Analysis of a BMAP/D/1-Timer multiplexer

Gábor Horváth [1]

Department of Telecommunications
Budapest University of Technology and Economics
Budapest, Hungary

Miklós Telek [2]

Department of Telecommunications
Budapest University of Technology and Economics
Budapest, Hungary

Abstract

In this paper we introduce and analyze a model of a multiplexer queue with a batch Markovian arrival process and a special, timer based, non-work-conserving service discipline. We show that the embedded process at departures is an M/G/1 type process with proper state partitioning, which can be efficiently analyzed by matrix geometric methods. We derive the expressions to compute the distribution of the waiting time. The paper ends with numerical experiments, and points out some interesting features of the system.

1 Introduction

There are network protocols in packet switched networks in which multiplexers compose fixed size packets from the incoming data units. Such a protocol is the ATM AAL2 protocol [1], which plays an important role in the radio access networks of 3G mobile telecommunication systems.

In an AAL2 multiplexer there is an additional component of transmission delay. When there is not enough data to compose a complete packet the multiplexer waits for incoming data to fill a packet. This additional delay can be limited with the use of a timer. When this timer expires the multiplexer transmits an incomplete packet.

[1] Email: ghorvath@hit.bme.hu
[2] Email: telek@hit.bme.hu

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

A variant of this kind of multiplexer behaviour is discussed in [9]. In this paper we discuss a continuous time matrix geometric model of this multiplexer behaviour in case of batch Markovian arrival process (BMAP) arrival.

The paper is organized as follows. Section 2 gives the description of the considered queueing model, for which the queue size analysis is provided by Section 3. Based on the queue size analysis the computation of the waiting time distribution is presented in Section 4. Section 5 summarizes the algorithm. Numerical examples and the analysis of the results are provided in Section 6. Finally, Section 7 concludes the paper.

2 Model Description

The BMAP/D/1-Timer multiplexer queue is a continuous time stochastic process. The arrival process is a continuous time batch Markovian arrival process ([6]), where the batch size represents the number of data units arriving at an arrival instant. The server performs the multiplexing: it awaits L data units, compiles one package from them, and serves it with a deterministic service time Δ (according to the FCFS queueing discipline).

There is an extra waiting time for the jobs caused by the server when it waits for more arrivals if there are not enough data units to form a complete output packet. To decrease this waiting time, a timer is introduced (T). This timer ensures that when a data unit reaches the head of the queue (positions 1 to L, thus it will be included in the next packet), it will be served in time T, even if there are not enough arrivals to form a complete packet during this time. In that case the server forms a partially filled output packet, and serves it the same service time (Δ).

To give a better description of the service mechanism, we summarize the behavior in the following 3 points. These 3 points will be referred to many times in the sequel, and will be used as 3 cases requiring different treatments during the analysis:

P1. While there are L data units or more in the queue, the timer is not running. The server takes the first L data units out of the queue, compiles the packet, and starts the service. After the service ends (time Δ), and the queue size still exceeds L, a new service starts. *In our model the queue contains the waiting data units only; the ones whose service already started but not yet finished are not included, they are stored in a temporary buffer.*

P2. If the queue size gets below L but above 0 when the service begins (after taking out the L data units whose service has started), the timer starts. When the service is ready and the queue is still below L, no new service begins until the timer elapses, or the sufficient amount of data arrives (see Figure 1 (b)).

P3. If an arrival happens when the queue is empty, and the batch size of the arrival is below L, the timer is started immediately. The first service can

start when the queue size exceeds L, or when the timer elapses. Of course the service can not start while the packet in the server did not leave the system (Figure 2).

The timer value (T) can even be smaller than Δ. In this case only arrivals into empty queue could be affected by the timer, if the remaining service time at the arrival instant is less than the timer. The system is non work-conserving with $T > 0$ and work-conserving with $T = 0$. We focus on the first case here.

Considering the requirements of the system, the setting of the timer is an engineering problem. The trade off is that if the timer is long, the packet utilization is high, but the waiting time increases. If the timer is short, the waiting time is less, but the packet utilization is lower. The packet utilization is characterized by the mean amount of data transmitted in a packet.

The incoming traffic of the multiplexer is described by a continuous time batch Markovian arrival process with cardinality m [6]. Its $m \times m$ generator is denoted by D. The arrival process itself is characterized by a set of matrices D_i ($\sum_{i=0}^{K} D_i = D$), where $[D_k]_{i,j}$ corresponds to the arrival of k data units with a state transition from i to j.

The background Markov process and the counting process are denoted by $J(t)$, and $N(t)$. Then $N(t), J(t)$ together form a Markov chain with the following generator:

$$\boldsymbol{\mathcal{Q}} = \begin{bmatrix} D_0 & D_1 & D_2 & \dots \\ & D_0 & D_1 & \dots \\ & & D_0 & \dots \\ & & & \ddots \end{bmatrix}.$$

In the following Sections we denote the transient state transition matrix of this Markov chain by $P(k,t)$, using the following definition:

$$[P(k,t)]_{i,j} \;=\; Pr(N(t) = k,\; J(t) = j \mid N(0) = 0,\; J(0) = i). \qquad (1)$$

The BMAP is a strong modelling tool for traffic characterization in Markovian analysis. It can be constructed from measured traffic behavior [8,5].

3 Queue Length Distribution at Packet Departures

The queueing system of Section 2 can be characterized by a discrete time Markov chain at packet departure instants. Embedding at departures usually leads to a so-called M/G/1 type structure, as it does in our case, too. The transition probability matrix builds up as follows:

$$\mathcal{X} = \begin{bmatrix} \mathcal{B} & \dots \\ \mathcal{A} & \dots \\ & \mathcal{A} & \dots \\ & & \mathcal{A} & \dots \\ & & & \mathcal{A} & \dots \\ & & & & \ddots & \vdots \end{bmatrix} \tag{2}$$

Usually M/G/1 type matrices are defined by their quadratic matrix blocks. Now we define the matrix by its block rows, because it simplifies the definition. The states inside the $m \times m$ blocks are reflecting the state of the arrival process; a transition from block i to block j means that the queue size changed from i to j since the last departure instant.

Matrix row $\mathcal{A}$ corresponds to case P1 of Section 2. In this case the inter departure time is exactly Δ, since the timer does not play a role. At the next embedded point the server will decrease the queue by L, so the queue size change equals to the arrivals during Δ minus L. Thus, $\mathcal{A}$ is defined by:

$$\mathcal{A} = \begin{bmatrix} P(0,\Delta) & P(1,\Delta) & P(2,\Delta) & P(3,\Delta) & \dots \\ & P(0,\Delta) & P(1,\Delta) & P(2,\Delta) & \dots \\ & & P(0,\Delta) & P(1,\Delta) & \dots \\ & & \ddots & \ddots & \vdots \\ & & & P(0,\Delta) & \dots \end{bmatrix}_{L \times m \text{ rows}}, \tag{3}$$

where $P(k,\Delta)$ is given by Eq. (1).

The definition of $\mathcal{B}$ is more complex due to the effect of the timer. We further divide $\mathcal{B}$, to distinguish between the completely idle (P2) and not completely idle (P3) cases:

$$\mathcal{B} = \begin{bmatrix} \hat{\mathcal{B}} \\ \hline \\ \tilde{\mathcal{B}} \\ \\ \end{bmatrix}. \tag{4}$$

The first block row (with height m) describes the transitions from the idle buffer (these matrices are denoted by $\hat{\mathcal{B}}$, and correspond to P3 in Section 2),

the other rows describe transitions from buffer levels 1 – L-1 (P2 in Section 2).

In both cases the timer starts (in case of $\hat{\boldsymbol{\mathcal{B}}}$ after the first arrival, if the batch size is less than L), and the next transition may happen later than Δ. The evolution of the queue size between levels 1 and L is important to capture the effect of the timer. Therefore we define the following two Markov chain generators. The continuous time Markov chain generator $\boldsymbol{Q}$ follows the queue size increase process between 1 and L, and $\boldsymbol{Z}(t)$ is the transition probability matrix of the buffer size increase process during time t (thus $\boldsymbol{Z}(t) = e^{\boldsymbol{Q}t}$):

$$\boldsymbol{Q} = \begin{bmatrix} D_0 & D_1 & \dots & D_{L-2} \\ & D_0 & \dots & D_{L-3} \\ & & \ddots & \vdots \\ & & & D_0 \end{bmatrix}, \quad \boldsymbol{Z}(t) = \begin{bmatrix} P(0,t) & P(1,t) & \dots & P(L-2,t) \\ & P(0,t) & \dots & P(L-3,t) \\ & & \ddots & \vdots \\ & & & P(0,t) \end{bmatrix}. \quad (5)$$

$\boldsymbol{\Pi}(t)$ describes the behavior of the timer. $[\boldsymbol{\Pi}(t)]_{i,j}$ is the probability that starting with i ($1 \leq i \leq L$) data unit in the buffer the state of the system will be j just after the start of the next service. The next service can start when the necessary number of data units have arrived until time t (first term in Eq. (6)), or at time t the buffer content is served even if a full packet does not come out (second term of Eq. 6). This probability matrix is computed by:

$$\boldsymbol{\Pi}(t) = \int_0^t e^{\boldsymbol{Q}\tau} d\tau \cdot \begin{bmatrix} D_{L-1} & D_L & \dots & D_K & & \\ D_{L-2} & D_{L-1} & \dots & D_{K-1} & D_K & \\ \vdots & \vdots & \ddots & \dots & \dots & D_K \\ D_1 & D_2 & \dots & \dots & \dots & \dots & D_K \end{bmatrix} + e^{\boldsymbol{Q}t} \cdot \begin{bmatrix} I_{m\times m} & 0 & \dots \\ I_{m\times m} & 0 & \dots \\ \vdots & \vdots & \\ I_{m\times m} & 0 & \dots \end{bmatrix}. \quad (6)$$

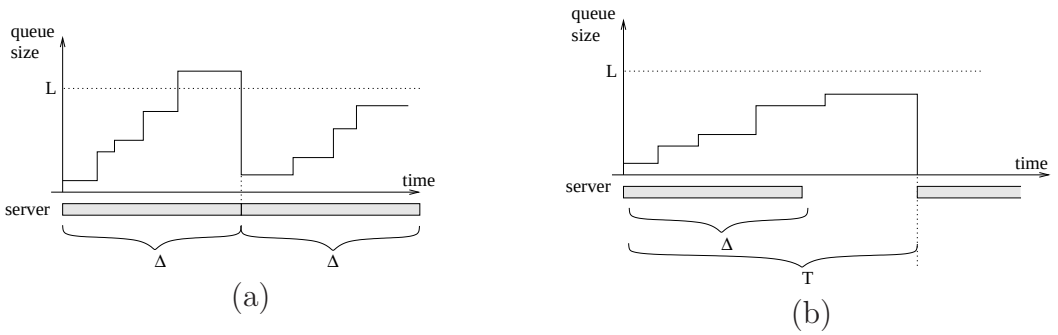

Fig. 1. Possible scenarios starting with buffer size 1 to L

If the buffer size is between 1 and $L-1$ ($\tilde{\boldsymbol{\mathcal{B}}}$), it has to be investigated how many data units arrive during the Δ interval (during which the server

is busy). If the arriving data units increase the buffer above L, the service of a new packet starts just after finishing the previous one (this events are expressed by the first matrix term of Eq. 7, and are depicted on Figure 1 (a)). If the buffer level is still below L (second term of Eq. 7, shown in Figure 1 (b)), an additional delay follows, with a maximal length of $(T-\Delta)^+$, since the timer started at the moment when the buffer decreased below L. Thus:

$$\tilde{\boldsymbol{\mathcal{B}}} = \begin{bmatrix} P(L-1,\Delta) & P(L,\Delta) & \dots \\ P(L-2,\Delta) & P(L-1,\Delta) & \dots \\ \vdots & \vdots & \dots \\ P(1,\Delta) & P(2,\Delta) & \dots \end{bmatrix} + \boldsymbol{Z}(\Delta) \cdot \boldsymbol{\Pi}((T-\Delta)^+), \qquad (7)$$

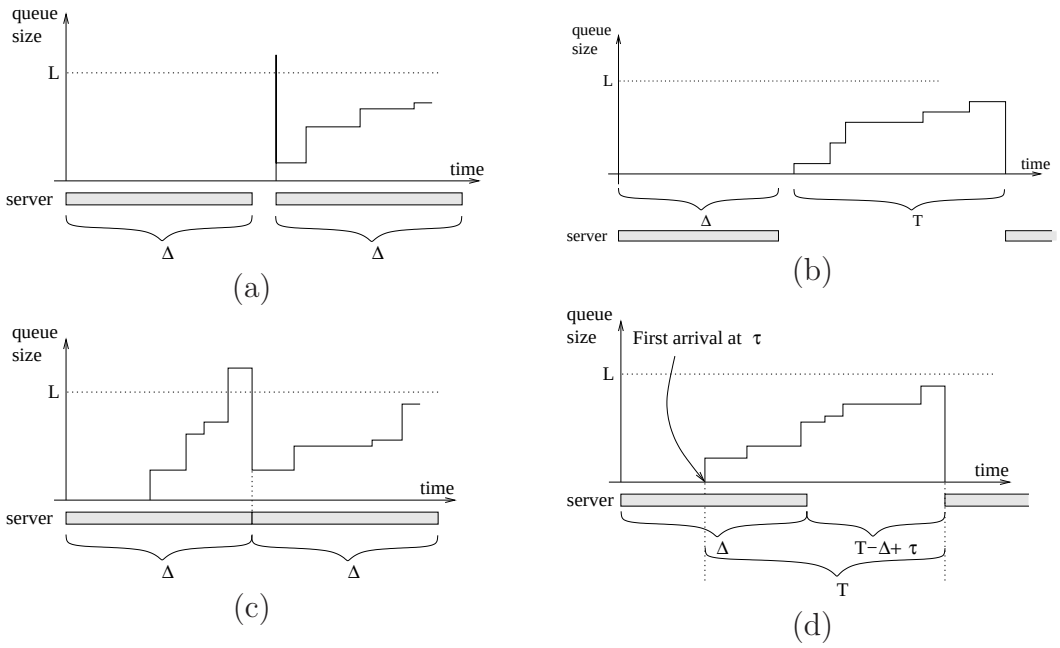

Fig. 2. Possible scenarios starting with empty buffer

If the buffer became empty at the beginning of the service of a packet ($\hat{\boldsymbol{\mathcal{B}}}$), we have two cases. It is possible that there are no arrivals during the server busy time (during Δ). In this case the next arrival can bring the queue above L, immediately causing a start of a new packet service (first term of Eq. 8, and Figure 2 (a)); or the queue remains below L, and a waiting period comes (waiting for more data units) for a maximal length of T, since the arrival into the empty queue initiated the timer (second term of Eq. 8, and Figure 2 (b)). The second case is when there were arrivals during the server busy time. In this case the arrivals can bring the queue above L, and the service of a new packet starts just after the end of service of the previous one (third term of Eq. 8, and Figure 2 (c)). The arrival can leave the system below L, and a waiting period is started with a maximal length of $(T-(\Delta-\tau))^+$, since

the timer started at the first arrival time (τ) and from the timer $\Delta - \tau$ time already expired when the server becomes empty (fourth term of Eq. 8, and Figure 2 (d)). Thus we have:

$$\begin{aligned}\hat{\boldsymbol{\mathcal{B}}} &= P(0,\Delta)(-D_0)^{-1}\left[D_L\ D_{L+1}\ \dots\ D_K\right] + \\ &+ P(0,\Delta)(-D_0)^{-1}\left[D_1\ D_2\ \dots\ D_{L-1}\right]\cdot\boldsymbol{\Pi}(T) + \\ &+ \left[P(L,\Delta)\ P(L+1,\Delta)\ \dots\right] + \\ &+ \int_0^{\Delta} e^{D_0\tau}\cdot\left[D_1\ D_2\ \dots\ D_{L-1}\right]\cdot\boldsymbol{Z}(\Delta-\tau)\cdot\boldsymbol{\Pi}((T-\Delta+\tau)^+)d\tau.\end{aligned} \tag{8}$$

The steady state distribution of the embedded Markov chain (2) is partitioned according to the state partitioning in the following way:

$$\boldsymbol{x} = \left[\underbrace{\left[p_0\ p_1\ \dots\ p_{L-1}\right]}_{\boldsymbol{x}_0}\ \underbrace{\left[p_L\ p_{L+1}\ \dots\ p_{2L-1}\right]}_{\boldsymbol{x}_1}\ \dots\right],$$

where the p_is are vectors of size m. In this section we defined the $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ blocks of $\boldsymbol{\mathcal{X}}$. The steady state probability vector that satisfies

$$\boldsymbol{x} \;=\; \boldsymbol{x}\,\boldsymbol{\mathcal{X}}$$

can be obtained by a matrix analytic method summarized in Appendix D.

4 Waiting Time Distribution

The waiting time distribution $P(W > w)$ is the probability that the waiting time of an arriving batch (measured from its arrival to its departure) exceeds a given threshold w. It can be easily computed if some parameters are kept fixed. These parameters are: the length of the buffer at the arrival (k), the remaining server occupation time (t_1), and the maximal departure time measured from the point when the the buffer descends below level L (t_2). If we know the particular values of these parameters, the waiting time distribution $P_W(k,t_1,t_2)$ can be computed by the following way:

$$P_W(k,t_1,t_2) = \begin{cases} h & \text{if } W^* < t_1 + \left\lceil \frac{k}{L} \right\rceil \Delta, \\ 0 & \text{if } k < L,\ \ W^* \geq t_1 + t_2 + \left\lceil \frac{k}{L} \right\rceil \Delta, \\ & \text{if } k \geq L,\ \ W^* \geq t_1 + (T-\Delta)^+ + \left\lceil \frac{k}{L} \right\rceil \Delta, \\ \left[e^{\boldsymbol{Q}(W^*-\Delta)}h\right]_{\{k/L\}} & \text{otherwise,} \end{cases} \tag{9}$$

where h is a vector of ones with size m.

The first item corresponds to the case when the server occupancy time plus the service time of packets in the queue exceeds the waiting time requirement. In this case $P(W > w)$ equals to one.

The second item covers the case when the waiting time requirement is surely satisfied. This happens if the waiting time requirement is larger than the server occupancy time, plus the service time of the packets in the queue, plus the maximal possible delay caused by the timer. The latter quantity is t_2 if the buffer size is less than L (this is the definition of t_2). It is $(T-\Delta)^+$ if $k \geq L$, because the server will be occupied when the last segment gets below L in the buffer.

The third item means that the waiting time exceeds w if the L long block in the queue, which k belongs to, is still not filled up until $w-\Delta$. $[e^{\boldsymbol{Q}t}]_i$ is the probability that the arrival process did not generate enough arrivals to leave this block until time t, if there were i data units in the block at the beginning. $\{k/L\}$ (where $\{\}$ denotes the remainder of the division) is the buffer position inside the L long block after the arrival.

To obtain the waiting time distribution, we have to multiply $P_W(k, t_1, t_2)$ by the probability of the given k, t_1 and t_2 parameters.

Since the resulting expression is long, we split it to 3 parts to make it easier to follow. This splitting is according to the partitioning of the P1,P2,P3 cases in Section 2. In each part we produce the waiting time properties multiplied by their weights (q_i), and the sum of the weights used at the end to normalize the results (q_{n_i}).

P1 Last departure left the system above L ($i > L$, with probability p_i). Let assume a tagged packet arrival at time t (measured from the last service instant). The length of the queue will be the sum of the queue size at the last departure (i), the number of data units arrived before the tagged one (j), and the size of the arrived batch (k). The server is busy for at least $\Delta - t$, and the last fragment of the data waits at most $T-\Delta$ after the server becomes idle. In this case:

$$q_1 = \sum_{i=L}^{\infty} p_i \int_0^{\Delta} \sum_{j=0}^{\infty} P(j,t) \sum_{k=1}^{\infty} D_k \cdot P_W(i+j+k, \Delta - t, (T-\Delta)^+) dt \quad (10)$$

This expression considers the arrivals during Δ, thus the weight is the following (which can also be obtained from Eq. (10) by setting $P_W(i, t_1, t_2) = 1$):

$$q_{n_1} = \sum_{i=L}^{\infty} p_i \int_0^{\Delta} e^{Dt} dt D_A h$$

P2 The last departure left the system between 1 and $L-1$. Waiting time of arrivals during Δ can be calculated similarly to the previous case, but the queue can be still below L after Δ. Then the server becomes idle (reflected by the second term of Eq. 11), and the timer runs for at most $T-\Delta$ (since

it has been started at the beginning of the last service). Thus we have:

$$q_2 = \sum_{i=1}^{L-1} p_i \Bigg[\int_0^{\Delta} \sum_{j=0}^{\infty} P(j,t) \sum_{k=1}^{\infty} D_k \cdot P_W(i+j+k, \Delta - t, (T-\Delta)^+) dt + \\ + \sum_{\ell=0}^{L-i-1} P(\ell, \Delta) \int_0^{(T-\Delta)^+} \sum_{j=0}^{L-\ell-i-1} P(j,t) \sum_{k=1}^{\infty} D_k \cdot P_W(i+\ell+j+k, 0, T-\Delta-t) dt \Bigg] \tag{11}$$

Now we covered the arrivals during Δ, plus the arrivals until the packet was filled up totally or the timer elapsed. Thus q_{n_2} is:

$$q_{n_2} = \sum_{i=1}^{L-1} p_i \int_0^{\Delta} e^{Dt} dt D_A h + \sum_{i=1}^{L-1} p_i \sum_{\ell=0}^{L-i-1} P(\ell, \Delta) \sum_{j=0}^{L-\ell-i-1} \int_0^{T-\Delta} P(j,t) dt D_A h$$

P3 The last departure left the system empty. For Δ the server is still busy. We consider 2 subcases according to the time of the first arrival, that can happen before or after Δ. In both cases the timer is started at the first arrival.

· The first arrival occurs during Δ. The waiting time probability consists of 3 parts. The first corresponds to the first arrival. The second computes the other remaining arrivals in Δ (conditioned on the time point of the first arrival), and the third term considers the case when the buffer is still below L after Δ.

$$q_{3a} = p_0 \Bigg[\int_0^{\Delta} e^{D_0 t} \sum_{k=1}^{\infty} D_k P_W(k, \Delta - t, (T_{CU} - \Delta + t)^+) dt + \\ + \int_0^{\Delta} e^{D_0 t} \sum_{\ell=1}^{\infty} D_\ell \cdot \\ \cdot \int_0^{\Delta - t} \sum_{j=0}^{\infty} P(j,s) \sum_{k=1}^{\infty} D_k P_W(\ell + j + k, \Delta - t - s, (T - \Delta + t + s)^+) ds \, dt + \\ + \int_0^{\Delta} e^{D_0 t} \sum_{\ell=1}^{L-1} D_\ell \sum_{i=0}^{L-\ell-1} P(i, \Delta - t) \cdot \\ \cdot \int_0^{(T-\Delta+t)^+} \sum_{j=0}^{L-\ell-i-1} P(j,s) \sum_{k=1}^{\infty} D_k P_W(\ell + i + j + k, 0, T - \Delta + t - s) ds \, dt \Bigg] \tag{12}$$

The weight of this case consists of the arrivals during Δ and – if the packet is still not ready after Δ – the arrivals till the packet is filled up or the timer elapses:

$$q_{n_{3a}} = p_0 \int_0^{\Delta} e^{Dt} dt D_A h + \int_0^{\Delta} e^{D_0 t} \sum_{k=1}^{L-1} D_k \sum_{\ell=0}^{L-k-1} P(\ell, \Delta - t) \cdot \cdot \sum_{j=0}^{L-\ell-k-1} \int_0^{(T-\Delta+t)^+} P(j,s) ds \; dt \; D_A h \tag{13}$$

· After Δ the buffer is still empty. The first term is related to the waiting time of the first arrival. If the batch size of the first arrival is less than L, other arrivals may come at time t in the $(0, T)$ interval (second term), experiencing $T - t$ remaining timer value. Thus, q_{3b} and $q_{n_{3b}}$ are:

$$q_{3b} = p_0 e^{D_0 \Delta} (-D_0)^{-1} \sum_{k=1}^{\infty} D_k P_W(k, 0, T) + + p_0 e^{D_0 \Delta} (-D_0)^{-1} \sum_{\ell=1}^{L-1} D_\ell \int_0^T \sum_{j=0}^{L-\ell-1} P(j,t) \sum_{k=1}^{\infty} D_k P_W(\ell + j + k, 0, T - t) dt, \tag{14}$$

and

$$q_{n_{3b}} = p_0 e^{D_0 \Delta} (-D_0)^{-1} D_A h + p_0 e^{D_0 \Delta} (-D_0)^{-1} \sum_{\ell=1}^{L-1} D_\ell \sum_{j=0}^{L-\ell-1} \int_0^T P(j,t) dt D_A h. \tag{15}$$

Finally the waiting time distribution is computed by:

$$P(W > w) = \frac{q_1 + q_2 + q_{3a} + q_{3b}}{q_{n_1} + q_{n_2} + q_{n_{3a}} + q_{n_{3b}}}$$

The ith moment of the waiting time can be obtained similarly since knowing the same three parameters (k, t_1, t_2) the waiting time moments are easy to compute.

5 The Analysis Algorithm

Putting together the results of Sections 3 and 4, the algorithm works as follows:

(i) Construct the BMAP description of the traffic source.
If there are many different traffic sources, each has to be modelled by

individual BMAP, and aggregated together by Kronecker operations. For example, if $D_k^{(i)}$ are the BMAP matrices of traffic source i and we have P independent BMAPs, the aggregated BMAP is constructed as:

$$
\begin{aligned}
D_0 &= D_0^{(1)} \oplus D_0^{(2)} \oplus \cdots \oplus D_0^{(P)} \\
D_1 &= D_1^{(1)} \otimes I \otimes \cdots \otimes I + I \otimes D_1^{(2)} \otimes \cdots \otimes I + \cdots + I \otimes I \otimes \cdots \otimes D_1^{(P)} \\
D_2 &= D_2^{(1)} \otimes I \otimes \cdots \otimes I + I \otimes D_2^{(2)} \otimes \cdots \otimes I + \cdots + I \otimes I \otimes \cdots \otimes D_2^{(P)} \\
&\quad + D_1^{(1)} \otimes D_1^{(2)} \otimes I \otimes \cdots \otimes I + D_1^{(1)} \otimes I \otimes D_1^{(3)} \otimes \cdots \otimes I + \ldots \\
&\quad + I \otimes I \otimes \cdots \otimes D_1^{(P-1)} \otimes D_1^{(P)}
\end{aligned}
$$

... and so on.

(ii) Construct $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$ matrices according to Section 3. Numerical problems might arise during this step. To overcome these, we used the following procedure.
- The computation of $P(k,t)$ matrices is according to Appendix C. The number of computed $P(k,\Delta)$ matrices determine the size of $\boldsymbol{\mathcal{A}}$ and $\boldsymbol{\mathcal{B}}$.
- In the computation of $\boldsymbol{\Pi}(t)$ the two quantities, $\int_0^t e^{\boldsymbol{Q}x}dx$ and $e^{\boldsymbol{Q}t}$, can be efficiently computed together using randomization (see Appendix B).
- In $\hat{\boldsymbol{\mathcal{B}}}$ the integral can be evaluated by the trapezoid rule or by the Simpson rule. The $e^{D_0 t}$ matrix is calculated by randomization again.

(iii) Solve the steady state of the obtained M/G/1 type Markov chain. We used a method presented in [7], summarized in Appendix D. The computation of the steady state probabilities is stopped when they get small (e.g., $p_i < 10^{-6}$).

(iv) Compute the waiting time distribution according to Appendix A. All arising numerical issues are already mentioned. We note that the matrices of which the matrix exponential is computed are either small (of size $m \times m$) or do not depend on the integral variable, so instead of numerical integration they can be computed by randomization. The only exception is q_{3a}, where the numerical integration is unavoidable.

6 Numerical Results

We implemented the above discussed computation method in MATLAB, and also developed a simulation tool in OMNeT++ ([2]) to check the correctness of both the expressions and the MATLAB implementation. The figures show both the MATLAB (with lines) and the simulation results (indicated with points).

We evaluated the following example. There are N traffic sources with on-off behavior. The on and the off periods are exponentially distributed, with parameters α and β, respectively. During the on period each traffic

source generates traffic with exponentially distributed inter arrival times, with intensity λ. At each arrival instant the source generates k data units with probability $p(1-p)^{L-k}/(1-(1-p)^L)$, which is a geometrical distribution with parameter p truncated at L. The multiplexer assembles packages containing a maximum of L data units.

These parameters have the following values through the examples:

$L =$	8
$\alpha =$	1/1000
$\beta =$	1/1500
$\lambda =$	50
$N =$	5
$p =$	(changing)
$\Delta =$	(changing)

Figure 3 shows the distribution of the waiting time with different timer values. We used parameters $p = 0.8$ and $\Delta = 2$ in this example. To magnify the effect of timer, the parameters have been chosen to give low load. The figure verifies the assumption that the higher the timer value is, the bigger is the waiting time. Furthermore, two jumps can be observed in the curves,

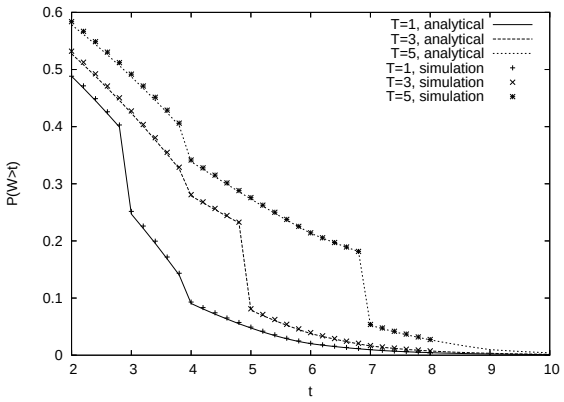

Fig. 3. Distribution of the waiting time

one at 2Δ, the one at $T + \Delta$. The explanation of the jumps is the following. Let assume there is a given amount of data ($< L$) waiting in the queue, and the server is idle (timer is running). With a newly arrived batch (the tagged arrival) the queue size exceeds L, so a packet is compiled and the service starts. The second fragment of the tagged arrival (that has not been included in the packet) waits exactly Δ if enough arrivals are coming during the server occupancy time, or it waits exactly T, if there are not enough arrivals in T. These two cases (increased by the time spent in the server) provide the jumps on the distribution function. The jump at $T + \Delta$ also contains the probability

that a batch arrives into an empty system, and there were not enough arrivals until the timer elapsed.

In our second example the service time is varying between 0 and 5, and we examine the probability of exceeding the $w = 5$ waiting time. Figure 4 (a) and (b) correspond to the $p = 0.4$ and to the $p = 0.8$ setting. There are

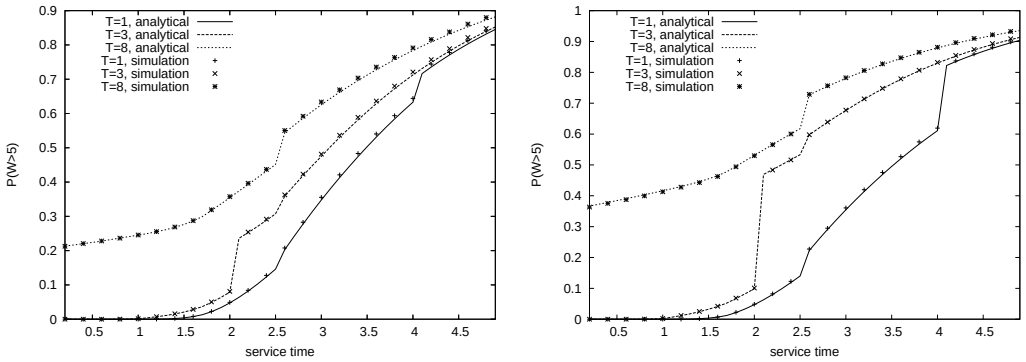

Fig. 4. $P(W > 5)$ vs. service time

jumps again at $\Delta = w/2 = 2.5$ and at $\Delta = w - T$. The explanation is the same as in the first example, the first jump is related to the jump at 2Δ on the distribution function, and the second one is related to the one at $T + \Delta$. It is an interesting feature of the system that the $P(W > w)$ probability does not reduce to zero by decreasing the service time to zero (i.e., increasing the output capacity to infinity) when $T > w$ as it is in Figure 4 with $T = 8$.

In all the examples the computation of one point of the waiting time distribution with our MATLAB implementation took few seconds on a Pentium4 2.4 GHz machine, while the simulation could give acceptable results only after about 1 minute. The only exception is the last example, where the execution took a longer time (1-2 minutes) at $N = 15$. Of course, with a C implementation the speed of our analytical algorithm can be increased substantially.

7 Conclusion

In this paper we provided a model and an algorithm to compute the the waiting time distribution of BMAP/D/1-Timer multiplexer queues. The proposed algorithm is designed for an efficient numerical implementation.

The last section presents a numerical example. The analytical results showed a rather perfect fit with the simulation results. In this example we also investigated the special features of the waiting time distribution of such queues.

A Numerical method for the waiting time distribution

Expressions Eq. (10), (11), (12) and (14) are difficult to compute efficiently, because numerical integration is usually slow and inaccurate. Fortunately most of the numerical integrations can be eliminated. Substituting Eq. (9) into the above mentioned expressions, most of the integrals will take the form $\int_a^b e^{\boldsymbol{Q}t}dt$, which can be efficiently computed by randomization (see Appendix B). In favor of completeness, here we give q_1, q_2, q_{3a} and q_{3b} with all possible numerical integrations eliminated.

Substituting Eq. (9) into Eq. (10), we get:

$$q_1 = \sum_{i=L}^{\infty} p_i \sum_{j=0}^{\infty}\sum_{k=1}^{\infty}\Big(\int_0^{s_1} P(j,t)dtD_k h + \int_{s_1}^{s_2} P(j,t)dtD_k \left[e^{\boldsymbol{Q}(w-\Delta)}h\right]_{\{\frac{i+j+k}{L}\}}\Big). \tag{A.1}$$

The limits of the integral s_1 and s_2 are derived from the definition of $P_W(k,t_1,t_2)$ (see Eq. (9)); below s_1 $P_W(\bullet)$ equals to 1, and above s_2 it equals to zero, thus:

$$\begin{aligned} s_1 &= \min\{\Delta, \Big(\Delta + \lceil\frac{i+j+k}{L}\rceil\Delta - w\Big)^+\} \\ s_2 &= \min\{\Delta, \Big(\Delta + \lceil\frac{i+j+k}{L}\rceil\Delta + (T_{CU}-\Delta)^+ - w\Big)^+\} \end{aligned} \tag{A.2}$$

Similarly, Eq. (11) can be transformed to the following:

$$\begin{aligned} q_2 = &\sum_{i=1}^{L-1} p_i \sum_{j=0}^{\infty}\sum_{k=1}^{\infty}\Big(\int_0^{s_1} P(j,t)dtD_k h + \int_{s_1}^{s_2} P(j,t)dtD_k \left[e^{\boldsymbol{Q}(w-\Delta)}h\right]_{\{\frac{i+j+k}{L}\}}\Big) + \\ &\sum_{i=1}^{L-1} p_i \sum_{l=0}^{L-i-1} P(l,\Delta) \sum_{\ell=0}^{L-i-1}\Bigg[\sum_{k=1}^{L-i-\ell-j-1}\int_0^{((T-\Delta)^+-w+\Delta)^+} P(j,t)dtD_k \left[e^{\boldsymbol{Q}(w-\Delta)}h\right]_{\{\frac{i+j+k+\ell}{L}\}} + \\ &\sum_{k=L-i-\ell-j}^{\infty}\Bigg(I_{\{w<\lceil\frac{i+j+k+\ell}{L}\rceil\Delta\}}\cdot\int_0^{(T-\Delta)^+} P(j,t)dtD_k h + \\ &I_{\{\lceil\frac{i+j+k+\ell}{L}\rceil\Delta\le w<\lceil\frac{i+j+k+l}{L}\rceil\Delta+(T-\Delta)^+\}}\cdot\int_0^{(T-\Delta)^+} P(j,t)dtD_k \left[e^{\boldsymbol{Q}(w-\Delta)}h\right]_{\{\frac{i+j+k+l}{L}\}}\Bigg)\Bigg]. \end{aligned} \tag{A.3}$$

Where the integral limits s_1 and s_2 are already defined by Eq. (A.2).

q_{3a} can be transformed to the following – much longer but numerically

better computable – expression:

$$
\begin{aligned}
q_{3a} = p_0 \sum_{k=1}^{L-1} \Bigg(& \int_0^{(2\Delta-w)^+} e^{D_0 t} dt D_k h + \\
& + I_{\{w<T+\Delta\}} \int_{\max\left((2\Delta-w)^+,(\Delta-T)^+\right)}^{\Delta} e^{D_0 t} dt D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h\right]_k \Bigg) + \\
+ p_0 \sum_{k=L}^{\infty} \Bigg(& \int_0^{s_1} e^{D_0 t} dt D_k h + \int_{s_1}^{s_2} e^{D_0 t} dt D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h\right]_{\{\frac{k}{L}\}} \Bigg) + \\
+ p_0 \int_0^{\Delta} e^{D_0 t} dt \sum_{\ell=1}^{\infty} D_\ell \sum_{j=0}^{\infty} \sum_{k=1}^{\infty} \Bigg[& I_{\{\ell+j+k<L\}} \Bigg(\int_0^{(2\Delta-w-t)^+} P(j,s) ds D_k h + \\
& + \Bigg(\int_{(2\Delta-w-t)^+}^{(\Delta-t-T)^+} P(j,s) ds + \\
& + I_{\{w<T+\Delta\}} \int_{(\Delta-t-T)^+}^{\Delta-t} P(j,s) ds \Bigg) D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h\right]_{k+\ell+j} \Bigg) + \\
& + I_{\{\ell+j+k\geq L\}} \left(\int_0^{s_1-t} P(j,s) ds D_k h + \int_{s_1-t}^{s_2-t} P(j,s) ds D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h\right]_{\{\frac{l+j+k}{L}\}} \right) \Bigg] dt \\
+ p_0 \int_0^{\Delta} e^{D_0 t} \sum_{\ell=1}^{L-1} D_\ell \sum_{i=0}^{L-\ell-1} P(i, \Delta - t) \sum_{j=0}^{L-\ell-i-1} \Bigg[& \\
\sum_{k=1}^{L-\ell-i-j-1} \int_0^{(T+t-w)^+} & P(j,s) ds \left[e^{\boldsymbol{Q}(w-\Delta)} h\right]_{k+l+i+j} + \\
+ \sum_{k=L-\ell-i-j}^{\infty} \Bigg(& I_{\{w<\lceil \frac{\ell+i+k+\ell}{L} \rceil \Delta\}} \int_0^{(T-\Delta+t)^+} P(j,s) ds D_k h + \\
& + I_{\{\lceil \frac{\ell+i+k+\ell}{L} \rceil \Delta \leq w < \lceil \frac{\ell+i+k+\ell}{L} \rceil \Delta + (T-\Delta)^+\}} \int_0^{(T-\Delta+t)^+} P(j,s) ds \, D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h\right]_{\{\frac{k+i+j+\ell}{L}\}} \Bigg] dt.
\end{aligned}
\tag{A.4}
$$

$q_{n_{3a}}$ can also be rewritten into a numerically more efficient form. First we rewrite Eq. (13) in matrix form:

$$
q_{n_{3a}} = p_0 \int_0^{\Delta} e^{Dt} dt D_A h + \int_0^{\Delta} e^{D_0 t} \left[D_1 \dots D_{L-1}\right] e^{\boldsymbol{Q}(\Delta-l)} \cdot \int_0^{(T-\Delta+t)^+} e^{\boldsymbol{Q}s} ds \, dt \begin{bmatrix} D_A h \\ \vdots \\ D_A h \end{bmatrix}.
\tag{A.5}
$$

With some algebra it simplifies to:

$$q_{n_{3a}} = p_0 \left(\int_0^\Delta e^{Dt} dt D_A h + \left[[P(1,\Delta)\ P(2,\Delta)\ \dots\ P(L-1,\Delta)] - \right.\right.$$
$$\left.\left. - \int_0^\Delta e^{D_0 t} dt \left[D_1\ \dots\ D_{L-1} \right] e^{\boldsymbol{Q}T} \right] (-\boldsymbol{Q})^{-1} \begin{bmatrix} D_A h \\ \vdots \\ D_A h \end{bmatrix} \right),$$

which does not contain any numerical integral. For q_{3b} we have the following expression:

$$\begin{aligned} q_{3b} = p_0 e^{D_0\Delta}(-D_0)^{-1} \Bigg[& \sum_{k=1}^{L-1} I_{\{w<T+\Delta\}} \cdot D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h \right]_k + \sum_{k=L}^{\infty} \Big(I_{\left\{w<\lceil \frac{k}{L} \rceil \Delta\right\}} \cdot D_k h + \\ & + I_{\left\{ \lceil \frac{k}{L} \rceil \Delta < w < \lceil \frac{k}{L} \rceil \Delta + (T-\Delta)^+ \right\}} \cdot D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h \right]_{\left\{\frac{k}{L}\right\}} \Big) + \\ & + \sum_{\ell=1}^{L-1} D_l \sum_{j=0}^{L-\ell-1} \Bigg[\sum_{k=1}^{L-\ell-j-1} \int_0^{(T+\Delta-w)^+} P(j,t) dt D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h \right]_{\ell+j+k} + \\ & + \sum_{k=L-\ell-j}^{\infty} \Big(I_{\left\{w<\lceil \frac{\ell+j+k}{L} \rceil \Delta\right\}} \cdot \int_0^T P(j,t) dt D_k h + \\ & + I_{\left\{ \lceil \frac{\ell+j+k}{L} \rceil \Delta < w < \lceil \frac{\ell+j+k}{L} \rceil \Delta + (T-\Delta)^+ \right\}} \cdot \int_0^T P(j,t) dt D_k \left[e^{\boldsymbol{Q}(w-\Delta)} h \right]_{\left\{\frac{\ell+j+k}{L}\right\}} \Big) \Bigg] \Bigg]. \end{aligned}$$

B Randomization Algorithm

The randomization algorithm is an efficient and numerically stable method to compute $\boldsymbol{P}(t) = e^{\boldsymbol{Q}t}$ and $\boldsymbol{L}(t) = \int_0^t e^{\boldsymbol{Q}\tau} d\tau$ (see [4]). First we have to convert our continuous time generator $\boldsymbol{Q}$ to a stochastic matrix $\boldsymbol{Q}^*$:

$$q = \max_i |q_{ii}|$$
$$\boldsymbol{Q}^* = \boldsymbol{Q}/q + \boldsymbol{I}$$

Then, the following expressions provide the two quantities we need:

$$\boldsymbol{P}(t) = \sum_{i=0}^{\infty} \boldsymbol{Q}^{*i} e^{-qt} \frac{(qt)^i}{i!}$$
$$\boldsymbol{L}(t) = \frac{1}{q} \sum_{i=0}^{\infty} \boldsymbol{Q}^{*i} \sum_{i+1}^{\infty} e^{-qt} \frac{(qt)^j}{j!}$$

These can be computed together in the same iterative algorithm. With large $\boldsymbol{Q}$ matrices the most expensive step is the matrix multiplication (to calculate the ith power of $\boldsymbol{Q}$ from its $i-1$th), has to be computed only once. Then, weighting $\boldsymbol{Q}^{*i}$ with appropriate poisson coefficients gives $\boldsymbol{P}(t)$, and $\boldsymbol{L}(t)$.

If $e^{\boldsymbol{Q}t}$ is not needed, only $a \cdot e^{\boldsymbol{Q}t}$ or $e^{\boldsymbol{Q}t} \cdot b$ has to be computed (with a

and b being vectors), then the numerical solution is much faster, because the matrix-matrix multiplications are replaced with vector-matrix multiplications.

A further advantage of this algorithm is that its error can be bounded in advance, since $\boldsymbol{Q}^{*i}$ is stochastic. If the summation runs up to N, the error is less than $1 - \sum_{i=0}^{N} e^{-qt}\frac{(qt)^i}{i!}$.

C Computation of $P(k,t)$ matrices

The infinite matrix $\boldsymbol{\mathcal{Q}}$ (in Section 2) is the generator of the arrival process. Thus, its transition probability matrix at time t is:

$$\begin{bmatrix} P(0,t) & P(1,t) & P(2,t) & \dots \\ & P(0,t) & P(1,t) & \dots \\ & & P(0,t) & \dots \\ & & & \ddots \end{bmatrix} = e^{\boldsymbol{\mathcal{Q}}t}.$$

Multiplying both sides by $[\ I_{m\times m}\ \ 0\ \ 0\ \ \dots\ \ 0\ ]$ we get:

$$[\ P(0,t)\ \ P(1,t)\ \ \dots\ ] = [\ I_{m\times m}\ \ 0\ \ 0\ \ \dots\ ] \cdot e^{\boldsymbol{\mathcal{Q}}t}$$

These infinite vectors and matrices have to be cut at index K, which should be chosen to cover "most" arrivals during t. The block vector randomization method (see Appendix B) can be used to calculate $P(k,t)$. Alternatively, it is possible to use the following randomization based recursive relation:

$$P^{(n)}(k,t) = \sum_{i=0}^{k} P^{(n-1)}(i,t) \cdot D'_{k-i} \cdot e^{-qt}\frac{(qt)^i}{i!},$$
$$P^{(0)}(0,t) = I_{m\times m} \cdot e^{-qt}, \quad P^{(0)}(k,t) = 0_{m\times m} \ \text{ if } k > 0,$$

where $q = \max_{i\in S}[D_0]_{i,i}$, $D'_0 = D_0/q + I$ and $D'_k = D_k/q,\ k > 0$. The further n is increased, the more exact result we get: as written in Appendix B, the error is less than $1 - \sum_{i=0}^{n} e^{-qt}\frac{(qt)^i}{i!}$.

D Buffer Length Distribution Embedded at PDU Departure Instants

For this problem we use a numerical method presented in [7]. Here we give a short summary about this method, without going into details. Vectors $\boldsymbol{x}_i$ are computed by the following recursion, if $i \geq 1$:

$$\boldsymbol{x}_i = \left(\boldsymbol{x}_0\widehat{\boldsymbol{B}}_i + \sum_{j=1}^{i-1}\boldsymbol{x}_j\widehat{\boldsymbol{A}}_{i+1-j}\right)(\boldsymbol{I} - \widehat{\boldsymbol{A}}_1)^{-1} \tag{D.1}$$

This recursion is commonly referred as Ramaswami formula, which is numerically stable because it includes only additions and multiplications. The iteration can be stopped if the elements of the result vector are very close to zero. Matrices $\widehat{\boldsymbol{A}}_i$ and $\widehat{\boldsymbol{B}}_i$ are obtained by the following backward recursion, using the fundamental matrix G:

$$\widehat{\boldsymbol{B}}_k = \boldsymbol{B}_k + \widehat{\boldsymbol{B}}_{k+1}\boldsymbol{G} \tag{D.2}$$

$$\widehat{\boldsymbol{A}}_k = \boldsymbol{A}_k + \widehat{\boldsymbol{A}}_{k+1}\boldsymbol{G} \tag{D.3}$$

The recursion starts at the index where $\widehat{\boldsymbol{B}}_K = 0$ and $\widehat{\boldsymbol{A}}_K = 0$ ($\boldsymbol{B}_{K-1}$ or $\boldsymbol{A}_{K-1}$ are the last non zero matrices).

The computation of matrix $\boldsymbol{G}$ (also known as fundamental matrix) is the most critical point in the solution of M/G/1 type Markov chains. We suggest the following simple iteration to obtain it, but we also note that when the utilization approaches 1, there are much faster (but more complex) methods (see [3]):

$$\boldsymbol{G}_0 = \boldsymbol{0}$$

$$\boldsymbol{G}_{k+1} = (\boldsymbol{I} - \boldsymbol{A}_1)^{-1} \sum_{\nu=0,\nu\neq 1}^{\infty} \boldsymbol{A}_\nu \boldsymbol{G}_k^\nu$$

This iteration can be stopped when the sum of the elements in each row of $\boldsymbol{G}$ are close to 1.

It only remains to compute $\boldsymbol{x}_0$:

$$\boldsymbol{x}_0 = (\boldsymbol{\kappa}\tilde{\boldsymbol{\kappa}})^{-1}\boldsymbol{\kappa}.$$

Where $\boldsymbol{\kappa}$ is the solution of $\boldsymbol{\kappa} = \boldsymbol{\kappa}\boldsymbol{K}$, with $\boldsymbol{K}$ defined by:

$$\boldsymbol{K} = \sum_{\nu=0}^{\infty} \boldsymbol{B}_\nu \boldsymbol{G}^\nu.$$

And $\tilde{\boldsymbol{\kappa}}$ is computed by:

$$\boldsymbol{A}^* = \sum_{\nu=0}^{\infty} \boldsymbol{A}_\nu \sum_{k=0}^{\nu-1} \boldsymbol{G}^k$$

$$\boldsymbol{B}^* = \sum_{\nu=0}^{\infty} \boldsymbol{B}_\nu \sum_{k=0}^{\nu-1} \boldsymbol{G}^k$$

$$\tilde{\boldsymbol{\kappa}} = \boldsymbol{h} + \boldsymbol{B}^*(\boldsymbol{I} - \boldsymbol{A}^*)^{-1}\boldsymbol{h}.$$

References

[1] ITU-T Recommendation I.363.2, B-ISDN ATM Adaptation Layer Type 2 Specification, Toronto, 1997.

[2] OMNeT++ Discrete Event Simulation System, http://www.omnetpp.org.

[3] Dario Bini and Beatrice Meini. On the Solution of a Nonlinear Matrix Equation arising in Queueing Problems. *SIAM Journal on Matrix Analysis and Applications*, 17(4):906–926, 1996.

[4] D. Gross and D. Miller. The randomization technique as a modeling tool and solution procedure for transient Markov processes. *Operations Research*, 32:343–361, 1984.

[5] A. Horváth, G. I. Rózsa, and M. Telek. A map fitting method to approximate real traffic behaviour. In *8th IFIP Workshop on Performance Modelling and Evaluation of ATM & IP Networks*, pages 32/1–12, Ilkley, England, July 2000.

[6] G. Latouche and V. Ramaswami. *Introduction to Matrix Analytic Methods in Stochastic Modeling*. ASA-SIAM, 1999.

[7] Marcel F. Neuts. *Structured Stochastic Matrices of M/G/1 Type and their Applications*. Dekker, 1989.

[8] T. Ryden. An em algorithm for estimation in Markov modulated Poisson processes. *Computational statist. and data analysis*, 21:431–447, 1996.

[9] Hiroshi Saito. Performance evaluation of AAL2 voice multiplexer. *Performance Evaluation*, 42(1):57–83, 2000.