A Numerical Framework for Solving Discrete Finite Markov Models Applied to the AAL-2 Protocol

Michael Menth, Department of Distributed Systems, Institute of Computer Science, University of Würzburg, Am Hubland, D-97074 Würzburg, FRG.
Notker Gerlich[†‡], Siemens AG, ICN M NT 14, D-81359 München, FRG.

Abstract

In performance analysis of modern communication systems discrete Markov modeling techniques have become increasingly important. This paper presents a numerical framework for solving large discrete and finite Markov models in an efficient way. The first part outlines the basic theory and establishes the numerical framework. This framework is applied to a performance analysis of the ATM Adaptation Layer type 2 protocol in the second part.

1 Introduction

Markov models play an important role in performance evaluation of communication systems since the pioneering works of A. K. Erlang at the beginning of this century. The digital revolution brought about communication technologies that base on a small number of fixed size data units like the cells in the Asynchronous Transfer Mode (ATM) [1] system. Given discrete basic time and data units in the system that is to be modeled, a discrete model offers itself as the basis for performance evaluation studies.

A number of recent performance studies that base on a discrete Markov model [2–18] exhibit the same underlying analysis pattern: The state evolution of a discrete time Markov chain is expressed by a recursive equation, much like the well-known Lindley Equation [19] but in the discrete domain. After having translated the recursive equation of random variables into an iterative procedure on probability mass functions the average state distribution is computed. Based on this distribution performance measures like loss and delay probabilities may be calculated. The sketched method, which is referred to as Discrete Time Analysis (DTA) by some authors, provides numerical results only; it does not obtain closed-form formulae.

The modelers skill consists in devising the recursive equation. Once the equation is found, there is the problem of turning the equation into an efficient numerical program. In the references cited above the iterative procedure is achieved in different ways using convolution and transformation operators.

This paper aims at a more systematic way of deriving the numerical program from the recursive equations. Specifically, it presents an efficient numerical framework that implements the iterative procedure directly from the recursive equation. The numerical framework is derived by formalizing the approaches of [2–18] in Section 2. The formalization exhibits that all the studies above utilize a backward equation, i.e., the modeler has to find the inverse of the state transition function to set up the numerical program. If the backward equation is rearranged into a forward equation, the program can be derived syntactically. Additionally, we enhance the discrete time analysis method towards the solution of cyclo-stationary systems. The enhancement is required for the analysis of the ATM adaptation layer type 2 protocol in Section 3 by which we demonstrate the application of the framework.

[†] Work done while the author was with the Department of Distributed Systems, Institute of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg FRG.

[‡] The author acknowledges the support of the Deutsche Forschungsgemeinschaft (DFG) under grant Tr-257/3.

2 Establishing the Framework

We illustrate the formalization of the DTA approaches [2–18] by the DTA of the discrete time GI$^{[X]}$/D/1–S queuing system as presented by Tran-Gia and Ahmadi [15]. We briefly recall their analysis in the first section; the details can be found in the original publication. In the second section we formalize the DTA approach. The formalization leads to the development of the numerical framework we aim at.

2.1 Example DTA: The GI$^{[X]}$/D/1–S Queue

The discrete time GI$^{[X]}$/D/1–S queuing system has a finite queue size, a constant service time, and general interarrival times of batches with a general batch size distribution. The unfinished work process of the system is a Discrete Time Markov Chain (DTMC). Tran-Gia and Ahmadi [15] represent the state evolution of the DTMC by a recursive equation that relates the state of the unfinished work process observed immediately prior to the arrival of a batch to the state observed upon arrival of the preceding batch:

$$U_{n+1} \quad = \quad \max[\,\min(U_n + B_n,\ S) - A_n,\ 0],$$

where the following notation is employed:

- U_n random variable for the unfinished work immediately prior to the arrival instant of the n-th batch;
- B_n random variable for the size of the n-th batch;
- A_n random variable for the time interval between the arrival instants of the n-th and $(n+1)$-th batch;
- S capacity of the queue.

Note that with the discrete time unit equal to the constant service time, A_n customers can be served while A_n time units pass by.

The iterative algorithm for calculating the successive state distributions* is stated as

$$u_{n+1}(k) \quad = \quad \pi_0[\,\pi^S(u_n(k) \circledast b_n(k)) \circledast a_n(-k)],$$

where the sweep operators $\pi_0(\cdot)$ and $\pi^S(\cdot)$ are defined by

$$\pi_0(z(k)) \quad = \quad \begin{cases} 0 & k < 0 \\ \sum_{i=-\infty}^{0} z(i) & k = 0 \\ z(k) & k > 0 \end{cases}$$

$$\pi^S(z(k)) \quad = \quad \begin{cases} z(k) & k < S \\ \sum_{i=S}^{\infty} z(i) & k = S \\ 0 & k > S \end{cases}$$

and the $\circledast$-symbol denotes the discrete convolution

$$\begin{aligned} z(k) \quad &= \quad z_1(k) \circledast z_2(k) \\ &= \quad \sum_{j=-\infty}^{\infty} z_1(k-j) \cdot z_2(j). \end{aligned} \qquad (1)$$

Provided A_n and B_n are each independent and identically distributed, the DTMC is homogeneous and the iterative algorithm converges to the limiting distribution $u(k) = \lim_{n\to\infty} u_n(k)$, which equals the average state distribution (apart from pathological cases). From $u(k)$ the authors derive the performance measures of the system.

2.2 Formalization

The decisive part of any discrete time analysis is the iterative computation of the stationary state distribution of the DTMC. It is this part which we are interested in formalizing it.

The required stationary state distribution x always exists [20] and is obtained by taking the limit of the average of the successive state distributions x_n observed at discrete time instants $n = 0, 1, \ldots$

$$x \quad = \quad \lim_{N\to\infty} \frac{1}{N} \sum_{n=0}^{N-1} x_n \qquad (2)$$

In case of an aperiodic DTMC, the limiting state distribution of x_n exists and consequently equals the stationary state distribution. Taking the limit of x_n converges faster than computing Equation (2) and, therefore, the limit of x_n is usually taken as the stationary state distribution. A fast computation of the stationary state distribution for a periodic DTMC will be described later in this section.

*We use the term "distribution" shorthand for probability mass function.

However, in both cases before taking the limit the successive state distributions need to be computed. In the discrete time analysis approach an iteration is applied to this end. In the following we formalize this iterative procedure.

For a DTMC, we denote the discrete state space by $\mathcal{X}$. The system is described by the discrete random variable $X_n \in \mathcal{X}$ at the discrete *renewal points* $\{t_n | n \in \mathbb{N}\}$ which satisfy the memoryless property

$$\begin{aligned}\Pr(X_{n+1} = s_{n+1} \,|\, X_n = s_n, \ldots, X_0 = s_0) \\ = \Pr(X_{n+1} = s_{n+1} \,|\, X_n = s_n). \quad (3)\end{aligned}$$

Thus, $\{X_n | n \in \mathbb{N}\}$ is one realization of the DTMC. In a homogeneous DTMC the (single-step) transition probabilities $p_n(i,j) = \Pr(X_{n+1} = j \,|\, X_n = i)$ are independent of n and are consequently written as $p(i,j)$. The state transition matrix is denoted by

$$P \;=\; \big[p(i,j)\big]. \qquad (4)$$

The first step of a discrete time analysis consists in embedding a DTMC into the system evolution. That means to define a discrete system state X and to identify discrete time instants where the memoryless property holds for the evolution of X. In our example above, the DTMC is embedded at the batch arrival instants; its state is defined by the unfinished work U.

The further evolution of the system state depends on the current state and on exterior factors. It must be independent of the predecessors of the current state for the memoryless property (3) to hold. We summarize the exterior factors by a system influencing variable Y that has discrete space $\mathcal{Y}$. In general, both the state variable X and the influencing variable Y may be composite variables. In our above example, the influencing variable Y consists of two components: the interarrival time A and the batch size B.

Due to the memoryless property of the DTMC the state transition from X_n to X_{n+1} depends only on X_n and Y_n. Since Y_n is an identical and independently distributed random variable, its subscript can be omitted. The next DTA step is the representation of the state evolution by a relation that is recursive in the state variable X. Formally, it means defining a recursive *state transition function* $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{X}$

$$X_{n+1} \;=\; f(X_n, Y).$$

In our above example, the state transition function was defined by

$$\begin{aligned}U_{n+1} &= f(U_n, A, B) \\ &= \max[\,\min(U_n + B,\; S) - A,\; 0].\end{aligned}$$

Often it is possible to identify further, say r, discrete time instants between the renewal points used so far where the memoryless property holds. Denoting the states of the DTMC between instants n and $n+1$ by $X_n^i,\ 0 \le i < r$ we may derive state transition functions

$$\begin{aligned}X_n^{i+1} &= f^i(X_n^i, Y); \quad 0 \le i < r \\ X_{n+1}^0 &= f^{r-1}(X_n^{r-1}, Y),\end{aligned}$$

where X_n^0 are the formerly used renewal points X_n. The state transition function f is the composition of the transition functions f^i

$$f \;=\; f^{r-1} \circ f^{r-2} \circ \cdots \circ f^0. \qquad (5)$$

In our example, we can identify Markov points immediately before and immediately after the batch arrival instants according to [15]. The function f^0 describes what happens to the unfinished work upon arrival of a batch, i.e., the transition from immediately before to immediately after the n-th arrival instant; the function f^1 describes the server working off the unfinished work between arrivals, i.e., the transition from immediately after the n-th arrival to immediately prior to the $(n+1)$-th arrival:

$$\begin{aligned}U_n^1 &= f^0(U_n^0, B) = \min(U_n^0 + B, S) \\ U_{n+1}^0 &= f^1(U_n^1, A) = \max(U_n^1 - A, 0).\end{aligned}$$

The advantage of introducing the additional renewal points is obvious: it simplifies the setup of the state transition function f. Furthermore, it reduces the computational complexity of the numerical program as will become apparent later.

The next step is to turn the state transition function into an efficient numerical program. The program computes the successive state distributions

$x_n(k),\ n = 0, 1, \ldots,$ by iteration and, thus, requires an equation which is recursive in the state distribution. The numerical programs of the DTA studies [2–18] follow the law of total probability

$$x_{n+1}(k) = \sum_{i \in \mathcal{X}, j \in \mathcal{Y}} \Pr(X_{n+1} = k \mid X_n = i \wedge Y = j) \cdot x(i) \cdot y(j), \quad (6)$$

without stating the equation explicitly. Since the conditional probability $\Pr(X_{n+1} = k \mid X_n = i \wedge Y_n = j)$ equals 1 if $k = f(i,j)$ and 0, otherwise, we get

$$x_{n+1}(k) = \sum_{\{(i,j) \mid f(i,j)=k\}} x_n(j) \cdot y_n(i). \quad (7)$$

A good example is the use of the discrete convolution in the GI$^{[X]}$/D/1–S analysis above (1):

$$\begin{aligned} z(k) &= \sum_{\{(i,j) \mid f(i,j)=k\}} z_1(i) \cdot z_2(j), \quad \text{where} \\ f(i,j) &= i + j \end{aligned}$$

since the convolution is the numerical program that corresponds to the sum of two independent random variables. Computing this sum requires the pre-image $\{(i,j) | f(i,j) = k\}$ and, hence, the inverse f^{-1}. Setting $i = k - j$ and varying i from $-\infty$ to ∞ yields the usual formula of the discrete convolution.

Indeed, requiring the inverse of the state transition functions is characteristic for all the operators employed in the numerical programs of the DTA studies. For this reason we call the approach taken a *backward* method. The problem with using the backward method in a systematic derivation of the numerical program is that the pre-image needs to be computed. That problem is hidden by employing operators like the discrete convolution. But using these operators fails with multi-dimensional state variables as in [3] or [6]. The need to avoid the computation of the pre-image provided the stimulus for our developing of the forward method which we turn our attention to in the next section.

2.3 The Forward Method

In Equation (7) the probability of state k is computed by summing the compound probabilities of the tuples (i,j) in the pre-image of k with respect to the transition function f. This was derived from Equation (6) and can be achieved by summing the compound probabilities $x_n(i) \cdot y(j), (i,j) \in \mathcal{X} \times \mathcal{Y}$, weighted by the according conditional probabilities to yield state k. Advantage can be taken of the fact that the conditional probability is either zero or one, i.e., each tuple contributes its probability to exactly one successor state. Therefore, the $|\mathcal{X}|$ successor state probabilities x_{n+1} can be computed during one traversal of $\mathcal{X} \times \mathcal{Y}$ by adding the compound probability of tuple $(i,j) \in \mathcal{X} \times \mathcal{Y}$ to the one of its successor state $f(i,j)$. This observation suggests the *forward iteration* (**Algorithm 1**): After initializing the successor distribution x_{n+1} with 0, the algorithm traverses $\mathcal{X} \times \mathcal{Y}$ adding $x(i) \cdot y(j)$ to the probability of state $f(i,j)$. Since the algorithm uses the transition function f directly we call the approach a *forward* method.

Input: state distribution x_n and influencing distribution y
  Initialize x_{n+1} with zeros
  for all $i \in \mathcal{X}$ **do**
    for all $j \in \mathcal{Y}$ **do**
      $x_{n+1}(f(i,j)) := x_{n+1}(f(i,j)) + x_n(i) \cdot y(j)$
    end for
  end for
Output: x_{n+1}

Algorithm 1: Forward Iteration

Implementing the forward iteration for our above example we get the GI$^{[X]}$/D/1–S forward iteration (**Algorithm 2**). If we employ the transition functions f^0 and f^1 for implementing the forward iteration the resulting algorithm (**Algorithm 3**) has two single loops instead of one double loop. It is obviously faster than the above iteration. In general, only some components of a composite influencing variable Y are relevant for a specific transition function f^i. In the above example, f^0 requires only component B and f^1 requires only component A of the influencing variable $Y = (A, B)$. In the extreme case, we may decompose f into r state transition functions in a system having an r-dimensional influence variable Y where each transition function depends only on a single compo-

nent of Y. Then the iteration algorithm consists of r single loops instead of a single r-fold loop. In other words, the complexity of the algorithm reduces from $O(|\mathcal{X}| \cdot \prod_{i=0}^{r-1} |\mathcal{Y}_i|)$ to $O(|\mathcal{X}| \cdot \sum_{i=0}^{r-1} |\mathcal{Y}_i|)$.

Input: state distribution u_n and
influencing distributions a and b
Initialize u_{n+1} with zeros
for $i := 0$ **to** S **do**
for $j := 1$ **to** $max(B)$ **do**
for $k := 1$ **to** $max(A)$ **do**
$u_{n+1}(f(i,j,k)) := u_{n+1}(f(i,j,k)) + u_n(i) \cdot b(j) \cdot a(k)$
end for
end for
end for
Output: u_{n+1}

Algorithm 2: GI$^{[X]}$/D/1–S Forward Iteration

Input: state distribution u_n^0,
influencing distributions a and b
Initialize distribution u_n^1 with zeros
for $i := 0$ **to** S **do**
for $j := 1$ **to** $max(B)$ **do**
$u_n^1(f^0(i,j)) := u_n^1(f^0(i,j)) + u_n^0(i) \cdot b(j)$
end for
end for
Initialize u_{n+1}^0 with zeros
for $i := 0$ to S **do**
for $j := 1$ to $max(A)$ **do**
$u_{n+1}^0(f^1(i,j)) := u_{n+1}^0(f^1(i,j)) + u_n^1(i) \cdot a(j)$
end for
end for
Output: u_{n+1}^0

Algorithm 3: GI$^{[X]}$/D/1–S Forward Iteration with Additional Renewal Points

2.4 Relationship to Other Methods

It is an important question how our numerical framework is related to other iterative methods for solving a DTMC (see the thorough treatment by Stewart [21]). The basis of this class of methods to compute the limiting distribution of the DTMC is the *power iteration* equation

$$x_{n+1} = x_n P$$

that requires the state transition matrix P (4). The algorithm for computing the transition matrix P from the transition function reveals the relation to the power iteration.

Input: influencing distribution y
Initialize P with zeros
for all $i \in \mathcal{X}$ **do**
for all $j \in \mathcal{Y}$ **do**
$p(i, f(i,j)) := p(i, f(i,j)) + y(j)$
end for
end for
Output: P

Algorithm 4: Transition Matrix

Employing the transition function leads to the algorithm traversing the non-zero entries of P only. It is comparable to using a sparse storage scheme for P where storing the row index is replaced by computing it by means of function f. By comparing both algorithms we observe that the forward iteration algorithm intermingles the computation of P with the vector-matrix multiplication of the power iteration equation. In summary, the forward method (as well as the backward method) implements a *sparse power iteration without computing the iteration matrix explicitly.*

Thus, our numerical framework combines the relatively simple derivation of the model with the advantage of coping with huge state spaces. Each iteration step involves $|\mathcal{X}| \cdot |\mathcal{Y}|$ ($|\mathcal{X}| \cdot \sum_{i=0}^{r} |\mathcal{Y}_i|$, respectively) multiplications, for each of which the state transition function must be calculated. The backward method spares the expense of computing the transition function at the price of added complexity for deriving the numerical program. Apart from that, the framework inherits its numerical characteristics like convergence behavior etc. from the power iteration method.

2.5 Coping with Periodicity

For an aperiodic DTMC the limiting distribution always exists

$$x = \lim_{n \to \infty} x_n \tag{8}$$

and it equals the stationary distribution of the system. For periodic DTMC Equation (8) does not hold. In this section we describe the add-ons for the (backward or forward) iteration to cope with periodicity. The notations in the following are taken from Feller [22].

A DTMC is said to be *periodic of period* p or *p-cyclic* if the number of single-step transitions required on leaving any state to return to that same state by any path is a multiple of some integer $p > 1$; if no such $p > 1$ exists the DTMC is called aperiodic. The state set of a p-cyclic DTMC may be partitioned into p distinctive periodic classes. These classes are ordered such that a single step transition from a state of class j is only possible to enter a state of class $(j+1) \mod p$. Therefore, a path of p steps leads always to a state of the same class. Furthermore, in the DTMC with transition matrix P^p each periodic class forms a closed set. From the last two statements follows that the DTMC with transition matrix P^p is aperiodic. Consequently, there exist p limiting distributions (8), each corresponding to one class,

$$x^{(j)} = \lim_{n\to\infty} x_0 P^j (P^p)^n, \qquad 1 \le j \le p.$$

Since the stationary distribution of a DTMC equals the average state distribution (2), it can be computed by

$$x = \frac{1}{p} \sum_{j=1}^{p} x^{(j)}.$$

For the iteration algorithm to cope with periodicity, the period p of the model must be considered when testing for convergence of the iteration. To this end, the outcome of the n-th iteration step must be compared with the distribution of the $(n-p)$-th iteration step, e.g. $\| x_n - x_{n-p} \| < \epsilon$ Thus, one needs to store the distributions of p consecutive iteration steps. Once convergence is established, one simply has to average the p stored distributions to obtain the stationary distribution.

It remains to compute the period before starting the iteration. Stewart [21] presents an efficient algorithm that calculates the period of an irreducible DTMC. For reducible DTMC the algorithm must be enhanced. A directed graph may be associated with the DTMC, the vertices of the graph correspond to the states of the DTMC and the edges correspond to transitions among states which can be computed using the state transition function f. A depth-first-search algorithm [23] may be employed to compute the closures which stand for irreducible DTMCs. The period of the whole DTMC is the lowest common multiple of the periods of all persistent closures.

2.6 Recipe

Given a finite discrete Markov system, i.e., the system's salient features are measured in discrete units having a finite range, the following recipe summarizes our numerical framework. (If the system is continous the *embedded Markov chain* technique [19] may be employed to obtain a discrete Markov chain.)

Recipe:

1. Find the state variable, the influencing factors of the system and identify renewal points.
2. Set up the state transition function(s).
3. Compute the period.
4. Apply the forward iteration algorithm to obtain the state distribution within a convergence criterion that takes the period into consideration.

Based on the state distribution performance measures may be derived. The following section demonstrates the application of the recipe to a model of the AAL-2 protocol.

3 Application to AAL-2

In order to provide bandwidth efficient ATM transmission to traffic that is characterized by low bitrate, short and variable length packets, and delay sensitiveness, ITU-T specified the ATM Adaptation Layer Type 2 (AAL-2) [24]. The transmitting system multiplexes packets into a protocol data unit (CPS-PDU) that is passed as ATM cell payload onto the ATM layer. If one CPS-PDU has not enough space to accommodate the packet, the

packet is split and overlaps two CPS-PDUs. In order to ensure a maximum multiplexing delay a timer function may be used. Each time a new CPS-PDU is started to be filled a timer may be started. If the timer runs out the cell is scheduled for transmission even before the CPS-PDU is filled.

The AAL-2 uses the ATM layer service to transport service data units from one end system to another through an ATM network. For the numerical results we assume that the ATM layer service is Constant Bit Rate. The traffic stream must be shaped according to the Peak Cell Rate (PCR) negotiated in the traffic contract. A traffic shaper ensures that the ATM cells of the connection keep the minimum inter-cell distance $T = 1/\text{PCR}$ by delaying cells if necessary. **Figure 1** depicts the model

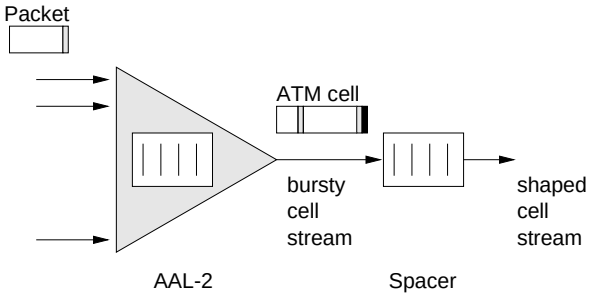

Figure 1: Model of AAL-2 Combined with a Spacer

we are going to analyze. Packets arriving at the AAL-2 are multiplexed into ATM cells; the cells are subject to spacing. If the spacer queue is exceeded, the cell is dicarded and packet loss occurs.

3.1 State Variables, Influencing Factors, and Renewal Points

Employing the model we derived in a previous paper [3], we define the system state variable as

$$X = (U, T, S).$$

Component U denotes the number of data units packed already into the CPS-PDU while T records the age of the oldest packet contained in the CPS-PDU; component S indicates the amount of time a cell will have to wait at the shaper prior to transmission. The range of X is $([0, L_u],\ [0, L_t],\ [0, L_s])$, where L_u denotes the size of the CPS-PDU, L_t is the AAL-2 timeout value, and L_s is the spacer size which corresponds to its maximum delay.

As in the above example we can identify the arrival instants as renewal points. In particular, we use the instants immediately prior to a packet arrival and immediately after such an event for conceiving the state transition function. The influencing variables are the size of the arriving packet V and the interarrival time A, respectively. Thus we have

$$Y = (V, A).$$

Thanks to two different renewal points we can use a decomposition of

$$X_{n+1} = f(X_n, Y) = f^1(f^0(X_n, V), A)$$

to set up a fast numerical program.

3.2 State Transition Functions

Since we identified two renewal points we have to set up two state transition functions: f^0 that describes the state transition from immediately before an arrival to immediately after that event, and f^1 that covers the state transition from immediately after an arrival to immediately before the next arrival.

Establishing f^0 (**Algorithm 5**) we must distinguish two cases depending on whether the arriving packet completes the PDU ($U_n^0 + V_n \geq L_u$) or not.

Input: state X_n^0 and packet length V
if $(U_n^0 + V < L_u)$ **then** {cell not completed}
$U_n^1 := U_n^0 + V$
$T_n^1 := T_n^0$
$S_n^1 := S_n^0$
else {cell completed}
$U_n^1 := U_n^0 + V - L_u$
$T_n^1 := 0$
$S_n^1 := S_n^0 + T_s$
end if
Output: state X_n^1

Algorithm 5: Transition Function f^0

The constant T_s is the spacing interval. If a cell has been filled up, the spacer state is increased by T_s. To mark a cell loss, we allow $S^1 \in \{0, \ldots, L_s + T_s\}$.

Input: state X_n^1 and interarrival time A
  if $(S_n^1 \leq L_s)$ **then** {no cell discarded}
    $S' := S_n^1$
  else {cell discarded}
    $S' := S_n^1 - T_s$
  end if
  if $(U_n^1 = 0)$ **then** {empty cell}
    $U_{n+1}^0 := 0$
    $T_{n+1}^0 := 0$
    $S_{n+1}^0 := \max(0, S' - A)$
  else if $(L_t - T_n^1 \geq A)$ **then** {no timeout}
    $U_{n+1}^0 := U_n^1$
    $T_{n+1}^0 := T_n^1 + A$
    $S_{n+1}^0 := \max(0, S' - A)$
  else {timeout}
    $U_{n+1}^0 := 0$
    $T_{n+1}^0 := 0$
    $S'' := \max(0, S' - (L_t - T_n^1))$
    if $(S'' + T_s \leq L_s)$ **then** {cell sent}
      $S_{n+1}^0 := \max(0, S'' + T_s - (A - (L_t - T_n^1)))$
    else {cell discarded}
      $S_{n+1}^0 := \max(0, S' - A)$
    end if
  end if
Output: state X_{n+1}^0

Algorithm 6: Transition Function f^1

In f^1 (**Algorithm 6**) we must take in account if a cell loss occurred in the last transition step. If we recognize $S^1 \in \{L_s + 1, \ldots, L_s + T_s\}$, S has to be decreased by T_s to obtain the real spacer occupancy. Basically, three cases are distinguished: there are no packets waiting, there are packets and no timeout occurs, and there are packets and the timer runs out. Note that a timeout possibly occurs $L_t - T_n^1$ time units after the last arrival and the next arrival occurs $A - (L_t - T_n^1)$ time units after the timeout. During this interval the spacer state is continuously decreased by one unit per time unit. If the timer runs out, the cell may be sent or discarded.

3.3 Computation of the Period

It is easy to verify that the model is p-cyclic if the packet length is a constant c and the timer time is set large enough for no timeout to occur. In that case the state is $U_{n+k} = (U_n + k \cdot c) \mod L_u$ which is clearly periodic. Thus, depending on Y a period $p > 1$ may result from this step and must be considered in the iteration. The computation of the period is done according to the sketched algorithm in 2.5.

3.4 Computation of the Stationary Distribution

The transition functions f^0 and f^1 are used to implement an iteration according to the forward iteration algorithm. Note that f^0 requires only component V while f^1 needs only component A of influencing variable $Y = (V, A)$ which leads to the desired reduction in complexity.

Having obtained the state distribution our formulae in [3] compute the waiting time and packet loss probability.

3.5 Numerical Results

The numerical results provided in this section illustrate how the above analysis may be used for obtaining source traffic descriptors for CDMA traffic carried by an AAL-2 connection.

The standard 8k vocoder employed in the North-American CDMA cellular standard IS-95 [25] operates at four different rates according to speech activity and noise conditions. Depending on the rate, the vocoder generates variable length speech frames, one frame per 20 *ms*. Including 10 octets of address and frame quality information, frames of 256, 160, 120, and 96 *bit* are observed with probabilities 0.291, 0.039, 0.072, and 0.589, respectively.

The base station multiplexes the vocoder packets of all ongoing connections onto a single AAL-2 connection to the core network. The multiplexing is organized in such a way that justifies modeling the interarrival time of packets by a geometric distribution [3]. The transport capability of the underlying ATM pipe is 2 *Mbps* which is the capacity of T1/E1 links that are widely used in today's mobile network infrastructure. The overlaid AAL-2 connection uses the CBR service category which requires declaring the source traffic descriptor PCR.

Figure 2 shows the spacing delay for the multiplexed traffic of 18, 36, 54, and 72 voice sources under the constraint of not exceeding a packet loss probability of 10^{-6}. The timeout was set to 4 *ms*.

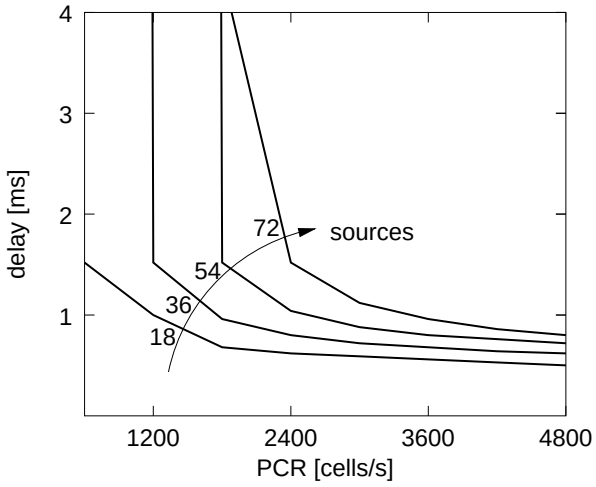

Figure 2: Packet Delay Depending on PCR

The horizontal axis shows the PCR and the vertical axis gives the expected spacing delay. The average cell rates being 409, 818, 1226, and 1636 cells/ *s* for 18, 36, 54, and 72 sources, respectively, approximately 1.5 times the average cell rate must be declared the PCR if 2 *ms* delay are allowed for spacing.

To figure out the maximum sustainable number of calls that do not exceed the loss probability of 10^{-6} and a delay of $4\,ms$ by more than 10^{-4}, we computed a scaled system with $|U| = 47$, $|T| = 97$ and $|S| = 131$. Thus, the DTMC consisted of 597229 states. The cardinality of the factor distributions were $|V| = 4$ and $|A| = 67$. The load of the system was $\rho = 0.92$. The computation of the stationary distribution with $\epsilon = 10^{-8}$ on a Sun Ultra 1 took about 1.5 hours.

4 Conclusion and Outlook

This paper presented a numerical framework for solving large finite discrete Markov models efficiently. The application of the framework was shown by a performance analysis of the AAL-2 protocol.

The numerical framework implements a computationally efficient sparse power iteration without computing the iteration matrix explicitly. To this end, the method requires the description of the model behavior by a state transition function. The application driven specification of the transition function is not too difficult even for complex models and may be eased by decomposing the transition function in a couple of functions. Function decomposition has the additional benefit of reducing the numerical complexity of the iteration.

Since storing the transition matrix is not required in the framework, it is extremely efficient with respect to storage consumption. Performing a sparse matrix multiplication it is also satisfying regarding running time. If memory is sufficient, the additional computational effort induced by the forward method can be overcome by using a sparse matrix for a composite transition (5) derived from the forward algorithm. Doing so, repetitive computations of f^i are avoided. Thus, the application driven specification of the model can be combined with a fast computation at the expenses of memory.

We implemented an analysis generator that takes the specification of the model and translates it directly into a numerical program using the forward method. The iteration algorithm may be improved by incorporating techniques that accelerate the convergence of the iteration.

To assist the analyst, good examples should be given to pass on key concepts in modeling complex systems.

References

[1] E. Rathgeb and E. Wallmeier, *ATM - Infrastruktur für die Hochleistungskommunikation*. Heidelberg: Springer, 1997.

[2] M. Dümmler and A. Schömig, "Using discrete-time analysis in the performance evaluation of manufacturing systems," in *SMOMS'99*, (San Francisco, CA, USA), 1999.

[3] N. Gerlich and M. Menth, "The performance of AAL-2 carrying CDMA voice traffic," in *Proc. 11th ITC Seminar*, (Yokohama, Japan), 1998.

[4] M. Ritter, "Discrete-time modeling of the frame-based generic cell rate algorithm," Research Report Series No. 190, Universität Würzburg, Institut für Informatik, Jan. 1998.

[5] O. Rose, "Interdeparture time correlations of the discrete-time GI/GI/1 queue," Research

Report Series No. 204, Universität Würzburg, Institut für Informatik, Apr. 1998.

[6] N. Vicari and R. Schedel, "Performance of the GFR-service with constant available bandwidth," Research Report Series No. 207, Universität Würzburg, Institut für Informatik, July 1998.

[7] N. Gerlich, P. Tran-Gia, K. Elsayed, and N. Jain, "Performance analysis of link carrying capacity in CDMA systems," in *Proc. ITC-15*, (Washington DC, USA), pp. 1159–1168, June 1997.

[8] M. Ritter, "Analysis of a queueing model with delayed feedback and its application to the ABR flow control," *Submitted to Computer Networks and ISDN Systems*.

[9] O. Rose, "Discrete-time analysis of a finite buffer with VBR MPEG video traffic input," in *Proc. ITC 15*, (Washington DC, USA), pp. 413–422, 1997.

[10] F. Hübner, "Dimensioning of a peak cell rate monitor algorithm using discrete-time analysis," in *Proc. ITC 14*, (Antibes Juan-les-Pins), pp. 1415–1424, 1994.

[11] M. Ritter and P. Tran-Gia, "Performance analysis of cell rate monitoring mechanisms in ATM systems," in *Proc. 3rd International Conference on Local and Metropolitan Communication Systems*, (Kyoto, Japan), pp. 129–149, 1994.

[12] P. Tran-Gia, "Discrete-time analysis technique and application to usage parameter control modelling in ATM systems," in *Proc. 8th Australian Teletraffic Research Seminar*, 1993.

[13] P. Tran-Gia and R. Dittmann, "A discrete-time analysis of the cyclic reservation multiple access protocol," *Performance Evaluation*, vol. 16, pp. 185–200, 1992.

[14] P. Tran-Gia, "Analysis of a load-driven overload control mechanism in discrete-time domain," in *Proc. ITC 12*, (Torino, Italy), 1988.

[15] P. Tran-Gia and H. Ahmadi, "Analysis of a discrete-time $G^X/D/1-S$ queueing system with applications in packet–switching systems," in *Proc. IEEE Infocom '88*, (New Orleans), pp. 0861–0870, 1988.

[16] P. Tran-Gia and E. Rathgeb, "Performance analysis of semidynamic scheduling strategies in discrete-time domain," in *Proc. IEEE Infocom '87*, (San Francisco, CA, USA), pp. 962–970, 1987.

[17] M. H. Ackroyd, "Computing the waiting time distribution for the G/G/1 queue by signal processing methods," *IEEE Transactions on Communications*, vol. COM-28, pp. 52–58, January 1980.

[18] M. H. Ackroyd, "Iterative computation of the M/G/1 queue length distribution via the discrete fourier transform," *IEEE Transactions on Communications*, vol. COM-28, pp. 1929–1932, November 1980.

[19] L. Kleinrock, *Queueing Systems*, vol. 1: Theory. New York: Wiley, 1975.

[20] B. Huppert, *Angewandte Lineare Algebra*. Berlin: de Gruyter, 1990.

[21] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*. Princeton: Princeton University Press, 1994.

[22] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. I. New York: Wiley, 3rd ed., 1968.

[23] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge: MIT Press, 5th ed., 1991.

[24] ITU-T, "Recommendation I.363.2. B-ISDN ATM adaptation layer type 2 specification." International Telecommunication Union, February 1997.

[25] TIA/EIA/IS-95A, "Mobile station – base station compatibility standard for dual mode wideband spread spectrum cellular systems." Telecommunications Industry Association, 1995.