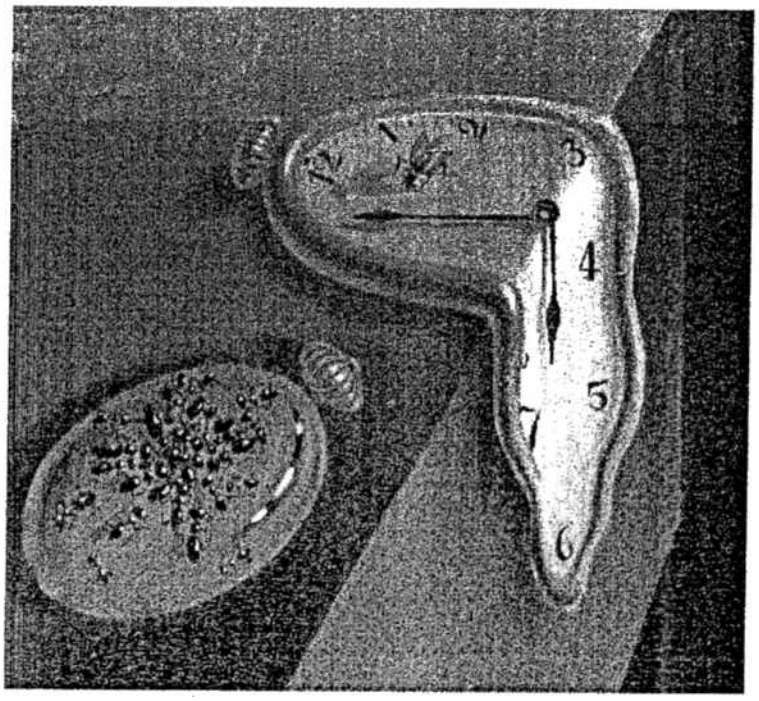
Institut für Automation
Abt. für Automatisierungssysteme

Technische
Universität
Wien

Projektbericht Nr. 183/1-44
August 1993

Determining the Size of Dedicated Shared Memory Areas for Client-Server Applications

U. Schmid



Ausschnitt aus: Salvador Dali, "Die Beständigkeit der Erinnerung"

Determining the Size of Dedicated Shared Memory Areas for Client-Server Applications

U. SCHMID[1]

August, 1993

Abstract. Many client-server applications exchange commands and parameters by means of a dedicated shared memory area. Designing an appropriate memory size is difficult in practice: If it is chosen too small, clients will frequently experience "memory exhausted" situations, causing them to block, to retry, or even to fail; making it too large wastes (usually scarce) high-speed shared memory. Our analysis provides a powerful yet easy-to-apply analytic tool supporting that design process: It allows to determine whether a given memory size T is sufficient to guarantee a certain duration of non-exhausting operation with a certain probability. Employing a combinatorial (i.e., a non-equilibrium rather than a queueing theory) approach, we established that the period of non-exhausting operation is asymptotically exponentially distributed with parameter $\lambda_T = 1/\mu_T$, where $\mu_T \sim C \cdot \kappa^{-T}$, $\kappa > 1$ for large T. The resulting uniform asymptotic formulae for the distribution function and all moments are easily applied in practice.

Keywords. dedicated shared memory size, client-server applications, FCFS scheduling, random trees, combinatorial and asymptotic probabilities.

1. MOTIVATION

We are currently developing a monitoring system *Versatile Timing Analyzer VTA*[2] designed for an (experimental) timing analysis of distributed real-time systems (see [**SSt**] for a short overview). During the development process, we encountered several instances of "low-level" client-server applications: Multiple client processors using a dedicated[3] shared memory area for posting commands and parameters which are to be executed by a single server processor.

Here are a few application examples from very different contexts:

(1) LAN coprocessors like Intel's 82596 employ a dedicated shared memory region organized as a pool of equally sized buffers (e.g., 64 bytes) for received data packets. During reception of a packet, the LAN coprocessor aquires buffers from the region and fills it with the incoming data. Note that multiple buffers needed for a large data packet may be aquired arbitrarily since modern LAN coprocessors support transparent buffer chaining via pointers (so that buffers must not be contiguous). Then, the CPU is notified that a packet has arrived (if necessary), causing it to perform some data processing. After that, the buffers are returned to the free buffer pool.

---

[1] Department of Automation, Technical University of Vienna, Treitlstraße 3, A-1040 Vienna. Email: s@auto.tuwien.ac.at

[2] Supported by the Austrian Science Foundation (FWF), grant P8390-TEC.

[3] Dedicated means that the memory region may not be used for other purposes, even if parts of it are (currently) free.

Experiencing a "memory exhausted" situation affects the overall system performance considerably since there is no other way than to discard the received packet and to signal "data lost" to the CPU. Higher level transmission protocols must provide for a necessary (and costly!) packet loss recovery.

(2) A second buffer area with a similar structure is used by LAN coprocessors to hold data packets which are to be transmitted. After aquiring and filling one or more buffers with the data, the CPU notifies the LAN coprocessor that some work has arrived (if necessary), causing it to perform packet transmission (connection setup, data transmission, retries, ...) and finally buffer reclaiming without further CPU intervention.

In case of "memory exhausted", a requesting client must be blocked until sufficient buffer space is available, causing some (unnecessary) performance degradation.

(3) Dedicated buffer areas are also often used as an interface between asynchronously operating data acquisition and data processing systems.

An example is provided by our monitoring system VTA and the real-time system under observation: Monitoring data (e.g., contents of some program variables) are extracted by instrumentation code, which has been inserted into the real-time system's software. It writes that data into a dedicated circular buffer area in shared memory, where it can eventually be accessed by the monitoring processor.

Experiencing a "memory exhausted" situation here causes a period of continuous monitoring to fail: Since it is clearly not desireable to block instrumentation code (and hence the real-time systems's program!) due to monitor insufficiencies, there is no other way than to discard some monitoring data and to signal data lost to the monitor.

(4) Finally, even some IPC mechanisms like sockets and message queues allow for dedicated buffer space; client-server processes using that type of IPC are therefore another application example.

"Buffers exhausted" situations in such applications cause client requests to block or to terminate with an error (and the need of later retry).

Choosing the "right" buffer area[4] size means balancing the tradeoff between large buffer areas (allowing for long periods of operation without encountering "buffers exhausted") and small ones (saving a lot of —often scarce— shared memory). In practice, this is largely done by *ad hoc* methods, based on feeling and experience (possibly guided by some queueing theory results). By the way, note that the required buffer area size does not depend on how the memory region is actually organized, e.g., whether a sophisticated heap management or a simple circular buffering scheme is used. However, we assume that arriving requests are serviced in the usual *first come first served* (FCFS) order.

Of course, the actual (in)appropriateness of a certain design decision becomes evident in the test phase or —at the latest— during operation. A necessary redesign, however, may prove expensive, in particular if hardware is involved. When designing our LAN coprocessor, for example, we eventually decided to provide 128 kbytes of shared memory, both for transmit and receive buffer areas. This was the result of architectural, space, and

[4] We will use the more specific term *buffer area* instead of the general term *shared memory area*.

power considerations and also influenced by the fact that most processors (e.g., M680xx) provide fast machine instructions supporting segmented adressing within 16 bit boundaries (such instructions are not useful for buffer areas larger than 64 kbytes). If, however, the provided 128 kbytes of memory had proved inadequate during the test phase, a total redesign of the LAN coprocessor (+ software) would have been necessary.

Apart from the buffer area size, there are additional design issues as well. We already mentioned the notification of the server (e.g., LAN coprocessor) in case of arriving work. This is usually only necessary if the server is idle, a condition which is sometimes difficult to check at the clients' side. Thus, the notification is either performed always or, alternatively, a server is used which performs a periodic lookup if it is idle. Of course, a long lookup period should decrease the overhead of unsuccessful lookups, but may cause a heavily loaded buffer.

In any case, it would be advantageous to have a powerful yet easy-to-apply analytic model which allows to verify a certain design decision prior to the actual implementation. We provide such a tool in this paper. Note that it does not rely on queueing theory, but on combinatorial and asymptotic methods from the analysis of algorithms.

The remainder of the paper is organized as follows: In Section 2 we introduce the underlying model and provide our major results. The subsequent Section 3 is devoted to the combinatorial investigations, Section 4 contains the asymptotic analysis. Finally, some conclusions and directions of further research are appended in Section 5.

2. Model and Major Results

Given the applications from Section 1, it is rather straightforward to model our problem as a standard (finite capacity) FCFS queueing system; the queue corresponds to the buffer area viewed from the server's perspective. With the usual assumptions of Poisson arrival and general service time distributions, this leads to an (imbedded) homogeneous Markov chain, and it seems straighforward to determine the buffer area size by means of the well-known (steady-state) distribution of the queue size (see [**KL**], [**TA1**], for example). However, it is absolutely not clear how to draw a useful design descision from there:

(1) What does it actually mean for the operation of our system, when the (steady-state) probability p_T that the buffer area size is larger than T is, say, $p_T = 0.01$?
(2) What is gained, for example, by doubling T?

Moreover, the results of [**TA1**] on the time-dependent queue size distribution are not easily applied, and relying on the steady-state distribution involves equilibrium assumptions which we do not consider as particularily meaningful for our purposes.

From the practical point of view, one would rather appreciate information on the time the system operates without experiencing a "buffers exhausted" situation; this is especially true with applications where the costs associated with such a situation are high (as in the examples of the monitoring buffer (3) and the receive area of the LAN coprocessor (1), cf. Section 1). More specifically, given a certain buffer area size T, we would like to know the probability distribution of the time our system operates without encountering a "buffers exhausted" situation when starting from an idle state.

It is not hard to see that this relates to a first passage time in the abovementioned Markov chain. The usual queueing theory device to attack such problems involves the

solution of a large dimensional system of equations — that approach is of course not easily applicable.

In our analysis, we employ a framework originally developed in our research on real-time scheduling (see [**DS1**], [**S**], [**BS1**], [**SB1**], [**BS2**], [**SB2**]), which relies on the combinatorial and asymptotic analysis of certain random trees. A similar approach has been successfully employed in the analysis of the period of stable operation of slotted ALOHA networks (cf. [**DS2**]) as well, so it is not too surprising to find it applicable here. However, it was striking how easily it adapted to the pecularities of our model.

The particular model underlying our analysis relies on *customers* which arrive probabilistically at a system consisting of a *FCFS queue* and a *single server*. Service is provided in multiples of some basic time unit which we call a *cycle*. There are both fine-granularity examples, e.g., $0.1 \cdots 1\,\mu s$ cycles modelling a single clock cycle in a machine instruction, and coarse-granularity ones, e.g., $10 \cdots 100\,ms$ cycles modelling the transmission of a single data packet over a certain network. Even larger cycle times may be found in certain synchronous systems (e.g., TDMA networks or time-triggered computation models).

The number of (new) customers arriving during a single cycle is assumed to be independent of the arrivals in the previous cycles; the appropriate probability generating function (PGF) is denoted by

$$A(z) = \sum_{k \geq 0} a_k z^k$$

and should meet the constraint $a_0 = A(0) > 0$.

The time the server needs to process a customer is assumed to be independent of the other services and of the abovementioned arrivals. The PGF of that *service time* (measured in cycles) is denoted by

$$L(z) = \sum_{k \geq 0} l_k z^k.$$

Note that the service time of a customer may be zero.

The operation of the server consists —as usual— of a sequence of alternating idle and busy periods. However, we model "forced" idle periods, that is, we assume that the server remains idle for a certain interval following a busy period (whether there are arrivals or not). The length of such an idle period has the PGF

$$I(z) = \sum_{k \geq 1} i_k z^k,$$

with the additional constraint $i_0 = 0$.

To match our applications, that model must be interpreted as follows: Each application job may consist of multiple buffers (= customers). The first arriving buffer initiates a job, and a buffer arriving subsequently is either associated to the previous initiating one or initiates a new job itself. The number of associated buffers is assumed to be geometrically distributed with parameter p. With

$$C(z) = \sum_{k \geq 1} c_k z^k, \qquad \text{and} \qquad D(z) = \sum_{k \geq 0} d_k z^k$$

denoting the PGFs of the service time of an initiating and an associated buffer, respectively, this is of course obtained by setting

$$L(z) = pD(z) + (1-p)C(z).$$

Note that $D(z) \equiv 1$ allows for associated buffers which do not contribute to the overall service time. This particular situation arises in applications like our monitoring buffer (3), cf. Section 1, where the service time of a job is (almost) independent of the number of associated buffers.

However, some care must be taken regarding the ordering of initiating and associated buffers. In fact, initiating and associated buffers need not arrive together (i.e., at once), cf. the buffer chaining technique mentioned in the LAN coprocessor examples, but must arrive at the latest during service. Consequently, the first buffer processed during a busy period must be an initiating one. Such restrictions obviously violate the usual independence assumptions, but may nevertheless be handeled by our approach.

Our forced idle periods allow for modelling several queue lookup strategies when the server is idle. $I(z) = z$ covers the (usual) case where the server resumes processing immediately (in case of some arrivals, of course), whereas $I(z) = z^d$, $d \geq 2$ models a periodic lookup after d cycles. Arbitrary random lookup intervals are covered by an arbitrary $I(z)$.

We now proceed with a short outline of our analysis, starting with a few definitions: A *busy period* of our system is defined to be an initial idle period of the server followed by a (possibly empty) busy period of the server. Such a busy period is called *T-feasible* if the maximum queue size during that busy period is at most T. A sequence of T-feasible busy periods followed by a non-feasible one is called a *T-run*, the sequence without the terminating non-feasible busy period is a *successful T-run*.

The random variable *successful T-run duration* $\mathcal{S}_T$ (all durations are of course measured in cycles), that is, the length of a sequence of busy periods not exhausting a buffer area size of T buffers, is the key issue in our derivations. $\mathcal{S}_T$ is of course only a lower bound to the whole duration of successful operation, but a very good one[5], and it admits to a rigorous mathematical analysis.

Noting that the T-feasible busy periods $\{\mathcal{B}_T^{(i)};\ i = 1, 2, \dots\}$ of a successful T-run are mutually independent since the beginning of idle periods form renewal points(!), it is easy to evaluate the PGF of $\mathcal{S}_T$: Let

$$B_T(z) = \sum_{k \geq 0} b_{k,T} z^k \quad \text{with} \quad b_{k,T} = \mathrm{P}\{\text{Duration of a } T\text{-feasible busy period is } k \text{ cycles}\}$$

denote the (improper, i.e., $B_T(1) < 1$ for finite T) PGF of the duration of a T-feasible busy period. Then, the PGF of the successful T-run duration $\mathcal{S}_T$ is given by

$$S_T(z) = \sum_{k \geq 0} s_{k,T} z^k = \frac{1 - B_T(1)}{1 - B_T(z)}. \tag{2.1}$$

[5] In fact, it is easy to show that all important quantities are asymptotically equivalent up to some lower-order terms.

This follows easily from the fact that the PGF of the duration of an arbitrary number of T-feasible busy periods is $\sum_{n\geq 0} B_T(z)^n$, and that the probability of the occurrence of the terminating non-feasible busy period equals $1 - B_T(1)$.

In [**DS2**] we have shown by singularity analysis on (2.1) that —under some (mild) conditions on $B_T(z)$— $\mathcal{S}_T$ is *always* asymptotically exponentially distributed with parameter $\lambda_T = 1/\mu_T$, where

$$\mu_T = \mathrm{E}[\mathcal{S}_T] = S'(1) = \frac{B_T'(1)}{1 - B_T(1)}.$$

Moreover, we provided uniform asymptotic expressions for the distribution function $v_{n,T} = \sum_{k=0}^{n} s_{k,T}$ and all moments $\mathrm{E}[\mathcal{S}_T^m]$ as $T \to \infty$. Since T is usually reasonable large in practice, asymptotic expressions are appropriate and, in fact, desireable. For example, in our particular LAN coprocessor design (cf. Section 1), we have a total of T=2000 of (64 byte) buffers available.

Hence, the analysis of our problem reduces to the investigation of $B_T(1)$ and $B_T'(1)$ for large T. For that purpose, we employ a framework developed in [**SB1**], which comprises two steps:

(1) We first establish a one-to-one correspondence between T-feasible busy periods and a certain family of random trees. From here, the PGF $B_T(z)$ is easily determined by combinatorial counting techniques.
(2) The required asymptotic expressions are then derived by applying certain asymptotic methods to $B_T(z)$.

We conclude this section with our major result:

THEOREM 2.1. *Let $A(z)$ with $a_0 = A(0) > 0$, $L(z) = pD(z) + (1-p)C(z)$ for $0 \leq p \leq 1$, $C(0) = 0$, and $I(z)$ with $I(0) = 0$ be the PGFs introduced in our model above, and suppose that either $\gcd\{n \geq 1 : [z^n]L(z) > 0\} = 1$ or $\gcd\{n \geq 1 : [z^n]I(z) > 0\} = 1$. If $P(z) = L(A(z))$ is such that $P'(1) < 1$ and $P''(z) \not\equiv 0$, then it is possible to find some $R > 1$ such that $z - P(z)$ has exactly two zeros $z = 1$ and $z = \kappa > 1$ within the closed disk $z \in \overline{D}(0, R)$, provided that the radius of convergence R_P of $P(z)$ is large enough (clearly, $\kappa < R < R_P$). If the radius of convergence of $I(A(z))$ is also greater than R, there exists some $\delta > 0$ such that*

(1) *the distribution function $v_{n,T} = \sum_{k=0}^{n} s_{k,T}$ of the successful T-run duration $\mathcal{S}_T$, where a buffer area size of T buffers is not exhausted, has a uniform asymptotic expansion*

$$v_{n,T} = 1 - \big(1 + O(1/\mu_T)\big)e^{-\mu_T^{-1}(1+O(1/\mu_T))n} + O\big(\mu_T^{-1}(1+\delta)^{-n}\big)$$

for $n \to \infty$ and $T \to \infty$,

(2) *the moments $\mathrm{E}[\mathcal{S}_T^m]$ of $\mathcal{S}_T$ have the uniform asymptotic expansion*

$$\mathrm{E}[\mathcal{S}_T^m] = \sum_{n\geq 1} n^m s_{n,T} = m!\big[\mu_T(1 + O(1/\mu_T))\big]^m + O\left(\mu_T^{-1}\frac{m!\sqrt{m}}{(2\pi e\delta)^m}\right)$$

for $T \to \infty$ and $m \geq 1$,

where

$$\mu_T = \frac{\Big(I'(1) - (1 - I(a_0))(L'(1) - C'(1))\Big)\kappa\big(P'(\kappa) - 1\big)}{\Big(\big(I(A(\kappa)) - I(a_0)\big)C\big(A(\kappa)\big)\kappa^{-1} - 1 + I(a_0)\Big)\big(1 - P'(1)\big)^2}\kappa^T + O\Big(\big(\kappa^2/R\big)^T\Big)$$

for $T \to \infty$; $\kappa > 1$ is easily computed by solving $P(x) - x = 0$ for $x > 1$ numerically. ∎

This result is of course easily applied in practice and answers the questions we asked at the beginning of this section. For example, given a particular buffer area size T, we know that, with a given probability $1 - q$, our system will survive a period of $n = n(T, q)$ cycles without experiencing "buffers exhausted"; clearly, $n(T, q)$ is the solution of $v_{n,T} = q$, cf. (1) in the theorem above. Neglecting all lower order terms, some easy calculations show that $n(T, q) \approx q\mu_T$. Since $\mu_T = \mathrm{E}[\mathcal{S}_T]$ grows exponentially in T, this means that doubling T, for example, yields exponentially increasing durations $n(T, q)$ of non-exhausting operation (with the same probability q).

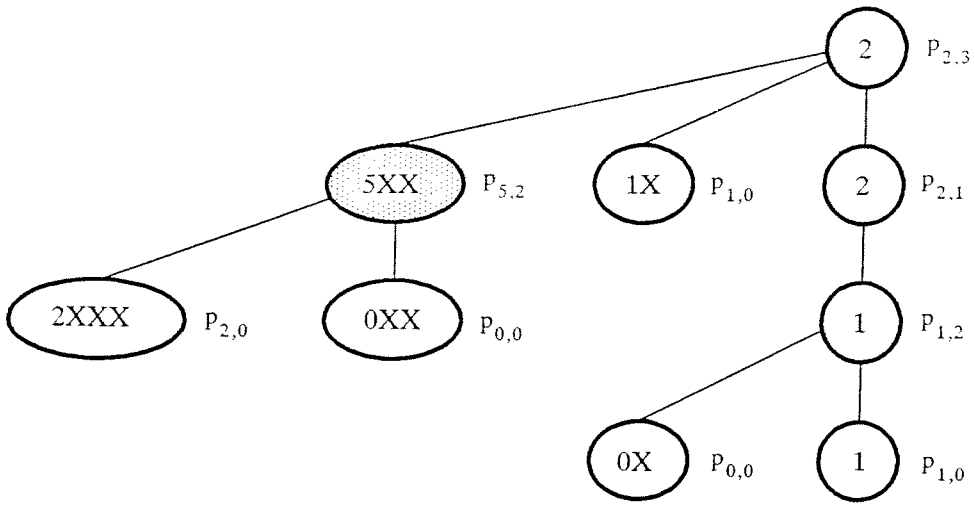
Regarding the effect of our forced idle periods, we see that only the factor of κ^T depends on $I(z)$, so that even relatively large expectations $I'(1)$ should not severly degrade the period of successful operation.

Combinatorics of a Random Tree

The combinatorial analysis contained in this section exploits a bijection between (feasible) busy periods and a certain family of random trees. This approach enables us to use powerful combinatorial methods to find the required PGF $B_T(z)$ quite easily. Note however that other combinatorial approaches are also possible, see [**TA2**] for an interesting example.

The random trees $\mathcal{B}_T$ corresponding to our T-feasible busy periods are similar to those we found in the analysis of FCFS scheduling of real-time tasks in [**SB1**]. They are constructed as follows: Each *vertex* (*node*) represents a single customer to be processed during the busy period; the root corresponds to a "virtual" customer "causing" the initial idle period of the server. A vertex v has k_v successors if exactly k_v new customers arrive during v's service. Additionally, we attach a weight p_{l_v,k_v} to each vertex v, which is the joint probability that the service time of the corresponding customer is l_v cycles and that k_v new customers arrive during that service time l_v.

The construction of the tree carries on according to a left-to-right preorder traversal. Consider the following example:



In the example tree above, we labeled each vertex v by a string of characters 'X', reflecting the queue (size) immediately prior to the beginning of v's service; the customer corresponding to v, which is obviously at the head of the queue (at the leftmost position) is represented by its service time l_v instead of the anonymous placeholder 'X'. Let us, for example, consider the (shaded) node s with label 5XX: Immediately prior to the beginning of the corresponding customer's service, the queue contains three entries which arrived during "service" of the "virtual" customer the root corresponds to. At the beginning of s's service, its customer (at the head of the queue) is removed from the queue and serviced during the next 5 cycles.

Of course, the whole duration of the busy period represented by such a tree is obtained by summing up all service times l_v of all nodes v of the tree. In addition, a short reflection of the construction process —guided by the strange node-alignment used in our example— shows that the queue size of all vertically aligned nodes is indeed the same! So we eventually arrive at the important conclusion that limiting the queue size by T corresponds to limiting the "width" of our aligned tree to T vertices.

We find it convenient to proceed with a combinatorial description method using *symbolic equations*, see [**VF**] for an introduction. Let us define $\mathcal{V}_{\ell,T}$, $\ell \geq 0$ to be the family of our random trees above which have a root with (a priori fixed) service time ℓ, and $\mathcal{V}_{*,T}$ to be the family of trees having a root with service time distributed according to our $L(z)$, cf. Section 2. Then, for $T \geq 1$, we obtain the following symbolic equations:

$$\mathcal{V}_{*,T} = l_0 \mathcal{V}_{0,T} + l_1 \mathcal{V}_{1,T} + \cdots + l_\ell \mathcal{V}_{\ell,T} + \cdots ,$$

where $l_\ell = [z^\ell]L(z)$ for $\ell \geq 0$ denotes the coefficient of z^ℓ in $L(z)$, and

$$\mathcal{V}_{\ell,T} = p_{\ell,0}\ \textcircled{\ell}\ + p_{\ell,1}\ \overset{\ell}{\underset{\mathcal{V}_{*,T}}{|}}\ + \cdots + \ p_{\ell,k}\ \overset{\ell}{\underset{\mathcal{V}_{*,T-k+1}\ \cdots\ \mathcal{V}_{*,T-1}\ \mathcal{V}_{*,T}}{\wedge}}\ + \cdots + \ p_{\ell,T}\ \overset{\ell}{\underset{\mathcal{V}_{*,1}\ \cdots\ \mathcal{V}_{*,T}}{\wedge}}$$

for all $\ell \geq 0$, where

$$p_{\ell,k} = [z^k]A(z)^\ell$$

cf. the definition of the arrival PGF $A(z)$ in Section 2.

Now, since probability weights have the same compositional properties as counting weights (the probability of the union and intersection of two disjoint and independent events equals the sum and the product, respectively, as it is the case for cardinalities of sets), the whole theory of translating admissible combinatorial constructions like the symbolic equation above to the corresponding ordinary generating functions (which are obviously PGFs in this case!) apply. Accordingly, we have to mark each node with service time ℓ by z^ℓ and to apply straightforward product and sum translations to obtain

$$V_{*,T}(z) = \sum_{\ell \geq 0} l_\ell V_{\ell,T}(z)$$

and

$$V_{\ell,T}(z) = z^\ell \sum_{k=0}^{T} p_{\ell,k} \prod_{j=T-k+1}^{T} V_{*,j}(z)$$

for $T \geq 1$, where the empty product must be interpreted as 1.

Multiplying the latter equation by l_ℓ and summing up for $\ell \geq 0$, we find

$$\begin{aligned} V_{*,T} &= \sum_{\ell \geq 0} l_\ell z^\ell \sum_{k=0}^{T} [s^k]A(s)^\ell \prod_{j=T-k+1}^{T} V_{*,j}(z) \\ &= \sum_{k=0}^{T} [s^k]L\big(zA(s)\big) \prod_{j=T-k+1}^{T} V_{*,j}(z). \end{aligned} \tag{3.1}$$

Introducing

$$\begin{aligned} W_n(z) &= \frac{1}{V_{*,n}(z)\cdots V_{*,1}(z)}, \\ W_0(z) &\equiv 1, \end{aligned} \tag{3.2}$$

and the corresponding bivariate generating function

$$W(s,z) = \sum_{k \geq 0} W_k(z)s^k,$$

we have

$$V_{*,T}(z) = \frac{W_{T-1}(z)}{W_T(z)}.$$

Multiplying the recurrence relation (3.1) by $W_T(z)$ yields the simple Cauchy-product

$$W_{T-1}(z) = \sum_{k=0}^{T} [s^k]L\big(zA(s)\big)W_{T-k}(z) = [s^T]L\big(zA(s)\big)W(s,z);$$

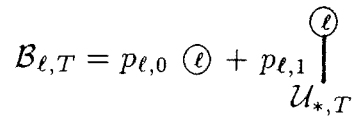
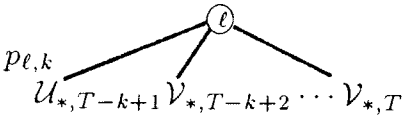
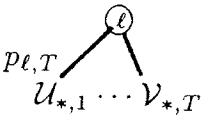
multiplying it by s^T and summing up for $T \geq 1$ shows $sW(s,z) = L\big(zA(s)\big)W(s,z) - L(za_0)$ and hence

$$W(s,z) = -\frac{L(za_0)}{s - L\big(zA(s)\big)}. \tag{3.3}$$

Now we are ready to deal with the additional features of our model, namely forced idle periods and the restriction that the first customer serviced during a busy period must not be an associated one, cf. Section 2. Noting that the first ("non-virtual") customer within a busy period is represented by the outer leftmost successor of the root, we have the following symbolic equations:

$$\mathcal{B}_T = \mathcal{B}_{*,T} = i_1\mathcal{B}_{1,T} + \cdots + i_\ell\mathcal{B}_{\ell,T} + \cdots, \tag{3.4}$$

where $i_\ell = [z^\ell]I(z)$, remember the PGF of forced idle periods from Section 2, and

$\mathcal{B}_{\ell,T} = p_{\ell,0}$  $+ \cdots +$  $+ \cdots +$ 

for all $\ell \geq 1$, where again $p_{\ell,k} = [z^k]A(z)^\ell$. The family $\mathcal{U}_{*,T}$ of trees is similar to $\mathcal{V}_{*,T}$, except that the service time of the root node is governed by the PGF $C(z)$ instead of $L(z) = pD(z) + (1-p)C(z)$. The symbolic equation reads

$$\mathcal{U}_{*,T} = \sum_{\ell \geq 0} c_\ell \mathcal{V}_{\ell,T};$$

note that we assumed $c_0 = 0$. Repeating the derivation of (3.1), the translation of the above equation yields

$$U_{*,T}(z) = \sum_{\ell \geq 0} c_\ell V_{\ell,T}(z) = \sum_{k=0}^{T} [s^k]C\big(zA(s)\big) \prod_{j=T-k+1}^{T} V_{*,j}(z).$$

Multiplying this relation by $W_T(z)$, a similar argument as before yields

$$U_{*,T}(z)W_T(z) = [s^T]C\big(zA(s)\big)W(s,z). \tag{3.5}$$

Translating the symbolic equation for $\mathcal{B}_{\ell,T}$ above, we find

$$B_{\ell,T}(z) = p_{\ell,0}z^\ell + z^\ell \sum_{k=1}^{T} p_{\ell,k}U_{*,T-k+1}(z) \prod_{j=T-k+2}^{T} V_{*,j}(z).$$

Multiplying the latter by i_ℓ and summing up for $\ell \geq 1$, cf. (3.4), we arrive at the following expression for our desired PGF $B_T(z)$:

$$\begin{aligned} B_T(z) = B_{*,T}(z) &= \sum_{\ell \geq 1} i_\ell z^\ell \Big([w^0]A(w)^\ell + \sum_{k=1}^{T} [w^k]A(w)^\ell U_{*,T-k+1}(z) \prod_{j=T-k+2}^{T} V_{*,j}(z) \Big) \\ &= I(a_0 z) + \sum_{k=1}^{T} [w^k]I\big(zA(w)\big)U_{*,T-k+1}(z) \prod_{j=T-k+2}^{T} V_{*,j}(z). \end{aligned}$$

Further multiplying this by $W_T(z)$ and remembering (3.5) shows

$$\begin{aligned}
B_T(z)W_T(z) &= I(a_0z)W_T(z) + \sum_{k=1}^{T}[w^k]I\big(zA(w)\big)U_{*,T-k+1}(z)W_{T-k+1}(z) \\
&= I(a_0z)W_T(z) + \sum_{k=1}^{T}[w^k]I\big(zA(w)\big)[u^{T-k+1}]C\big(zA(u)\big)W(u,z) \\
&= I(a_0z)W_T(z) + \sum_{k=1}^{T}[w^k]I\big(zA(w)\big)[u^{T-k}]\frac{C\big(zA(u)\big)W(u,z) - C(za_0)}{u} \\
&= I(a_0z)W_T(z) + [s^T]\Big(I\big(zA(s)\big) - I(a_0z)\Big)\frac{C\big(zA(s)\big)W(s,z) - C(za_0)}{s},
\end{aligned}$$

and we finally obtain the desired result

$$B_T(z) = I(a_0z) + \frac{[s^{T+1}]\Big(I\big(zA(s)\big) - I(a_0z)\Big)\Big(C\big(zA(s)\big)W(s,z) - C(za_0)\Big)}{[s^T]W(s,z)}. \tag{3.6}$$

It should be clear that the combinatorial approach used for our derivations is also applicable to the case where no queue size limitations are present; this easy task is left to the interested reader. We just state the (quite obvious) results

$$\begin{aligned}
V_*(z) &= V_{*,\infty}(z) = L\big(zA(V_*(z))\big) \\
U_*(z) &= U_{*,\infty}(z) = C\big(zA(V_*(z))\big) \\
B(z) &= B_\infty(z) = I(za_0) + \frac{I\big(zA(V_*(z))\big) - I(za_0)}{V_*(z)}U_*(z).
\end{aligned} \tag{3.7}$$

Note that $T(z) = zA\big(V_*(z)\big)$ is the well-known PGF of (weighted) rooted trees, solving the functional equation $T(z) = z\overline{P}\big(T(z)\big)$ with $\overline{P}(z) = A\big(L(z)\big)$.

4. Asymptotic Analysis

This section is primarily devoted to the computation of asymptotic expressions for

$$B_T(1) = I(a_0) + \frac{[s^{T+1}]\Big(I\big(A(s)\big) - I(a_0)\Big)\Big(C\big(A(s)\big)W(s) - C(a_0)\Big)}{[s^T]W(s)} \tag{4.1}$$

$$\begin{aligned}
B_T'(1) = I'(a_0)a_0 &+ \frac{[s^{T+1}]\Big(I'\big(A(s)\big)A(s) - I'(a_0)a_0\Big)\Big(C\big(A(s)\big)W(s) - C(a_0)\Big)}{[s^T]W(s)} \\
&+ \frac{[s^{T+1}]\Big(I\big(A(s)\big) - I(a_0)\Big)\Big(C'\big(A(s)\big)A(s)W(s) + C\big(A(s)\big)W'(s) - C'(a_0)a_0\Big)}{[s^T]W(s)} \\
&- \big(B_T(1) - I(a_0)\big)\frac{[s^T]W'(s)}{[s^T]W(s)}
\end{aligned}$$

as T gets large, where

$$W(s) = W(s,1) = -\frac{L(a_0)}{s - L\big(A(s)\big)}$$
$$W'(s) = W_z(s,1) = -\frac{L'(a_0)a_0}{s - L\big(A(s)\big)} - \frac{L(a_0)L'\big(A(s)\big)A(s)}{\big(s - L\big(A(s)\big)\big)^2} \tag{4.2}$$

according to (3.3). We will accomplish this task by singularity analysis, which exploits the fact that the T-th Taylor coefficient $[s^T]f(s)$ of a function $f(s)$ is mainly determined by the behavior of $f(s)$ near its dominant (i.e., smallest modulus) singularity. An overview to asymptotic methods may be found in [**VF**], [**FO**], and [**BE1**], for example.

Looking at the numerator and denominator of (4.1), we see that the dominant singularities come from $W(s)$ and $W'(s)$, caused by the vanishing denominator. The following lemma provides our simple major tool:

LEMMA 4.1. *Let $P(x)$ be a PGF with $P'(1) < 1$ and $P''(1) \not\equiv 0$. Then there is some $R > 1$ such that $P(z) - z$ has exactly two real, simple zeros $z = 1$ and $z = \kappa > 1$ within the closed disk $\overline{D}(0,R)$ with radius R around 0, provided that the radius of convergence R_P of $P(z)$ is large enough; clearly, $\kappa < R < R_P$.*

If $f(s)$ and $g(s)$ are two functions analytic in a domain properly containing $\overline{D}(0,R)$, we have

$$[s^T]\frac{f(s)}{s - P(s)} = -\frac{f(1)}{1 - P'(1)} + \frac{f(\kappa)}{\kappa\big(P'(\kappa) - 1\big)}\kappa^{-T} + O(R^{-T})$$
$$[s^T]\frac{g(s)}{\big(s - P(s)\big)^2} = \frac{g(1)(T+1)}{\big(1 - P'(1)\big)^2} - \frac{g'(1)}{\big(1 - P'(1)\big)^2} - \frac{P''(1)g(1)}{\big(1 - P'(1)\big)^3} + O(T\kappa^{-T})$$

for $T \to \infty$.

PROOF: That there are indeed two positive solutions $x = 1$ and $x = \kappa > 1$ of $P(x) - x = 0$ is straightforward when viewed geometrically; that there are no other (complex) zeros may be shown by an application of Rouche's theorem; a detailled proof is contained in [**SB1**] and omitted here for the sake of shortness.

The asymptotic expansions above are easily found by means of *subtracted singularities*: Expanding $s - P(s)$ in powers of $s - 1$ yields

$$\begin{aligned} s - P(s) &= \big(1 - P'(1)\big)(s-1) - \frac{P''(1)}{2}(s-1)^2 + O\big((s-1)^3\big) \\ &= \big(1 - P'(1)\big)(s-1)\Big(1 - \frac{P''(1)}{2\big(1 - P'(1)\big)}(s-1) + O\big((s-1)^2\big)\Big) \end{aligned} \tag{4.3}$$

for $s \to 1$. Since $f(s) = f(1) + O(s-1)$ for $s \to 1$, it follows that

$$\frac{f(s)}{s - P(s)} = \frac{f(1)}{1 - P'(1)} \cdot \frac{1}{s-1} + O(1) \qquad \text{for } s \to 1.$$

That means that the difference

$$\frac{f(s)}{s-P(s)} - \frac{f(1)}{1-P'(1)} \cdot \frac{1}{s-1}$$

has no singularity at $s = 1$ any more. However, there are still singularities of larger modulus, in particular, $s = \kappa > 1$. Note that the *subtracted singularity term* is regular at $z = \kappa$.

Expanding $s - P(s)$ in a neighborhood of κ yields

$$s - P(s) = \big(1 - P'(\kappa)\big)(s-\kappa) + O\big((s-\kappa)^2\big) \qquad \text{for } s \to \kappa,$$

and using $f(s) = f(\kappa) + O(s-\kappa)$ for $s \to \kappa$, we easily obtain

$$\frac{f(s)}{s-P(s)} = -\frac{f(\kappa)}{\kappa\big(1-P'(\kappa)\big)} \cdot \frac{1}{1-s/\kappa} + O(1) \qquad \text{for } s \to \kappa.$$

Reading off the coefficient $[s^T]$ in the geometric series constituting the subtracted singularity terms, we easily obtain the first expansion of our lemma; the remainder $O(R^{-T})$ follows from Cauchy's estimates since there are no further singularities within a disk properly containing $\overline{D}(0,R)$.

To prove the second expansion of our lemma, a simple algebraic manipulation starting from (4.3) shows

$$\frac{1}{\big(s-P(s)\big)^2} = \frac{1}{\big(1-P'(1)\big)^2} \cdot \frac{1}{(s-1)^2} + \frac{P''(1)}{\big(1-P'(1)\big)^3} \cdot \frac{1}{s-1} + O(1) \qquad \text{for } s \to 1.$$

Since $g(s) = g(1) + g'(1)(s-1) + O\big((s-1)^2\big)$ for $s \to 1$, we eventually arrive at

$$\frac{g(s)}{\big(s-P(s)\big)^2} = \frac{g(1)}{\big(1-P'(1)\big)^2} \cdot \frac{1}{(s-1)^2} + \frac{g'(1)}{\big(1-P'(1)\big)^2} \cdot \frac{1}{s-1} + \frac{P''(1)g(1)}{\big(1-P'(1)\big)^3} \cdot \frac{1}{s-1} + O(1).$$

Reading off the coefficient $[s^T]$ is simple; we just have to apply the well-known series expansion $(1-s)^{-2} = \sum_{T\geq 1} T s^{T-1}$ to obtain the major term of the second expansion of our lemma. To justify the remainder term, we only mention that the next singularity is another double pole at $s = \kappa$, yielding a remainder of order $O(T\kappa^{-T})$ (as may be seen by repeating the above analysis). This completes the proof of Lemma 4.1. ∎

Abbreviating $P(z) = L\big(A(z)\big)$ and applying Lemma 4.1 to (4.1), it is easy to determine the required asymptotics. Note that terms not involving $W(s)$ or $W'(s)$ contribute only a $O(R^{-T})$ to the coefficient $[s^T]$, which is again a straightforward consequence of Cauchy's estimates and our assumptions on the radii of convergence according to Theorem 2.1. We

obtain

$$B_T(1) = I(a_0) + \frac{\frac{\left(1-I(a_0)\right)L(a_0)}{1-P'(1)} - \frac{\left(I(A(\kappa))-I(a_0)\right)C(A(\kappa))L(a_0)}{\kappa\left(P'(\kappa)-1\right)}\kappa^{-(T+1)} + O(R^{-T})}{\frac{L(a_0)}{1-P'(1)} - \frac{L(a_0)}{\kappa\left(P'(\kappa)-1\right)}\kappa^{-T} + O(R^{-T})}$$

$$= I(a_0) + \Big(1 - I(a_0) - \frac{\big(I(A(\kappa)) - I(a_0)\big)C\big(A(\kappa)\big)\big(1-P'(1)\big)}{\kappa\big(P'(\kappa)-1\big)}\kappa^{-(T+1)} + O(R^{-T})\Big)$$

$$\cdot\Big(1 + \frac{1-P'(1)}{\kappa\big(P'(\kappa)-1\big)}\kappa^{-T} + O(R^{-T})\Big)$$

$$= 1 - \frac{1-P'(1)}{\kappa\big(P'(\kappa)-1\big)}\kappa^{-T}\Big(\big(I\big(A(\kappa)\big) - I(a_0)\big)C\big(A(\kappa)\big)\kappa^{-1} - 1 + I(a_0)\Big) + O(R^{-T})$$

and

$$B'_T(1) = I'(a_0)a_0 + \frac{[s^{T+1}]\frac{-\left(I'(A(s))A(s)-I'(a_0)a_0\right)C(A(s))L(a_0)}{s-P(s)} + O(R^{-T})}{[s^T]\frac{-L(a_0)}{s-P(s)}}$$

$$+ \frac{[s^{T+1}]\Big(I\big(A(s)\big) - I(a_0)\Big)\frac{-C'(A(s))A(s)L(a_0)-C(A(s))L'(a_0)a_0}{s-P(s)}}{[s^T]\frac{-L(a_0)}{s-P(s)}}$$

$$+ \frac{[s^{T+1}]\Big(I\big(A(s)\big) - I(a_0)\Big)\frac{-C(A(s))L(a_0)L'(A(s))A(s)}{\big(s-P(s)\big)^2} + O(R^{-T})}{[s^T]\frac{-L(a_0)}{s-P(s)}}$$

$$- \Big(1 - I(a_0) + O(\kappa^{-T})\Big)\frac{[s^T]\frac{-L'(a_0)a_0}{s-P(s)} - \frac{L(a_0)L'(A(s))A(s)}{\big(s-P(s)\big)^2}}{[s^T]\frac{-L(a_0)}{s-P(s)}}$$

$$= I'(a_0)a_0 + \frac{1}{\frac{L(a_0)}{1-P'(1)} + O(\kappa^{-T})}\Bigg(\frac{\big(I'(1) - I'(a_0)a_0\big)L(a_0)}{1-P'(1)}$$
$$+ \frac{\big(1-I(a_0)\big)C'(1)L(a_0)}{1-P'(1)} + \frac{\big(1-I(a_0)\big)L'(a_0)a_0}{1-P'(1)}$$
$$- \frac{\big(1-I(a_0)\big)L(a_0)L'(1)(T+2)}{\big(1-P'(1)\big)^2} + \frac{P''(1)\big(1-I(a_0)\big)L(a_0)L'(1)}{\big(1-P'(1)\big)^3}$$
$$+ \frac{I'(1)A'(1)L(a_0)L'(1) + \big(1-I(a_0)\big)C'(1)A'(1)L(a_0)L'(1)}{\big(1-P'(1)\big)^2}$$
$$+ \frac{\big(1-I(a_0)\big)L(a_0)L''(1)A'(1) + \big(1-I(a_0)\big)L(a_0)L'(1)A'(1)}{\big(1-P'(1)\big)^2}$$
$$- \frac{\big(1-I(a_0)\big)L'(a_0)a_0}{1-P'(1)} + \frac{\big(1-I(a_0)\big)L(a_0)L'(1)(T+1)}{\big(1-P'(1)\big)^2}$$
$$- \frac{P''(1)\big(1-I(a_0)\big)L(a_0)L'(1)}{\big(1-P'(1)\big)^3}$$
$$- \big(1-I(a_0)\big)\frac{L(a_0)L''(1)A'(1) + L(a_0)L'(1)A'(1)}{\big(1-P'(1)\big)^2} + O\big(T\kappa^{-T}\big)\Bigg)$$

$$= I'(1) + \frac{I'(1)P'(1)}{1-P'(1)} + \big(1-I(a_0)\big)\Big(C'(1) - \frac{L'(1)}{1-P'(1)} + \frac{C'(1)P'(1)}{1-P'(1)}\Big) + O\big(T\kappa^{-T}\big)$$
$$= \frac{I'(1) - \big(1-I(a_0)\big)\big(L'(1) - C'(1)\big)}{1-P'(1)} + O\big(T\kappa^{-T}\big).$$

From here it is easy to find

$$\mu_T = \frac{B_T'(1)}{1-B_T(1)}$$
$$= \frac{\Big(I'(1) - \big(1-I(a_0)\big)\big(L'(1) - C'(1)\big)\Big)\kappa\big(P'(\kappa)-1\big)}{\Big(\big(I\big(A(\kappa)\big) - I(a_0)\big)C\big(A(\kappa)\big)\kappa^{-1} - 1 + I(a_0)\Big)\big(1-P'(1)\big)^2}\kappa^T \cdot \Big(1 + O\big((\frac{R}{\kappa})^{-T}\big)\Big)$$

for $T \to \infty$, and what remains to do is to justify that the conditions of [**DS1**] are fulfilled. That conditions are

(1) $\lim_{T\to\infty} b_{n,T} = b_n$ for all $n \geq 0$.
(2) $b_{n,T} \leq b_n$ for all n and T.
(3) $B_T(1) < B(1) = 1$ for all finite T.
(4) The radius of convergence R of $B(z)$ must be greater than 1; (3) only implies $R \geq 1$.

(5) $B'(1) > 0$, which is equivalent to $B(z) \not\equiv 1$, that is, $b_0 < 1$. This condition has a probabilistic meaning: the expectation of $\mathcal{B}$ should be greater than zero.

(6) $d_B = \gcd\{n \geq 1 : b_n > 0\} = 1$, which means that the length of a busy period (i.e., all lengths) should not be a multiple of $d_B > 1$.

Conditions (1)–(3) and (5) are trivially fulfilled. To justify condition (4), we remember that

$$B(z) = I(za_0) + \frac{I\big(zA(V_*(z))\big) - I(za_0)}{V_*(z)} U_*(z), \tag{4.4}$$

cf. (3.7). $V_*(z) = L\big(T(z)\big)$ and $U_*(z) = C\big(T(z)\big)$ involve the PGF $T(z)$ of (weighted) rooted trees, that solves the functional equation $T(z) = z\overline{P}\big(T(z)\big)$ where $\overline{P}(z) = A\big(L(z)\big)$. It it is well-known (cf. [**MM**], [**S**], [**SB2**], for example) that $T(z)$ has radius of convergence $\rho > 1$ if $\overline{P}'(1) < 1$. This implies that both $V_*(z)$ and $U_*(z)$ have radius of convergence greater than 1 since all functions involved are PGF's, and the same holds for $I\big(zA(V_*(z))\big)$ since Theorem 2.1 assumes that the radius of convergence of $I\big(A(z)\big)$ is also greater than 1. Note that the denominator in the fraction of (4.4) cannot cause any problems since $\lim_{u\to 0}\big(I\big(zA(u)\big) - I(za_0)\big)u^{-1} = I'\big(za_0\big)a_1 < \infty$.

To justify condition (6), we remember that for any positive integer i, j with $\gcd\{i,j\} = 1$ there are integers $a \geq 0, b \leq 0$ such that $ai + bj = 1$. Hence, for any integer m, $m + ai = m + (-b)j + 1$, which shows that $\gcd\{m + ai, m + (-b)j\} = 1$ and thus $\gcd\{m + ki, m + lj : k \geq 0, l \geq 0\} = 1$ too. Now, any busy period consists of an initial idle period, $k_C \geq 0$ services governed by $C(z)$, and $k_L \geq 0$ services according to $L(z)$. It is clear that there are two "classes" I and J of busy periods (among others, of course) which start with some (fixed) idle period and a single C-governed service (with total duration m) and proceed with an arbitrary number $k \geq 0$ of (fixed) L-governed services with duration $k \cdot i$ and $k \cdot j$, respectively. Thus, $b_{m+k\cdot i}$ and $b_{m+k\cdot j}$ are both non-zero. Now, if $d_L = \gcd\{n \geq 1 : [z^n]L(z) > 0\} = 1$ according to Theorem 2.1, some i, j may be chosen with $\gcd\{i,j\} = 1$ and $d_B = 1$ follows.

To show that $d_I = \gcd\{n \geq 1 : [z^n]I(z) > 0\} = 1$ also implies $d_B = 1$, we have to distinguish two cases: If $I(z) = z$, there are obviously busy periods of duration 1, i.e., $b_1 > 0$ and the statement follows trivially; otherwise, there are at least two busy periods consisting of idle periods of length i and j with $\gcd\{i,j\} = 1$ only, and we are done too.

Thus, all conditions required for the application of Theorems 3.5 and 3.7 of [**DS1**] are established and our Theorem 2.1 follows. ∎

We finally mention that the conditions $d_L = 1$ or $d_I = 1$ are sufficient but not necessary to guarantee $d_B = 1$. However, it would be easy to provide a slightly extended version of Theorem 2.1 which covers the case $d_B > 1$ as well, cf. [**DS1**].

5. Conclusions

The major result of this paper are some easy-to-apply formulae which allow to quantify the effect of limited shared memory to the operation of a client-server system which exchanges commands via shared memory. We modelled the problem as a FCFS queueing system with limited queue size; the usual queueing theory results, however, have not been suitable to answer our questions.

Using combinatorial and asymptotic methods well-known from the analysis of algorithms and data structures, we studied the time S_T the system operates without exhausting a memory of size T. This random variable was found to be asymptotically exponentially distributed with a mean μ_T growing exponentially in T.

In our model, we considered (two) different classes of customers and dealt with (very modest) extensions of the usual independence assumptions. More specifically, we assumed that only a customer of a particular class ("initiators") may initiate a busy period. This assumption is important for modelling jobs which comprise several customers (i.e., groups) which are serviced in batches (but without requiring customers of a group to arrive simultaneously).

In a forthcoming paper, we will consider even more delicate independence "violations", which arise if (nearly) simultaneous arrivals of customers of a group take place. In that case, the first customer serviced after a period with no arrivals must be an "initiator", and not an arbitrary one as in our present model. Our approach seems to be very suitable to handle even such situations.

References

BE1. E. A. Bender, *Asymptotic methods in enumeration*, SIAM Review **16** (1974), 485–515.

BS1. J. Blieberger, U. Schmid, *Preemptive LCFS Scheduling in Hard Real-Time Applications*, Performance Evaluation **15** (1992), 203–215.

BS2. J. Blieberger, U. Schmid, *FCFS Scheduling in a Hard Real-Time Environment under Rush-Hour Conditions*, BIT **32** (1991), 370–383.

DS1. M. Drmota, U. Schmid, *Exponential Limiting Distributions in Queueing Systems with Deadlines*, SIAM J. Appl. Math. **53(1)** (1993), 301–318.

DS2. M. Drmota, U. Schmid, *The Analysis of the Expected Successful Operation Time of Slotted ALOHA*, (to appear in IEEE J. Inf. Th.).

KL. L. Kleinrock, "Queueing Systems," Vol. 1 and Vol. 2, John Wiley, New York, 1975.

MM. A. Meir, J. W. Moon, *On an Asymptotic Method in Enumeration*, Journal of Combinatorial Theory, Series A **51** (1989), 77-89.

S. U. Schmid, *Static Priority Scheduling of Aperiodic Real-Time Tasks*, (prepared for submission).

SB1. U. Schmid, J. Blieberger, *Some investigations on FCFS Scheduling in Hard Real-Time Applications*, J. Comput. Syst. Sci. **45** (1992), 493–512.

SB2. U. Schmid, J. Blieberger, *On nonpreemptive LCFS Scheduling with deadlines*, (submitted).

SSt. U. Schmid, St. Stöckler, *A Versatile Monitoring System for Distributed Real-Time Systems*, Proc. Safecomp '92, Zürich, Switzerland (1992).

TA1. L. Takács, "Introduction to the Theory of Queues," Oxford University Press, New York, 1962.

TA2. L. Takács, "Combinatorial Methods in the Theory of Stochastic Processes," Robert E. Krieger Publishing Company, Huntington, New York, 1977.

VF. J. S. Vitter, Ph. Flajolet, *Average Case Analysis of Algorithms and Data Structures*, Handbook of Theoretical Computer Science (J. van Leeuwen, ed.) (1990), North Holland.