

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

Michael Menth^{a,*} and Notker Gerlich^{b,**}

^aInst. of Computer Science, University of Ulm, Oberer Eselsberg, D-89069 Ulm, FRG.

^bInst. of Computer Science, University of Würzburg, Am Hubland, D-97074 Würzburg, FRG.

For performance analysis of modern communication systems discrete Markov modeling techniques have become important. This paper presents a numerical framework for solving large discrete Markov models efficiently. The first part outlines the backing 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) 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 [1–17] 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 [18] 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 for 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 involving convolution and transformation operators.

*Work done while the author was with the Inst. of Computer Science, University of Würzburg, FRG.

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

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 [1–17] in Section 2. The formalization exhibits that all the studies utilize a backward equation for the numerical program. It turns out that rearranging into a forward equation does the trick. 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.

2 A NUMERICAL FRAMEWORK FOR SOLVING DISCRETE MARKOV CHAINS

We illustrate the formalization of the DTA approaches [1–17] by the DTA of the discrete-time GI^[X]/D/1–S queuing system as presented by Tran-Gia and Ahmadi [14]. 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 inter-arrival 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 [14] 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:

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

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

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

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

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

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

and the \otimes -symbol denotes the discrete convolution

$$z(k) = z_1(k) \otimes z_2(k) = \sum_{j=-\infty}^{\infty} z_1(k-j) \cdot z_2(j).$$

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 $u(k) = \lim_{n \rightarrow \infty} u_n(k)$, 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 state distribution of the DTMC. It is this part which we are interested in formalizing it.

Strictly speaking, the required state distribution x is obtained by taking the limit of the state distributions x_n observed at discrete time instants $n = 0, 1, \dots$. Usually, taking the limit is described as $x = \lim_{n \rightarrow \infty} x_n$. Equivalently [19], the limit may be expressed by

$$x = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} x_n,$$

the limit of the average of the successive state distributions. We will come back to the second expression later in this section.

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 range by $\mathcal{X} = \{X_n \mid n \in \mathbb{N}\}$. The DTMC satisfies the memoryless property

$$\begin{aligned} \Pr(X_{n+1} = s_{n+1} \mid X_n = s_n, X_{n-1} = s_{n-1}, \dots, X_0 = s_0) \\ = \Pr(X_{n+1} = s_{n+1} \mid X_n = s_n) \end{aligned}$$

for all natural numbers n and all states s_n . In a homogenous DTMC the (single-step) transition probabilities $p_n(i, j) = \Pr(X_{n+1} = j \mid X_n = i)$ are independent of n and are consequently written as $p(i, j)$. The state transition matrix is denoted by $P = [p(i, j)]$.

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 which must be independent of the predecessors of the current state for the memoryless property to hold. We summarize the exterior factors by a system influencing variable Y that has discrete range \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 inter-arrival 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 . 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* $f : \mathcal{X} \times \mathcal{Y} \mapsto \mathcal{X}$

$$X_{n+1} = f(X_n, Y_n).$$

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

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

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

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

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

$$f = f^k \circ f^{k-1} \circ \dots \circ f^0.$$

In our example, we can — as the authors of [14] did — identify Markov points immediately before and immediately after the batch arrival instants. 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 $x_n(k)$, $n = 0, 1, \dots$, by iteration and, thus, requires an equation which is recursive in the state distribution. The numerical programs of the DTA studies [1–17] employ the equation

$$x_{n+1}(k) = \sum_{i \in \mathcal{X}, j \in \mathcal{Y}} \Pr(X_{n+1} = k | X_n = i \wedge Y_n = j) \cdot x_n(i) \cdot y_n(j),$$

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

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

A good example is the use of the discrete convolution in the GI^[X]/D/1-S analysis above:

$$z(k) = \sum_{\{(i,j) | f(i,j)=k\}} z_1(i) \cdot z_2(j),$$

where $f(i, j) = i + j$, since the convolution is the numerical program that corresponds to the sum of two independent random variables. Computing this function 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 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 [2] or [5]. 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 (*) 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 . Therefore, inverting f partitions the set $\mathcal{X} \times \mathcal{Y}$ into sets of tuples which contribute to the probability of a certain state k . This observation suggest the *forward iteration* algorithm: After initializing the successor distribution $x_{n+1}(k)$ 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.

Algorithm: Forward Iteration

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}

Implementing the forward iteration for our above example we get the GI^[X]/D/1-S Forward Iteration algorithm:

Algorithm: GI^[X]/D/1-S Forward Iteration

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}

If we employ the transition functions f^0 and f^1 for implementing the forward iteration algorithm the resulting algorithm 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 $k - 1$ transfer functions in a system having a k -dimensional influence variable Y where each transfer function depends only on a single component of Y . Then the iteration algorithm consists of k single loops instead of a single k -fold loop. In other words, the complexity of the algorithm reduces from $O(n^k)$ to $O(n)$.

Algorithm: GI^[X]/D/1-S Forward Iteration with Additional Renewal Points

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

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 [20]). 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 . The algorithm for computing the transition matrix P from the transition function reveals the relation to the power iteration.

Algorithm: Transition Matrix

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

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 $|\mathcal{X}| \cdot |\mathcal{Y}|$ 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.

2.4 Coping with Periodicity

The derivation of the forward method so far proceeded from the assumption that the model is aperiodic. Employing the transition equations only, the underlying DTMC may well be periodic with consequences for the convergence behavior of the forward iteration. In this section we describe the add-ons for the forward iteration to cope with periodicity.

An irreducible 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 $(j + 1) \bmod p$. 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 an irreducible closed set. From the last two statements

follows that the DTMC with transition matrix P^p is aperiodic. Consequently, there exist p limiting distributions, each corresponding to one class,

$$x^{(j)} = \lim_{n \rightarrow \infty} x_0 P^j (P^p)^n, \quad 1 \leq j \leq p.$$

Since the stationary distribution of an irreducible aperiodic DTMC equals

$$x = \lim_{N \rightarrow \infty} \frac{1}{N} \sum_{n=0}^{N-1} x_n$$

(see [19]) the average state distribution of the periodic chain is computed by

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

See Feller [21] for proofs.

For the forward iteration 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 j -th iteration step must be compared with the distribution of the $(j - p)$ -th iteration step. 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 eventually to obtain the stationary distribution.

It remains to compute the period before starting the iteration. Stewart [20] presents an efficient algorithm that calculates the period from the directed graph that 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. Since the state transition function f calculates the successors of a state it may be employed in a depth-first-search algorithm [22] that constructs the state graph starting from an arbitrary initial state.

2.5 Recipe

Given a finite discrete Markov system, i.e., the system's salient features are measured in discrete time units, the following recipe summarizes our numerical framework. (If the system is continuous the *embedded Markov chain* technique [18] may be employed to obtain a discrete Markov chain.)

Recipe:

1. Define the state variable of the system.
2. Identify renewal points.
3. Conceive the state transition function(s).
4. Compute the period.
5. 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 THE AAL-2 PROTOCOL

In order to provide bandwidth-efficient ATM transmission to traffic that is characterized by low bit-rate, short and variable length packets, and delay sensitiveness, ITU-T specified the ATM Adaptation Layer Type 2 (AAL-2) [23]. 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 we are going to analyze. Packets

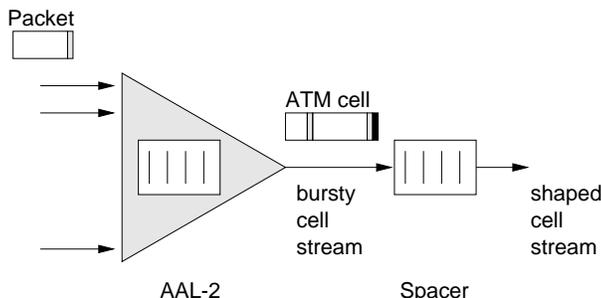


Figure 1: Model of AAL-2 combined with spacer

arriving at the AAL-2 are multiplexed into ATM cells; the cells are subject to spacing.

3.1 Define the state variable of the system

Employing the model we derived in a previous paper [2], 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 time-out value, and L_s is the maximum delay in the spacer.

3.2 Identify renewal points

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 inter-arrival time A . Thus we have

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

3.3 Conceive the 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 the 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.

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

$$f^0 : (U_n^1, T_n^1, S_n^1) = \begin{cases} (U_n^0 + V_n - L_u, 0, S_n^0 + T_s) & \text{if } U_n^0 + V_n \geq L_u \\ (U_n^0 + V_n, T_n^0, S_n^0) & \text{else.} \end{cases}$$

The constant T_s is the spacing interval.

The transition function f^1 distinguishes three cases: there are no packets waiting, there are packets and no time-out occurs, and there are packets and the timer runs out.

$$f^1 : (U_{n+1}^0, T_{n+1}^0, S_{n+1}^0) = \begin{cases} (U_n^1, T_n^1, [S_n^1 - A_{n+1}]^+) & \text{if } U_n^1 = 0 \\ (U_n^1, T_n^1 + A_{n+1}, [S_n^1 - A_{n+1}]^+) & \text{if } U_n^1 > 0 \\ & \wedge T_n^1 + A_{n+1} \leq L_t \\ (0, 0, [[S_n^1 - L_t + T_n^1]^+ + T_s - A_{n+1} + L_t - T_n^1]^+) & \text{else,} \end{cases}$$

where $[x]^+$ denotes the maximum of x and 0. Note that a time-out possibly occurs $L_t - T_n^1$ time units after the last arrival and the next arrival occurs $A_{n+1} - L_t + T_n^1$ time units after the time-out. During this interval the spacer state is continuously decreased one unit per time unit.

3.4 Compute the period

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

3.5 Apply the forward iteration algorithm

The transition functions f^0 and f^1 are used to implement a forward 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 [2] compute the waiting time and packet loss probability.

3.6 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 [24] 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, resp.

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 inter-arrival time of packets by a geometric distribution [2]. 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.

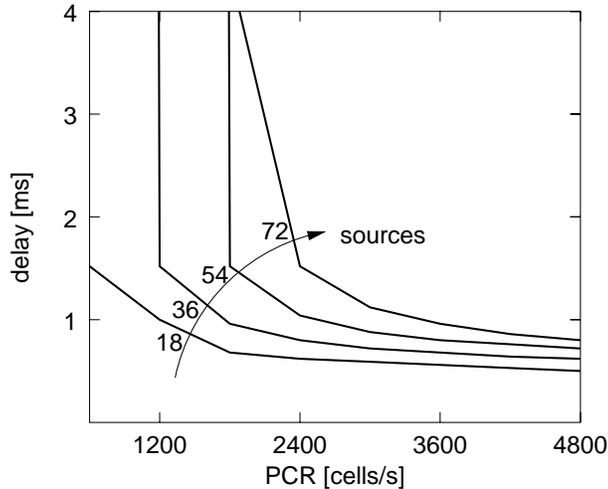


Figure 2: Source traffic descriptors for CDMA traffic

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. 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, resp., approximately 1.5 times the average cell rate must be declared the PCR if 2 ms delay are allowed for spacing.

4 CONCLUSIONS AND OUTLOOK

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

The new 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 recursive transition function. Devising 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. Decomposing the transition function has the additional benefit of reducing the numerical complexity of the iteration. The method lends itself to parallelization: the contribution of the various realizations of the influencing variables can be computed in parallel; simply summing obtains the overall solution. Since storing the transition matrix is not required the framework is efficient with respect to both the storage consumption and the running time.

The latter may be improved by incorporating techniques that accelerate the convergence of the iteration, e.g., by replacing the forward iteration by a Gauss-Seidel iteration.

REFERENCES

- [1] M. Dümmler and A. Schömig, "Using discrete-time analysis in the performance evaluation of manufacturing systems," in *SMOMS'99*, (San Francisco, CA, USA), 1999.
- [2] N. Gerlich and M. Menth, "The performance of AAL-2 carrying CDMA voice traffic," in *Proc. 11th ITC Seminar*, (Yokohama, Japan), 1998.
- [3] 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.
- [4] 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.
- [5] 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.
- [6] 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.
- [7] 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*.
- [8] 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.
- [9] 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.
- [10] 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.

- [11] 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.
- [12] 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.
- [13] P. Tran-Gia, "Analysis of a load-driven overload control mechanism in discrete-time domain," in *Proc. ITC 12*, (Torino, Italy), 1988.
- [14] 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.
- [15] 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.
- [16] 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.
- [17] 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.
- [18] L. Kleinrock, *Queueing Systems*, vol. 1: Theory. New York: Wiley, 1975.
- [19] B. Huppert, *Angewandte Lineare Algebra*. Berlin: de Gruyter, 1990.
- [20] W. J. Stewart, *Introduction to the Numerical Solution of Markov Chains*. Princeton: Princeton University Press, 1994.
- [21] W. Feller, *An Introduction to Probability Theory and Its Applications*, vol. I. New York: Wiley, 3rd ed., 1968.
- [22] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. Cambridge: MIT Press, 5th ed., 1991.
- [23] ITU-T, "Recommendation I.363.2. B-ISDN ATM adaptation layer type 2 specification." International Telecommunication Union, February 1997.
- [24] TIA/EIA/IS-95A, "Mobile station – base station compatibility standard for dual mode wideband spread spectrum cellular systems." Telecommunications Industry Association, 1995.