = \sup_\theta \left\{ \theta y - \lambda (e^\theta - 1) - \mu (e^{-\theta} - 1) \right\},
\]

i.e., the local rate function is independent of the current state \( x \). Straightforward calculus shows that the optimizer satisfies

\[
e^{\theta^*} = \frac{y + \sqrt{y^2 + 4\lambda \mu}}{2\mu},
\]

which yields

\[
\ell(y) = y \log \left( \frac{y + \sqrt{y^2 + 4\lambda \mu}}{2\lambda} \right) + \lambda + \mu - \sqrt{y^2 + 4\lambda \mu} =: \ell(y|\lambda, \mu).
\]

We now focus on the overflow probability

\[
P(\mathcal{X}_k(T) > z|\mathcal{X}_k(0) = z_0),
\]

with \( z > z_0 \). Using Theorem 6.1.1, we have that

\[
P(\mathcal{X}_k(T) > z|\mathcal{X}_k(0) = z_0) \approx e^{-I^*},
\]

where

\[
I^* := \inf_{f: f \in G, f(0) = z_0} I_T(f), \text{ with } G := \{ f | f(T) > z \}.
\]

In Lemma 5.1.16 of [161] it is shown that the MPP, i.e., the path \( f^* \) in the set \( G \) that minimizes \( I_T \), is a straight line from \( z_0 \) to \( z \) in the interval \([0, T]\), with cost

\[
I^* = I_T(f^*) = T \times \ell \left( \frac{z - z_0}{T}, \lambda, \mu \right)
\]

\[
= T \left( \frac{z - z_0}{T} \log \left( \frac{z - z_0}{2T\lambda} + \frac{1}{2\lambda} \sqrt{\frac{(z - z_0)^2}{T^2} + 4\lambda \mu} \right) \right)
\]

\[
+ \lambda + \mu - \sqrt{\frac{(z - z_0)^2}{T^2} + 4\lambda \mu}
\]

\[
=: C(z - z_0, T|\lambda, \mu).
\] (6.4)
Below we try to provide some additional interpretation for (6.4). First note that the cost of a Poisson process of rate $\lambda$ behaving like a Poisson process of rate $\lambda^*$ is, during one unit of time,

$$\tilde{I}(\lambda^*|\lambda) := \lambda^* \log \left( \frac{\lambda^*}{\lambda} \right) + \lambda - \lambda^*, \,$$

see page 20 of [161]. Here $\tilde{I}(\lambda^*|\lambda)$ is the Legendre transform of the logarithmic Moment Generating Function (MGF) of a random variable that has a Poisson distribution with mean $\lambda$. Clearly, $\tilde{I}(\mu^*|\mu)$ follows in the same way. Observe that indeed $\tilde{I}(p|p) = 0$, $p = \lambda, \mu$, as required.

In order to make sure that it becomes likely that $X(T) > z$, given that $X(0) = z_0$, we should have that $X_{\text{up}} (X_{\text{down}})$ behaves as a different Poisson process of rate $\lambda^*$ ($\mu^*$), where $(\lambda^* - \mu^*)T > z - z_0$. We thus get the minimization problem:

$$T \min_{\lambda^*, \mu^*} \left\{ \tilde{I}(\lambda^*|\lambda) + \tilde{I}(\mu^*|\mu) \right\},$$

over all $\lambda^*, \mu^*$ such that $(\lambda^* - \mu^*)T > z - z_0$. Straightforward calculations yield that the optimizers are

$$\lambda^* = \frac{z - z_0}{2T} + \frac{1}{2} \sqrt{\frac{(z - z_0)^2}{T^2} + 4\lambda\mu};$$
$$\mu^* = -\frac{z - z_0}{2T} + \frac{1}{2} \sqrt{\frac{(z - z_0)^2}{T^2} + 4\lambda\mu},$$

and the corresponding objective function value indeed equals (6.4).

### 6.3 Most probable path

In the previous section we obtained an approximation for the overflow probability in the $\text{M/M/1-PS}$ queue (where we assumed that there was no reflection at 0). In this section we use the same ideas to derive an approximation for $P$ in a general AFS network topology.

We first consider the cost $\mathbb{K}(f, T)$ of a path $f$, with $f(0) = n_0$, in the interval $[0, T]$. We find that

$$\mathbb{K}(f, T) = \sum_{i=1}^{M} \int_{0}^{T} \ell(f'_i(t)|\lambda_i, \nu_i(f(t))) \, dt.$$

From the logarithmic asymptotics stated in Theorem 6.1.1 it then follows that the following approximation applies:

$$P = \mathbb{P}(N(T) \in A|N(0) = n_0) \approx \exp \left( - \inf_{f: f(T) \in A, f(0) = n_0} \mathbb{K}(f, T) \right).$$

(6.6)
Importance Sampling in rate-sharing networks

Let \( f^* \) denote the path that minimizes the cost, i.e., the MPP. Since the down rates in our model are state-dependent, in contrast to what is the case for the free M/M/1-PS process, the MPP in general has a non-linear shape. In fact, in general no closed-form expression is available for the path that minimizes \( K(f, T) \). Equation (6.6) suggests that we should try to find an accurate approximation of \( f^* \) to obtain an estimate of \( P \), which is done below.

Divide \( T \) into \( m \) (which is typically a large number) subintervals of length \( \Delta_m := \frac{T}{m} \). Consider the contribution to a path of the \( k \)-th subinterval, i.e., the interval \( [k\Delta_m, (k+1)\Delta_m) \), for \( k = 0, \ldots, m - 1 \), and assume that the down rates are \( \nu_i(f(k\Delta_m)) \), \( i = 1, \ldots, M \), in this subinterval. Then the cost of this time interval, related to class \( i \) are given by

\[
\mathcal{C}(f_i((k+1)\Delta_m) - f_i(k\Delta_m), \Delta_m|\lambda_i, \nu_i(f(k\Delta_m))).
\]

Hence, we find that the total cost \( K_m(f, T) \) are

\[
\sum_{i=1}^{M} \sum_{k=0}^{m-1} \mathcal{C}(f_i((k+1)\Delta_m) - f_i(k\Delta_m), \Delta_m|\lambda_i, \nu_i(f(k\Delta_m))).
\]

Note that the higher the value of \( m \), the more accurate the approximation will be, i.e.,

\[
\lim_{m \to \infty} K_m(f, T) = K(f, T).
\]

Using the above, we can approximate \( K(f, T) \), for given \( m \in \mathbb{N} \), by \( K_m(f, T) \). Also, the path that minimizes \( K_m(f, T) \) can be regarded as an approximation of \( f^* \). In order to obtain this approximating path, optimization should be performed over all \( f_i(j\Delta_m), i = 1, \ldots, M, j = 0, \ldots, m \), i.e., \( (m+1)M \) entries, given that \( f(0) = n_0 \) and \( f(m\Delta_m) = f(T) \in A \).

Approximation (6.6) turns out not to be very accurate in general. Clearly, this is no surprise, as in Section 6.1.3 we already argued that Theorem 6.1.1 just gives the logarithmic asymptotics, and that we therefore have only a rough estimate of \( P \).

### 6.4 New input distributions

In the previous section we derived an approximation for \( P \) which required the calculation of an optimizing path. This path can be regarded as an approximation for the most likely way for the event to happen. That is, given that the event occurs, with overwhelming probability \( N(T) \in A \) is reached by a path close to this optimizing path. In this section we show how we can exploit the results of Section 6.3 to develop a methodology for obtaining an accurate estimate of \( P \).

Assume that we have (an accurate approximation of) the MPP

\[
f^* := \arg \inf_{f : f(T) \in A, f(0) = n_0} K(f, T),
\]
as discussed in the previous section. Suggested by (6.5), the following change-of-measure at time $t$ corresponds to $f^*$:

$$\lambda_i^*(t) := \frac{1}{2}(f^*_i)'(t) + \frac{1}{2}\sqrt{((f^*_i)'(t))^2 + 4\lambda_i\nu_i(f^*(t))};$$

$$\nu_i^*(t) := -\frac{1}{2}(f^*_i)'(t) + \frac{1}{2}\sqrt{((f^*_i)'(t))^2 + 4\lambda_i\nu_i(f^*(t))},$$

$i = 1, \ldots, M$. When, at time $t \geq 0$, the process is simulated with arrival rates $\lambda^*(t)$ and departure rates $\nu^*(t)$, given that the process starts at $n_0$ at $t = 0$, it is not hard to see that the $i$th coordinate of the expected position of the process at time $t$ is

$$n_{0,i} + \int_0^t \lambda_i^*(s)ds - \int_0^t \nu_i^*(s)ds = f_i^*(0) + \int_0^t (f_i^*)'(s)ds = f_i^*(t),$$

$i = 1, \ldots, M$, i.e., the process has the ‘correct’ expected position, under this change-of-measure.

In the Appendix we present an IS scheme that can be used to obtain an estimate of $P$. The basic idea underlying this scheme is to simulate the model with rates $\lambda_i^*(t)$ and $\nu_i^*(t)$, $i = 1, \ldots, M$. Typically, we only know these rates at $m+1$ time points, as in general the MPP is not explicitly known, but approximated, see Section 6.3. However, if one assumes the rates to be constant between two consecutive time points, i.e., in a subinterval, then each class essentially behaves as a free M/M/1-PS process with class-specific arrival and service rate in this subinterval, which is easy to simulate. For more details we refer to the Appendix.

In the next section we show that, compared to MC simulation, this scheme can considerably speed up the simulation, given that the underlying event is rare. That is, the number of runs that are needed to achieve some given level of confidence with the IS scheme, is substantially less than the number of runs needed with MC simulation.

6.5 Simulation results

In this section the performance of the IS algorithm is examined in case of a single-node network (shared by multiple traffic classes) and a linear network, respectively. These are the two simplest networks, and therefore of particular interest to gain insight. We have performed extensive simulation experiments for each of these two networks, and the results are presented below. We mention that, besides the results reported in this section, we have considered many other examples, in which usually a substantial speed-up is achieved.
6.5.1 Single-node network

We first consider a single-node network with capacity \( c \), where capacity is shared between \( M \) classes. In order to obtain the AFS allocation we have to solve the following optimization problem for state \( n \in \mathbb{N}_0^M \setminus \{0\} \):

\[
\begin{align*}
\text{max} & \quad \sum_{i=1}^{M} n_i U_i(x_i) \\
\text{subject to} & \quad \sum_{i=1}^{M} n_i x_i \leq c \\
& \quad x_i \geq 0, \quad i = 1, \ldots, M,
\end{align*}
\]

where \( U_i(x_i) \) is as defined in (1.3). It is a straightforward exercise to show that the optimizers are such that

\[
s_i(n) = n_i x_i^* = \kappa_1^{1/\alpha} n_i c \sum_{j=1}^{M} \kappa_j^{1/\alpha} n_j, \quad i = 1, \ldots, M. \tag{6.7}
\]

From (6.7) it follows that AFS in a single-node network corresponds to sharing in a DPS fashion, with relative weights \( \kappa_i^{1/\alpha}, i = 1, \ldots, M \), see [63]. We find [14] that

\[
d_i(n) = \min \left\{ \kappa_i^{1/\alpha} n_i c \sum_{j=1}^{M} \kappa_j^{1/\alpha} n_j, n_i r_i \right\}, \quad i = 1, \ldots, M.
\]

The steady-state distribution of \( N(t) \) is only known in explicit form for some special cases, given that the stability condition \( \sum_{i=1}^{M} \rho_i < c \) holds. In case \( \kappa_i = \kappa \) and \( r_i \geq c, \ i = 1, \ldots, M \), the steady-state distribution is given by Equation (1.7). The steady-state distribution is available as well in case \( \kappa_i = \kappa \) and \( r_i = r \leq c, \ i = 1, \ldots, M \).

The first part of the IS algorithm consists of finding a MPP. We have performed numerical experiments to gain insight in the typical shape of such a minimizing path. We consider the setting with \( M = 2, \lambda_1 = 0.75, \lambda_2 = 1.5, \mu_1 = 2, \mu_2 = 4, \kappa_1^{1/\alpha} = 1/3, \kappa_2^{1/\alpha} = 2/3, \ r_1 = 0.9, \ r_2 = 0.8, \) and \( c = 1 \), and we let \( T, n_0 \) and the set \( A \) vary. The results are depicted in Figure 6.1, and are obtained by using an optimization procedure in Mathematica 5.2. We solved the problem for \( m = 2^p, p = 1, \ldots, 5 \), and we used the minimizing path found for \( m = 2^{p-1} \) as starting path in the optimization procedure for \( m = 2^p, \) \( q = 2, \ldots, 5 \) (for \( m = 2 \) we do not have a nice starting path). Hence, the depicted paths are associated with \( m = 2^5 = 32 \). We note that the above approach is much faster than solving the optimization problem directly for \( m = 32 \) (without an appropriate starting path). We observed that the optimization problem can be solved in a relatively small amount of time if \( m \leq 32 \). For higher values of \( m \) the obtained path is almost identical to the one obtained for \( m = 32 \), but the computation requires more time. In the first, second and third column of Figure 6.1 we depict \( (f_1(i\Delta_{32}), f_2(i\Delta_{32})), (f_2(i\Delta_{32}), f_3(i\Delta_{32})), \) and \( (f_1(i\Delta_{32}), f_3(i\Delta_{32})), i = 0, \ldots, 32 \), respectively. Since we only know the minimizing paths at \( m + 1 = 33 \) time points, we linearly interpolate between consecutive points. We note that we have considered
many other scenarios besides the ones depicted in Figure 6.1. In these cases, the minimizing paths do not seem to be linear either.

Although the shapes of the MPPs corresponding to scenarios (a)-(c) are not always trivial, the shape of the path corresponding to scenario (d) perhaps requires some more explanation. In particular, the shape of the path corresponding to class 1 is surprising in this scenario: it first slightly decreases, and then it starts to increase. A possible explanation for this phenomenon may be the following. In [14] it was shown that there exists a unique point \( n^* = (n_1^*, n_2^*) \) such that \( \lambda_i = d_i(n^*), i = 1, 2. \) This is the equilibrium point of the so-called fluid limit: the system operates (most likely) most of the time around this point. The fluid limit is obtained by both speeding up the arrivals and service speed by a given factor, and then letting this factor go to infinity. It can be shown that the resulting normalized Markov process converges to a deterministic limit. From Proposition 2.1 in [14] it follows that \( n_1^* = 0.5625 \) and \( n_2^* = 0.46875 \) in scenarios (a)-(d). Recalling that the path starts in \( n_0 = (3, 0) \) in scenario (d), we see that the MPP initially evolves in the direction of the fluid limit, but then changes its direction to make sure that \( f_2(T) > 6. \) It remains, however, hard to fully explain the shapes of the MPPs in general. One can imagine that the MPP from any \( n_0 \) to any set \( A \) is more or less linear if \( T \) is relatively small. In contrast, if \( T \) is relatively large, then one can expect that the MPP first drifts to \( n^* \), and then changes its direction towards the set \( A \), see e.g. [123]. We remark that the equilibrium point \( n^* \) depends critically on the access-link rates [14].

To quantify the performance of the proposed IS scheme we take the same parameter values as above, where we let \( T, n_0 \) and the set \( A \) vary. We consider three structures for \( A \): (i) \( \{f | f_1(T) > a \} \), (ii) \( \{f | f_2(T) > a \} \) and (iii) \( \{f | f_1(T) + f_2(T) > a \} \), with \( a > 0 \). The results are presented in Tables 6.1-6.4. These results (and also the ones in the next subsection) are obtained with Mathematica 5.2 and are tested on a personal computer with an AMD Athlon 64 3500+ processor (2.2 GHz). In the tables \#IS (\#MC) denotes the number of runs needed with IS (MC) simulation to obtain a confidence of 95% and a relative efficiency (i.e., the ratio of the confidence interval half-length to the estimated value) of 10%, and \( \tau_{IS} (\tau_{MC}) \) denotes the time needed with IS (MC simulation). Note that \( \tau_{IS} \) consists of two parts: (a) finding the optimal path and (b) performing the simulation with the new input distributions.

Table 6.1 compares IS with MC simulation. The MC estimator is obtained by simulating independent runs of the original model (starting in \( n_0 \)) until time \( T \), and subsequently determining the fraction of the runs that are such that \( f(T) \in A \). The table shows that for a relatively large value of \( P \) (larger than 0.01), MC simulation yields an accurate estimate much faster than the IS scheme does. In contrast, for a relatively small value of \( P \) (smaller than 0.01), IS significantly outperforms MC simulation. Clearly, this is no surprise: the IS scheme presented in the Appendix is based on large deviations results, and therefore one expects this scheme to perform well in case the underlying event is rare, i.e., if \( P \) is relatively small.
Importance Sampling in rate-sharing networks

Figure 6.1: The minimizing paths of $K_{32}(f,T)$ in four scenarios. Scenario (a): $T = 3$, $A = \{f|f_1(T) > 8\}$ and $n_0 = (1,4)$. Scenario (b): $T = 6$, $A = \{f|f_1(T) > 8\}$ and $n_0 = (1,4)$. Scenario (c): $T = 3$, $A = \{f|f_2(T) > 12\}$ and $n_0 = (1,4)$. Scenario (d): $T = 1$, $A = \{f|f_2(T) > 6\}$ and $n_0 = (3,0)$. The left panel shows $f_1^*(\cdot)$ as function of time $t$. The middle panel shows $f_2^*(\cdot)$ as function of time $t$. The right panel shows the parametric plot of $(f_1^*(\cdot), f_2^*(\cdot))$. 
6.5 Simulation results

Table 6.1: Simulation results for structure (i): comparison with MC simulation (times in seconds).

<table>
<thead>
<tr>
<th>( T )</th>
<th>( a )</th>
<th>( n_{0,1} )</th>
<th>( n_{0,2} )</th>
<th>( P_{IS} )</th>
<th>( #_{IS} )</th>
<th>( \tau_{IS} )</th>
<th>( P_{MC} )</th>
<th>( #_{MC} )</th>
<th>( \tau_{MC} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>6.1 ( \times ) 10(^{-4})</td>
<td>1559</td>
<td>12.4</td>
<td>6.2 ( \times ) 10(^{-4})</td>
<td>522835</td>
<td>341.1</td>
</tr>
<tr>
<td>3</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>3.0 ( \times ) 10(^{-4})</td>
<td>3312</td>
<td>16.4</td>
<td>2.7 ( \times ) 10(^{-4})</td>
<td>1468866</td>
<td>523.4</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>1</td>
<td>4</td>
<td>4.0 ( \times ) 10(^{-2})</td>
<td>2441</td>
<td>17.2</td>
<td>1.4 ( \times ) 10(^{-2})</td>
<td>77851</td>
<td>37.7</td>
</tr>
</tbody>
</table>

Table 6.2: Simulation results for structure (i): rare events (times in seconds).

<table>
<thead>
<tr>
<th>( T )</th>
<th>( a )</th>
<th>( n_{0,1} )</th>
<th>( n_{0,2} )</th>
<th>( P_{IS} )</th>
<th>( #_{IS} )</th>
<th>( \tau_{IS} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>3.3 ( \times ) 10(^{-7})</td>
<td>3599</td>
<td>18.9</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>1</td>
<td>0</td>
<td>5.1 ( \times ) 10(^{-13})</td>
<td>4744</td>
<td>24.4</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>1</td>
<td>0</td>
<td>2.9 ( \times ) 10(^{-6})</td>
<td>6762</td>
<td>33.9</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>1</td>
<td>0</td>
<td>2.6 ( \times ) 10(^{-5})</td>
<td>12439</td>
<td>59.2</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1</td>
<td>0</td>
<td>2.5 ( \times ) 10(^{-8})</td>
<td>12526</td>
<td>63.7</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>1</td>
<td>4</td>
<td>4.2 ( \times ) 10(^{-8})</td>
<td>2861</td>
<td>17.5</td>
</tr>
<tr>
<td>1</td>
<td>14</td>
<td>1</td>
<td>4</td>
<td>7.8 ( \times ) 10(^{-13})</td>
<td>3349</td>
<td>20.6</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>1</td>
<td>4</td>
<td>6.0 ( \times ) 10(^{-6})</td>
<td>2680</td>
<td>19.5</td>
</tr>
<tr>
<td>2</td>
<td>14</td>
<td>1</td>
<td>4</td>
<td>1.2 ( \times ) 10(^{-9})</td>
<td>4686</td>
<td>28.5</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>1</td>
<td>4</td>
<td>6.2 ( \times ) 10(^{-5})</td>
<td>4173</td>
<td>25.8</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>1</td>
<td>4</td>
<td>5.6 ( \times ) 10(^{-8})</td>
<td>5847</td>
<td>37.4</td>
</tr>
</tbody>
</table>

Tables 6.2-6.4 show the performance of our scheme in case of rare events. As mentioned in Section 6.1.2, in this case MC simulation is inefficient. Therefore, we have decided not to compare the performance of the IS scheme with that of the MC simulation. These tables show that our scheme works remarkably well for rare events: we are able to estimate probabilities as small as 10\(^{-13}\) in a fast way.

The results also show that the performance of the IS scheme decreases as \( T \) increases (for fixed other model parameters), i.e., more runs are needed to achieve the required efficiency. This can be explained as follows. As \( T \) increases and \( m \) (the number of subintervals) remains constant, the approximation of the minimizing path becomes less accurate, and therefore the performance of the IS algorithm is also negatively affected.

We also empirically observed that, for fixed arbitrarily chosen \( n_0 \) and \( T \), the ratio

\[
\frac{\log E_{p'} (1_{f(T) \in A \cdot k} L^2(f))}{\log E_{p'} (1_{f(T) \in A \cdot k} L(f))},
\]

is close to (but smaller than) 2 for large values of \( k \), where \( p' \) is the IS distribution and \( A \cdot k := \{ y : y/k \in A \} \). It is noted that one can estimate both denominator and numerator in (6.8) by using the simulation output. The above suggests that our IS scheme is nearly asymptotically optimal [41], which however seems difficult to prove.
Importance Sampling in rate-sharing networks

6.5.2 Linear network

We next consider a linear network that consists of L nodes, where node i has capacity \( c_i \). There are \( M = L + 1 \) classes of users: each class corresponds to a specific route in the network. Class-\( i \) users require service at node \( i \), \( i = 1, \ldots, L \), whereas class-(\( L + 1 \)) users require service at all \( L \) nodes simultaneously, see Figure 1.7 for an illustration. For ease of notation relabel class-(\( L + 1 \)) users as class-0 users. In order to obtain the AFS allocation we have to solve the following optimization problem for state \( n \in N_0^{L+1} \setminus \{0\} \):

\[
\text{max } \sum_{i=0}^{L} n_i U_i(x_i),
\]

subject to \( n_0 x_0 + n_i x_i \leq c_i, \quad i = 1, \ldots, L \)

over \( x_i \geq 0, \quad i = 0, \ldots, L \),

where \( U_i(x_i) \) is as defined in (1.3). Only in case \( c_i = c, \ i = 1, \ldots, L \), i.e., if all nodes have the same capacity, there exist explicit expressions for the optimizing \( x_i^* \)s. In that case the optimizers are such that [23]

\[
s_0(n) = n_0 x_0^* = \frac{(\kappa_0 n_0^a)^{1/\alpha_c}}{(\kappa_0 n_0^a)^{1/\alpha} + (\sum_{j=1}^{L} \kappa_j n_j^a)^{1/\alpha}};
\]

\[
s_i(n) = n_i x_i^* = (c - s_0(n)) 1/n_i > 0, \quad i = 1, \ldots, L.
\]
Therefore, we find that $d_0(n)$ equals
\[
\min \left\{ \frac{(\kappa_0 n_0^a)^{1/\alpha}}{\left(\kappa_0 n_0^a\right)^{1/\alpha} + \left(\sum_{j=1}^L \kappa_j n_j^a\right)^{1/\alpha}}, n_0 \right\},
\]
and $d_i(n)$, for $i = 1, \ldots, L$, equals
\[
\min \left\{ \frac{(\sum_{j=1}^L \kappa_j n_j^a)^{1/\alpha}}{\left(\kappa_0 n_0^a\right)^{1/\alpha} + \left(\sum_{j=1}^L \kappa_j n_j^a\right)^{1/\alpha}}, n_i \right\},
\]
The steady-state distribution of $N(t)$ is only known in explicit form if $\alpha = 1$, $\kappa_j = \kappa$, $c_j = c$, and $r_i \geq c$, $i = 0, \ldots, L$, $j = 1, \ldots, L$, given that the stability condition $\max_{1 \leq i \leq L} \lambda_0 + \rho_i < c$ holds, see Theorem 7.2.1.

We test the performance of our IS scheme in case $L = 2$, $\lambda_0 = 2$, $\lambda_1 = 1$, $\lambda_2 = 1.75$, $\mu_0 = 5$, $\mu_1 = 2$, $\mu_2 = 4$, $r_0 = 0.8$, $r_1 = 0.6$, $r_2 = 0.3$, $\kappa_0 = 0.5$, $\kappa_1 = 2$, $\kappa_2 = 1$, $\alpha = 1$, and starting state $(1,1,2)$. Furthermore, we assume $c_i = c = 1$, $i = 1, 2$, so that we have a closed-form expression for $d_i(n)$, $i = 0,1,2$, and we let $T$ and $A$ vary. We assume that the structure of $A$ is of the form $\{ f | f_0(T) + f_1(T) > a_1, f_0(T) + f_2(T) > a_2 \}$, with $a_1, a_2 > 0$. The results are given in Tables 6.5 and 6.6.

The results again show that the rare event probabilities can be estimated rather efficiently. Compared to the single-node network, it now takes much more time to find the MPP (which in general has a non-linear shape), as one needs to optimize over more entries.

<table>
<thead>
<tr>
<th>$T$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$P_{IS}$</th>
<th>$#IS$</th>
<th>$\tau_{IS}$</th>
<th>$P_{MC}$</th>
<th>$#MC$</th>
<th>$\tau_{MC}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>6</td>
<td>$6.1 \times 10^{-2}$</td>
<td>23008</td>
<td>126.9</td>
<td>$6.5 \times 10^{-2}$</td>
<td>5527</td>
<td>8.8</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>8</td>
<td>$7.8 \times 10^{-3}$</td>
<td>2102</td>
<td>140.3</td>
<td>$7.7 \times 10^{-3}$</td>
<td>49912</td>
<td>78.5</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>10</td>
<td>$8.5 \times 10^{-2}$</td>
<td>5045</td>
<td>145.2</td>
<td>$4.6 \times 10^{-2}$</td>
<td>7908</td>
<td>23.4</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>$8.5 \times 10^{-3}$</td>
<td>4573</td>
<td>172.5</td>
<td>$8.9 \times 10^{-3}$</td>
<td>40859</td>
<td>121.3</td>
</tr>
<tr>
<td>3</td>
<td>12</td>
<td>12</td>
<td>$6.7 \times 10^{-3}$</td>
<td>9722</td>
<td>274.2</td>
<td>$3.0 \times 10^{-3}$</td>
<td>12010</td>
<td>52.6</td>
</tr>
</tbody>
</table>

Table 6.5: Simulation results for the linear network: comparison with MC simulation (times in seconds).

<table>
<thead>
<tr>
<th>$T$</th>
<th>$a_1$</th>
<th>$a_2$</th>
<th>$P_{IS}$</th>
<th>$#IS$</th>
<th>$\tau_{IS}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>15</td>
<td>15</td>
<td>$1.6 \times 10^{-12}$</td>
<td>6481</td>
<td>191.3</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>20</td>
<td>$7.0 \times 10^{-12}$</td>
<td>10782</td>
<td>199.7</td>
</tr>
<tr>
<td>2</td>
<td>20</td>
<td>20</td>
<td>$3.7 \times 10^{-12}$</td>
<td>10994</td>
<td>175.1</td>
</tr>
<tr>
<td>2</td>
<td>25</td>
<td>25</td>
<td>$3.1 \times 10^{-12}$</td>
<td>19326</td>
<td>312.0</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>30</td>
<td>$2.0 \times 10^{-12}$</td>
<td>48310</td>
<td>631.5</td>
</tr>
</tbody>
</table>

Table 6.6: Simulation results for the linear network: rare events (times in seconds).
6.6 Discussion

In this chapter we studied the transient behavior of the process $N(t)$. A topic for further research is the derivation of an approximation of $\pi(A)$, where $\pi(\cdot)$ denotes the steady-state distribution of $N(t)$. Using regenerative arguments, one can obtain $\pi(A)$ by dividing the expected time that the process spends in set $A$ during a cycle from $n_0$ to $n_0$, by the associated expected cycle time, see e.g. Corollary 1.4 in [12]. One may use specific measures to estimate both numerator and denominator, so-called measure specific dynamic IS, see e.g. [67]. Dynamic refers to the fact that per run the IS is turned on until the event of interest occurs and turned off thereafter.

Appendix

Below we present the pseudocode of an IS scheme that can be used to estimate rare event probabilities.

**IS Algorithm**

Compute (or approximate) the minimizing path $f^*$.
Divide $T$ into $m$ subintervals of length $\Delta_m := T/m$.

FOR $j = 1$ TO $R$
  $\tilde{N}_i(0) \leftarrow n_{0,i}$, $i = 1, \ldots, M$.

Set the likelihood ratio equal to 1: $L_j \leftarrow 1$.

FOR $k = 1$ TO $m$
  $\tilde{N}_i((k-1)\Delta_m) \leftarrow \tilde{N}_i((k-1)\Delta_m)$, $i = 1, \ldots, M$.
  Simulate Arrivals of type $i$ as Poisson process of rate $\lambda_i^*(k\Delta_m)$.
  Simulate Departures of type $i$ as Poisson process of rate $\nu_i^*(k\Delta_m)$.
  Thus $K$ events are generated, with inter-event times $t_1, \ldots, t_K$.

FOR $\ell = 1$ TO $K$
  IF Event($\ell$) = Arrival of type $i$
    THEN
      Update likelihood:
      $L_j \leftarrow L_j \times \exp((\lambda_i^*(k\Delta_m) - \lambda_i) t_\ell) \times (\lambda_i/\lambda_i^*(k\Delta_m))$.
      $\tilde{N}_i(k\Delta_m) \leftarrow \tilde{N}_i(k\Delta_m) + 1$.
  
  IF Event($\ell$) = Departure of type $i$ AND $\tilde{N}_i(k\Delta_m) > 0$
    THEN
      Update likelihood:
      $L_j \leftarrow L_j \times \exp((\nu_i^*(k\Delta_m) - \nu_i(\tilde{N}(k\Delta_m))) t_\ell)
      \times (\nu_i(\tilde{N}(k\Delta_m))/\nu_i^*(k\Delta_m))$.
      $\tilde{N}_i(k\Delta_m) \leftarrow \tilde{N}_i(k\Delta_m) - 1$.
  
  IF Event($\ell$) = Departure of type $i$ AND $\tilde{N}_i(k\Delta_m) = 0$
    THEN
      Set the likelihood ratio equal to 0: $L_j \leftarrow 0$.
  
  Abort current simulation run and proceed with the next run.
Set $t_K$ equal to 0 when $K = 0$.

FOR $i = 1$ TO $M$

Update likelihood:
\[
L_j \leftarrow L_j \times \exp((\lambda_i^*(k\Delta_m) - \lambda_i)(\Delta_m - t_K)) \\
\times \exp((\nu_i^*(k\Delta_m) - \nu_i(\tilde{N}(k\Delta_m)))(\Delta_m - t_K)).
\]

END

END

Put $1_j \leftarrow 1$ if $N(m\Delta_m) \in A$, and 0 else.

END

Estimator $P_{IS} \leftarrow R^{-1} \cdot \sum_{j=1}^{R} 1_j L_j$.

Justification of the IS algorithm: We simulate the process $\tilde{N}(t) = (\tilde{N}_1(t), \ldots, \tilde{N}_M(t))$ during a time period of $T$ units, given that $\tilde{N}(0) = n_0$, where

\[
\tilde{N}_i(t) := \tilde{N}_{i,\text{up}}(t) - \tilde{N}_{i,\text{down}}(t), \quad i = 1, \ldots, M,
\]

with $\tilde{N}_{i,\text{up}}(t)$ being a Poisson process of rate $\lambda_i^*(k\Delta_m)$ and $\tilde{N}_{i,\text{down}}(t)$ being a Poisson process of rate $\nu_i^*(k\Delta_m)$ if $t \in [(k-1)\Delta_m, k\Delta_m)$, $k = 1, \ldots, m$. Clearly, this corresponds to the process described in Section 6.1.1, but with different input distributions and with a different state space, as the state space of $\tilde{N}(t)$ is $\mathbb{Z}^M$, whereas that of $N(t)$ is $\mathbb{N}_0^M \subset \mathbb{Z}^M$. It follows from Section 6.1.2 that we can obtain an unbiased IS estimator of $P$ by simulating $\tilde{N}(t)$ and by keeping track of the likelihood ratio in each run.

In the algorithm we use that the interarrival times are exponentially distributed with mean $1/\lambda_i^*(k\Delta_m) (1/\lambda_i)$ under the new (old) measure if $t \in [(k-1)\Delta_m, k\Delta_m)$. Also, we exploit that the service requirements are exponentially distributed with mean $1/\nu_i^*(k\Delta_m) (1/\nu_i(\tilde{N}(t)))$, with $\tilde{N}_i(t) > 0$ under the new (old) measure, if $t \in [(k-1)\Delta_m, k\Delta_m)$. Clearly, if $\tilde{N}_i(t) = 0$ and a departure of class $i$ occurs, then we reach a state that is infeasible in our model (that is, under the original probability measure), so that we set $L$ equal to zero when this occurs. Since the likelihood ratio will stay zero once it has reached zero, one can abort the current simulation run. By simulating $R$ independent runs, adding all the likelihood ratios at time $m\Delta_m = T$ of the runs that are such that $\tilde{N}(T) \in A$, and dividing this sum by $R$, we obtain an unbiased estimator of $P$.

Remark: The obvious advantage of the above algorithm is that the change-of-measure has to be computed just once, and can be applied in all runs. The drawback is that there is no control within the run: if the process happens to deviate from the minimizing path, it is not directed back towards this path. These considerations may lead to the following approach. Denote by $f^*([n_0, A, T])$ the minimizing path corresponding to the probability $P$. Define

\[
g(s|t) := f^*(s|\tilde{N}(t), A, T - t),
\]
i.e., suppose that we find ourselves in state $\tilde{N}(t)$ at time $t$, and we wish to reach the set $A$ at time $T$, then $g(s|t)$ defines the most likely position at time $s+t$. Note that this implies that $g(s|0) = f^*(s)$. This suggests that one should use the rates

$$\tilde{\lambda}_i(t) := \frac{1}{2} g'_i(0|t) + \frac{1}{2} \sqrt{(g'_i(0|t))^2 + 4\lambda_i \nu_i(\tilde{N}(t))};$$

$$\tilde{\nu}_i(t) := -\frac{1}{2} g'_i(0|t) + \frac{1}{2} \sqrt{(g'_i(0|t))^2 + 4\lambda_i \nu_i(\tilde{N}(t))},$$

$i = 1, \ldots, M$. It can be checked that also for these rates the expected position at time $t$ is $f^*(t)$, but the difference with the first algorithm is that the process evolution is better controlled, cf. [21, 55, 57]. In practice the interval $[0, T]$ is again split into $m$ subintervals, and the rates $\tilde{\lambda}_i(k\Delta_m)$ and $\tilde{\nu}_i(k\Delta_m)$ are used in the $k$-th interval. Unfortunately, this approach is very time-consuming, as it requires the calculation of a minimizing path in each of the $m$ subintervals.