Availability Analysis of Transaction Processing Systems based on User-Perceived Performance

Varsha Mainkar
AT&T Labs,
Holmdel, NJ 07733, USA.
mainkar@att.com

Abstract

Transaction processing systems are judged by users to be correctly functioning not only if their transactions are executed correctly, but also if most of them are completed within an acceptable time limit. Therefore, in this paper, we propose a definition of availability for systems for whom there is a notion of system failure due to frequent violation of response time constraints. We define the system to be available at a certain time if at that time the fraction of transactions meeting a deadline is above a certain user requirement. *This definition leads to very different estimates of availability measures such as system downtimes as compared with more traditional measures. We conclude that for transaction processing systems, where the user's perception is important, our definition more correctly quantifies the availability of the system.*

1 Introduction

Transaction processing systems of today, such as ticket booking systems, product ordering systems, or trouble ticketing systems are *real-time systems*, i.e., correct functioning of the system is not only determined by whether the transaction was executed correctly, but whether it was completed within a certain amount of time. That is, *the transaction response time* is an important performance metric. This metric is also the most important user-perceived metric, and often the one by which the user judges the quality of the transaction processing system. For instance, if a simple query on a typical library database takes 5 minutes to complete, the user may consider the query transaction failed. The maximum acceptable delay, in this case, could be 30 seconds. Thus the transaction has a 30 second *deadline* to complete, or else it is considered failed. However, unlike *hard* real-time systems [10], the user would not consider this one deadline violation as leading to *system failure*. However, if such a deadline violation happens very frequently within a certain time interval, the user will judge the library database system to be non-operational. For example, if the deadline is missed by, say, 15 out of the last 20 transactions in a 30 minute interval, the user will consider the system failed. In this paper, we propose that the correct measure of availability of a transaction processing system is this *user-perceived availability* which is very strongly based on the *performance*, especially the *response time* performance of the system.

In practice, a common "performance requirement" for transaction processing systems, is often in terms of the transaction response time, e.g., "90 % of all transactions must complete in 20 seconds", which quantifies the performance from the users's perspective. A separate "availability requirement" is also usually specified for any transaction processing system, which could be in terms of maximum allowed downtime per month. However, such an availability requirement is incomplete until a definition is provided of when a system is considered available. When the transaction processing system is a multiple server system, this definition of availability is usually made in terms of the minimum number of servers that should be operational. Such a definition is *implicitly* based on a notion of performance; i.e., it implies that having a certain minimum number of servers operational, guarantees a certain minimum level of performance. An approach to unifying performance and availability metrics in this manner has been presented by Levy and Wirth [4]. In their approach, they use two descriptors of the state of the system : the "congestion state" (number of jobs in a system, etc.) and the "availability state" (e.g. number of operational servers). The availability of the system is defined as the probability of being in any one of the availability states of the system where a minimum performance requirement is met, where this performance requirement could be a user-perceived one such as the response time.

Although our basic argument is the same as the one by Levy and Wirth, we take their approach a step further by eliminating the intermediate definition of availability in terms of the minumum number of servers and link the availability definition even more explicitly to the user-perceived performance. Further, in the context of transaction processing systems, we focus on one particular metric, i.e., the probability of a transaction missing a deadline. Thus we define availability as the probability that the system is functioning at a minimum performance level as perceived by the user. Specifically, we define availability as the *probability that at any time a required minimum fraction of transactions are finishing within a given deadline*. As we shall see in the analysis presented in this paper, such a definition accounts for failure caused not only due to failures of the servers, but also the temporary degradation of performance due to transient overloading of the system, which may not last long, but may still result in the user perceiving the system as temporarily unavailable. This results in an availability number that is substantially different from the availability as calculated by traditional approaches.

Performance-based reliability measures have been defined before for hard real-time systems, by Shin and Krishna [8]. However, in that definition, even if a *single* task missed its "hard" deadline, this event leads to a system failure. Muppala, Woolet and Trivedi [7] discuss modeling of both hard and "soft" real-time systems in their paper. However, the soft real-time systems considered in their paper fail only when all the servers fail. Thus their is no notion of performance-based failure for the soft-real-time systems. In this paper, we have considered systems, where one transaction missing a deadline does not lead to system failure, nevertheless, there is a "softer" notion of system failure due to frequent violation of response time constraints. We believe that this class of systems represent a large number of transaction processing systems that are currently operational.

Another characteristic of transaction processing systems, which affects the performance-based availability, is the fact that the rate at which transactions arrive to this system can vary significantly. In other words, transaction arrival rates have typical peak periods, during which the arrival rate could be even twice as much as the average and even five to ten times the rate at off-peak periods. System failure is then determined not only by the number of servers that failed, but the load on the system at the time they failed. Such a periodic behavior of transaction load has been modeled by Shin, Krishna and Lee in [9], where they present a very general methodology to formally derive resource control strategies for distributed systems. Our focus in this paper is different : we study the effect of peak and non-peak hours on availability when the definition of availability explicitly incorporates the performance of the transaction processing system.

In the rest of the paper, we first propose our formal definition of availability of transaction processing systems (Section 2), then describe our model for computing availability (Sections 3 and 4), and then provide an illustrative example (Section 5). In Section 6, we describe an efficient approximate method for computing availability, and also provide some more examples. We summarize and conclude the paper in Section 7.

2 Definition of System Availability

Let us describe a system to be *available* at time t, if the expected fraction of transactions arriving in the small interval $(t, t + \delta t)$ that will miss their deadlines is *less than* a given constant ϕ. For an infinitesimally small δt, this also means that the system is available at time t, if the probability that a transaction, arriving at this time, misses its deadline is less than ϕ.

Thus the performance/availability requirement on the system is given in terms of the *response time distribution of a transaction*. If $D(t)$ is the random variable denoting the response time at time t (i.e. the response time seen by a transaction that arrives at time t), the system is considered available at time t if

$$Pr[D(t) > d] \leq \phi. \tag{1}$$

Here d is a given "soft" deadline, and ϕ is given as a requirement. The problem is to compute the availability at time t, given that the system is defined to be available at time t by Equation (1). Define $F_d(t)$ as

$$F_d(t) = Pr[D(t) \leq d]. \tag{2}$$

Then availability, $A(t)$, at time t, is given by

$$A(t) = Pr[F_d(t) > 1 - \phi]. \tag{3}$$

With such a definition of availability, we capture the inherent dependence between performance and availability in one equation. We term this measure as the *user-perceived availability*. The key observation about such a definition of availability is that it takes into account the possibility of transient system failure (i.e. unresponsiveness) triggered purely by transient overloads caused due to the random nature of transaction arrivals. Thus, it will take into account the possibility that even though there are no server failures in the system, there could be a temporary system failure due to a temporary overload.

The computation of availability at time t will proceed as follows : let the state of the system at time t be denoted by $S(t)$. (This state description will include the operational state of the servers, and the state of the transactions in the system.) Let $P_i(t)$ denote the probability that $S(t) = S_i$. Now, let $F_{d|i}$ be the probability that the response time of a transaction is less than d, given that the system is in state S_i at the time of arrival of the transaction. Then define

$$\begin{aligned} r_i &= 1, \quad \text{if } F_{d|i} > 1-\phi \\ &= 0, \quad \text{if } F_{d|i} \leq 1-\phi. \end{aligned} \tag{4}$$

Then, if Ω is the set of all states that the system can be in, the user-perceived availability is given by :

$$A(t) = \sum_{S_i \in \Omega} r_i P_i(t). \tag{5}$$

which represents summing over all probabilities of being in the states in which the probability of missing a deadline is less than ϕ.

To compute $A(t)$ as given by Equation (5), we first need to compute $P_i(t)$ and then r_i. To compute r_i, we need the response time distribution of a transaction, given that the system is in a certain state at the time of arrival of the transaction. Also, for steady-state availability, we need the limiting values of these quantities. The next two sections explain how we do these computations.

3 The Model for State Probabilities

Consider a distributed system of N identical servers. Let the transaction processing rate of each server be μ. The transaction arrival rate at time t is $\lambda(t)$. Each server fails at the rate γ and can be repaired at the rate η. Let us assume that there is only one repair person available for the servers. The incoming transactions join a common queue, and are served in a first-in-first-out manner, each server processing one transaction at a time. We assume that the arrival rate follows a periodic pattern with a period of 24 hours. In every 24 hour period we assume we have three different periods of varying loads. This is based on the typically observed transaction arrival behavior which has a "peak", period, a "medium" period and a "low" load period. More formally, we define $\lambda(t)$ as :

$$\begin{aligned} \lambda(t) &= \lambda_1, \ 0 \leq t \leq t_0 \\ &= \lambda_2, \ t_0 < t \leq t_1 \\ &= \lambda_3, \ t_1 < t \leq 24. \end{aligned} \tag{6}$$

where t_0 and t_1 are constants.

Furthermore, we assume that in each period, transactions arrive according to a Poisson process. We also assume that the failure, repair and transaction processing times are exponentially distributed. We assume that there is a limited buffer space, B, for queueing transactions in the system. Under these assumptions, the model for the duration of time for which the arrival rate is constant is a homogeneous continuous-time Markov chain with a finite state-space.

The system state can be described by a 4-tuple (l, m, p, q), where l is the number of servers that are operational but idle, m is the number of servers that are failed, p is the number of servers busy processing transactions and q is the number of transactions waiting in the queue. A Markov model with this state description can be generated; however, for large buffer size and number of servers, this model can become quite large. A higher-level specification language such as a *Stochastic Reward Net* (SRN) [1], along with a tool such as SPNP [2], can be used to generate the infinitesimal generator matrix corresponding to this Markov chain.

We generate three different Markov chains, corresponding to the three different arrival rates λ_1, λ_2 and λ_3. We term these as the "Phase 1", "Phase 2" and "Phase 3" Markov models respectively and denote the infinitesimal generator matrices corresponding to these three Markov models by $\mathbf{Q}_1$, $\mathbf{Q}_2$, and $\mathbf{Q}_3$.

These Markov models can be solved in phases to compute the overall probability of the distributed system being in a state S_i at time t. Now, it is clear that for $t \leq t_0$ the probability vector at time t is simply given by :

$$\mathbf{P}(t) = \mathbf{P}(0)e^{\mathbf{Q}_1 t},$$

where $\mathbf{P}(0)$ is the initial state probability vector and $\mathbf{P}(t)$ is the state probability vector at time t. For $t_0 < t \leq t_1$, the state probability vector at time t is given by :

$$\mathbf{P}(t) = \mathbf{P}(t_0)e^{\mathbf{Q}_2(t-t_0)} = \mathbf{P}(0)e^{\mathbf{Q}_1 t_0}e^{\mathbf{Q}_2(t-t_0)}.$$

For $t_1 < t \leq 24$, we have :

$$\mathbf{P}(t) = \mathbf{P}(t_1)e^{\mathbf{Q}_3(t-t_1)} = \mathbf{P}(0)e^{\mathbf{Q}_1 t_0}e^{\mathbf{Q}_2(t_1-t_0)}e^{\mathbf{Q}_3(t-t_1)}.$$

Generalizing from the above equations, we have for $t = 24n + s$, $n \geq 0$, $0 \leq s < 24$, n is a non-negative integer :

$$\begin{aligned} \mathbf{P}(t) &= \mathbf{P}(0)\mathbf{E}^n e^{\mathbf{Q}_1 s}, \quad 0 \leq s \leq t_0 \\ &= \mathbf{P}(0)\mathbf{E}^n e^{\mathbf{Q}_1 t_0}e^{\mathbf{Q}_2(s-t_0)} \quad t_0 < s \leq t_1 \\ &= \mathbf{P}(0)\mathbf{E}^n e^{\mathbf{Q}_1 t_0}e^{\mathbf{Q}_2(t_1-t_0)}e^{\mathbf{Q}_3(s-t_1)}, \ t_1 < s \leq 24. \end{aligned} \tag{7}$$

where $\mathbf{E} = [e^{\mathbf{Q}_1 t_0 + \mathbf{Q}_2(t_1-t_0) + \mathbf{Q}_3(24-t_1)}]$. These probabilities can be computed using a tool such as SPNP [2], which uses sophisticated numerical techniques based on uniformization for efficient computation of transient probabilities.

Note that the matrix $\mathbf{E}$ is the stochastic transition probability matrix corresponding to the discrete-time Markov chain (DTMC) embedded at the points 0,24,48,.... This DTMC is finite, aperiodic and irreducible and therefore the matrix $\mathbf{E}^n$ has a limit as n tends to infinity. Let

$$\mathcal{E} = \lim_{n \to \infty} E^n.$$

The rows of matrix $\mathcal{E}$ are identical and correspond to the steady state probability vector $\mathbf{\Pi}$ of the embedded DTMC. Therefore $\mathbf{P}(0)\mathcal{E}$ is also equal to $\mathbf{\Pi}$.

Then as $t \to \infty$, the continuous time probability is given by :

$$\begin{aligned}
\lim_{t\to\infty} \mathbf{P}(t) &= \lim_{n\to\infty} \mathbf{P}(24n+s) \\
&= \mathbf{\Pi} e^{\mathbf{Q}_1 s}, \quad 0 \le s \le t_0 \qquad (8) \\
&= \mathbf{\Pi} e^{\mathbf{Q}_1 t_0} e^{\mathbf{Q}_2 (s-t_0)} \qquad t_0 < s \le t_1 \qquad (9) \\
&= \mathbf{\Pi} e^{Q_1 t_0} e^{\mathbf{Q}_2 (t_1-t_0)} e^{\mathbf{Q}_3 (s-t_1)}, \quad t_1 < s \le 24. \qquad (10)
\end{aligned}$$

Thus, it is clear that the steady-state probability is periodic with a period of 24 hours. Let us denote the steady-state probability vector at a large t, $t = 24n + s$, by $\pi(s), 0 \le s \le 24$. Equations (8)-(10) provide us with the steady-state values of the state probabilities, $\pi(s)$, which can then be used to determine the steady-state availability of the system.

4 Model for Response Time Distribution

The previous section showed us how to calculate the $P_i(t)$'s and their steady-state values. In this section, we show how we calculate the r_i's. For this purpose, we must calculate the response time distribution of the transaction, given that the system state at the time of its arrival is i. We use the *tagged job approach* [5, 6] in computing the response time distribution. In the tagged job approach we use the *PASTA* [11] property of Poisson arrivals, i.e. the property that "Poisson arrivals see time averages". Thus, the probability that a transaction arriving at time t sees the system in a certain state i is given by the state probability at time t, which is $P_i(t)$. This $P_i(t)$ can be computed using the model described in the previous section. For the response time distribution, we build another Markov model (termed the "tagged job model") where apart from the regular state description, we additionally keep track of a "tagged job" (or tagged transaction). Thus the system state will now be described by $(l, m, p, q, t_w, t_s, t_c, t_f)$, where t_w is 1 if the tagged job is waiting in the queue, and 0 otherwise, t_s is 1 if the tagged job is being processed by a server and 0 otherwise, t_c is 1 if the tagged job is completed, and 0 otherwise, and t_f is 1 if the server fails while processing the tagged job and 0 otherwise. Suppose we want to compute the response time distribution of a transaction, given that the system was in a particular state $S_i = (L, M, P, Q)$, $S_i \in \Omega$, at the time of its arrival. We proceed by setting the initial state of the tagged job model to $(L, M, P, Q, 1, 0, 0, 0)$. This state represents the arrival of a tagged job to the system, when the state was (L, M, P, Q). The transitions between states should be such that the tagged job gets a server only when all the Q jobs that were in front of it get servers, so as to represent the FIFO queueing discipline. Note that we do not model new arrivals of transactions to this system, because transactions arriving after the tagged transaction are irrelevant to the computation of response time distribution of the tagged job. The tagged job is either processed or there is a server failure during its processing. Thus, either $t_c = 1$ or $t_f = 1$. States in which either $t_c = 1$ or $t_f = 1$ are made absorbing states. (We can generate this Markov model also using a Stochastic Reward Net specification.) Note that since no new transaction arrivals are modeled, we do not have different "phases" of Markov models in this case.

The probability that the response time is less than u, then, is the probability of the tagged job Markov chain being in a state where $t_c = 1$ at time u. This can be found by solving the tagged job Markov model for its time-dependent state probability. Denote the state probability vector for the tagged job model with initial state $S_i \in \Omega$, by $\mathbf{R^i}$, and let $\mathbf{R^i}_C(u)$ denote the probability of being in any of the absorbing states where $t_c = 1$ at time u. Thus, for the specified deadline d, we can find $\mathbf{R^i}_C(d)$. Then, r_i is 1 if $\mathbf{R^i}_C(d) \ge 1 - \phi$ and 0 otherwise. We can now find r_i's for all $i \in \Omega$, by creating a tagged job model corresponding to each $i \in \Omega$.

The availability at time t, is then given by $\sum_{i\in\Omega} r_i P_i(t)$. Since the steady-state probabilities of this system are periodic, the steady-state availability of such a system will also be periodic with a period of 24 hours. Let $A(s)$ denote the steady state availability at s hours after the beginning of a 24 hour period. Then, $A(s)$ is given by $\sum_i r_i \pi_i(s)$, where $\pi(s)$ is given by Equations (8)-(10).

5 Example

Consider a distributed system with five identical servers. Each server can process 1200 transactions per hour. Each server has a mean time to failure (MTTF) of 4000 hours and a mean time to restore (MTTR) of 4 hours. The transaction arrival rate is 600 transactions per hour from time 0 to 10 hours, 4000 transactions per hour from 10 to 16 hours and 2000 transactions per hour from 16 to 24 hours. The maximum number of transactions waiting to be processed in this system can be 50. The system is considered to be failed at any time if at that time the probability that a transaction has a response time of less than 9 seconds is less than 80 %. Given this definition, we would like to compute the steady-state availability of the system.

We solve the model as described in the previous sections; i.e. to compute the steady state probabilities, we use the first Markov model. We do this by first computing the DTMC steady state vector as follows : we set the initial state to that which has all servers operational and no transactions in the system, and compute $\mathbf{P}(10)$, $\mathbf{P}(16)$ and $\mathbf{P}(24)$ as described in Section 3, and repeat this process by setting the initial state probability of the Phase 1 Markov model to $\mathbf{P}(24)$ and then re-computing $\mathbf{P}(10)$ and so on. After iterating this process a few times until convergence is achieved we set $\mathbf{\Pi}$ to the converged value of $\mathbf{P}(24)$. Then setting the initial probability of the Phase 1 Markov model to $\mathbf{\Pi}$, the probability $\pi(s)$ may be computed for $0 \le s \le 10$, by solving the Phase 1 Markov model for time-dependent probabilities from 0 to 10 hours. Similarly, by setting the initial probability of the Phase 2 Markov model to $\pi(10)$, and solving the Phase 2 Markov model for time-dependent probabilities from 0 to 6 hours, we get $\pi(s)$ for $10 < s \le 16$. Finally, by setting the initial probability of the Phase 3 Markov model to $\pi(16)$, and solving the Phase 3 Markov model for time-dependent probabilities from 0 to 8 hours, we get $\pi(s)$ for $16 < s \le 24$.

Once we have the state probabilities, we can compute the availability also as described in the previous section, using the tagged job model. Since we are mainly interested in steady-state availability, we compute it by using the vector $\pi(s)$. This will give us the steady-state availability $A(s)$, $0 \le s \le 24$.

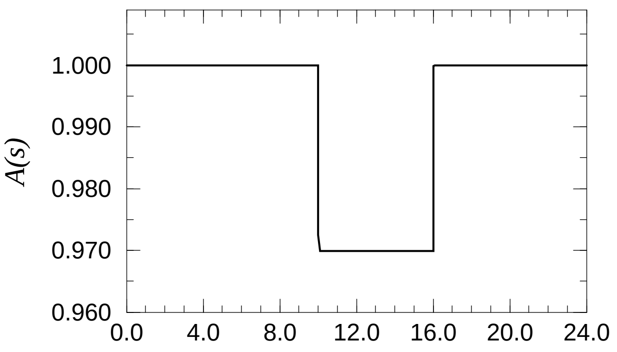

Figure 1. Steady-state Availability for a 1-day period

Figure 1 shows the instantaneous availability over a period of one day, when the system has reached steady-state, for this example.(Though the curves look perfectly vertical, as if there are discontinuities in the plot, this is the effect of granularity of the plot, and in fact there will be a gradual change in the availability at 10 and 16 hours). Note that the availability in the 6-hour peak period in the middle of the day is, as expected, significantly lower than the availability during the other hours in the day. The availability in each interval quickly reaches a steady-state in that interval : that availability is 99.9999982 % in the first interval, 96.986841 % in the second interval, and 99.994225 % in the third interval.

To calculate the downtime per month (30 days), we can integrate numerically, the curve shown in Figure 1. Thus

$$\text{Downtime per month} = 30 \times (24 - \int_0^{24} A(s)ds) \quad \text{hours}.$$

The downtime per month using this calculation in this case, is 325 minutes (5.4 hours). If the availability requirement is that downtime be less than one hour per month, this model tells us that five servers are not enough to meet this requirement.

5.1 “Implicit” Model Approach

We can compare this approach with a more simple approach for calculating availability with an implicit notion of performance, as discussed in Section 1. For using this approach to compute the steady-state availability of the system, we must first determine the minimum number of servers that is required to be operational for the system to be considered available. This is determined by using a separate performance model.

For each of the time-intervals where transaction arrival rate is constant, we can use an $M/M/c$ model (assuming that buffer space is large enough) to compute the steady-state response time distribution of the system for each c, $c = 1, \ldots, 5$, where c is the number of servers. The response time distribution of the $M/M/c$ model is given by a closed-form expression which can be found in [3].

For the arrival rates = 600, 4000 and 1200 per hour, the probability of the steady state response time being less than 9 seconds is shown in Table 1 for the number of servers, $c = 1, \ldots, 5$. It is clear from the table that for the first

	c = 1	c =2	c = 3	c = 4	c = 5
λ = 600	0.60	0.93	0.95	0.95	0.95
λ = 4000				0.78	0.93
λ = 2000		0.55	0.92	0.95	0.95

Table 1. Response Time Probabilities Using $M/M/c$ model

10 hours, at least 2 out of 5 servers are required to meet the performance requirement; for the next 6 hours, all five servers are required and for the next 8 hours, at least 3 servers are required. This evaluates to 5 $\times 10^{-10}$ % unavailability in the first 10 hours, 0.5 % unavailability in the next 6 hours, and 10^{-6} % unavailability in the next 8 hours.

This is equivalent to 54 minutes of downtime per month (30 days). If the availability requirement is for the downtime to be less than one hour every month, then this analysis tells us that five servers are enough for meeting the requirement (albeit closely). Thus, there is a substantial difference between the estimates made using this implicit model, and estimates made using explicit user-perceived availability. As described earlier, the user-perceived availability takes into account the possibility of system failure purely due to not meeting the response time requirement, which may or may not be triggered due to server failures. Therefore, the downtime estimate, in this case, was larger using this method as compared with the implicit model (this is not always true). The issue of whether performance degradation due to the random nature of arrivals and due to queueing should be "counted" while estimating downtime is a debatable one. However, we argue that irrespective of whether the cause was a server failure or simply a temporary congestion, the user will perceive the degradation of the performance. If this perceived degradation is substantial enough, it will result in the user determining the system to be "unavailable". Thus if a user-perceived definition of availability is to be accepted, we believe it must reflect this perception as accurately as possible.

6 Approximate Method for computing User-Perceived Availability

The method for computing availability using Markov models as described in Section 3 and Section 4 has several disadvantages. First, if the buffer space for waiting transactions is large (which is likely in modern systems, where memory is fairly cheap), the number of states generated will be simply too large and the model will become essentially unsolvable. Second, the tagged job approach for computing response-time distribution is a tedious one where the response time model must be solved for each initial state, for all states generated in the steady state model. Therefore a simpler way to compute the user-perceived availability is necessary.

We do this by following the expected approach of decomposing the model into a server failure and repair model and the transaction arrival and service model. We solve the transaction model for response time performance in steady-state assuming that the number of operational servers and the transaction arrival rate does not change in a short time. Such an approximation is acceptable because the time-scales of the failure-repair events and the duration of "phases" are orders of magnitude higher than the time-scales of transaction arrivals and service. Note though, that we do this while still keeping the availability definition strictly in terms of the user-perceived performance. Thus though we use a two-level model to solve for this availability, our *definition* of availability does not change to being "two level".

Let us first compute the probability at any instant in the steady-state of the transaction model that the system is available as defined by Equation (1), *given* that there are c servers operational at that time. For doing that, we basically use Equations 2 through 5 of Section 2. The first quantity that can be computed is $F_{d|i}$, which is the probability that an arriving transaction misses the deadline, given that the system is in state S_i, where $S_i = i$ is now simply the number of transactions in the system. Recall that we assume exponential service time μ for the servers. Thus, given i transactions in the system and c servers, the response time D is a $EXP(\mu)$ random variable, if $i < c$; and D is the sum of an $ERLANG(\mu c, i-c+1)$ random variable and an $EXP(\mu)$ random variable, if $i \geq c$. Let

$$F_{exp}(t) = 1 - e^{-\mu t}, \tag{11}$$

and

$$F_{erl}(t) = 1 - \sum_{j=0}^{i-c} \frac{(c\mu t)^j}{j!} e^{-c\mu t}. \tag{12}$$

Then

$$\begin{aligned} F_{d|i} &= F_{exp}(d), \qquad \forall\, i < c \\ &= (F_{erl} \otimes F_{exp})(d) \quad \forall\, i \geq c \end{aligned} \tag{13}$$

where $\otimes$ represents the convolution operator. Given the $F_{d|i}$'s, the r_i's can be found using Equation (4).

To find the probability of having i transactions in the system we assume a simple $M/M/c$ model. (Since, as mentioned before, transaction buffer sizes can be quite large, we assume an infinite buffer space.) If $a = \lambda/\mu,$, where λ is the transaction arrival rate, the steady state probabilities for an $M/M/c$ model where $a < c$ are given by :

$$\Pi_0 = \left[\sum_{k=0}^{c-1} \frac{a^k}{k!} + \frac{a^c}{c!(1-\frac{a}{c})} \right]^{-1} \tag{14}$$

$$\Pi_j = \frac{a^j}{j!}\Pi_0 \qquad 1 \leq j \leq c-1 \tag{15}$$

$$\Pi_j = \frac{a^j}{c!c^{j-1}}\Pi_0 \qquad j \geq c \tag{16}$$

Now, it is clear that there will be a certain threshold K after which all r_i's with $i \geq K$ will be 0. Then, the conditional steady-state availability is given by $\sum_{i=0}^{K} r_i \Pi_i$.

Recall that this entire calculation was made under the condition that there are c servers operational and that the transaction arrival rate is λ. Denote the availability derived in such a way by $A_{c,\lambda}$, and let Q_c denote the probability that there are exactly c out of N servers available. Let $q = \eta/(\gamma + \eta)$ denote the availability of a single server Then

$$Q_c = \frac{N!}{c!(N-c)!} q^c (1-q)^{N-c}. \tag{17}$$

Then, $A_\lambda = \sum_{c=1}^{N} A_{c,\lambda} Q_c$, is the availability of the system under a certain load λ. Assuming the transaction load pattern as described in Section 3, the downtime in minutes per day is given by :

$$[\,(1 - A_{\lambda_1}) \times t_0 + (1 - A_{\lambda_2}) \times (t_1 - t_0) + (1 - A_{\lambda_3}) \times (24 - t_1)\,] \;\times\; 60. \qquad (18)$$

The following shows the approximate method of computing user-perceived availability, in a step-by-step algorithmic manner :

1. First compute Q_c for $c = 1, \ldots, N$.
2. For each transaction arrival rate $\lambda = \lambda_1, \lambda_2$ and λ_3, do the steps 3 thru 7.
3. For each number of servers $c = c_0, \ldots, N$, where c_0 is the minimum number of servers for which $\lambda < c_0\mu$, do steps 4 thru 6.
4. For $i = 0, 1, 2, \ldots$, compute r_i, by computing $F_{d|i}$ as shown in Equation (13), until r_i is zero for some $i = K$, after which all r_i's will be zero.
5. For $i = 0, \ldots, K$, compute Π_i, the steady-state probabilities.
6. Set $A_{c,\lambda} = \sum_{i=0}^{K} r_i \Pi_i$
7. Set $A_\lambda = \sum_{c=c_0}^{N} A_{c,\lambda} Q_c$.
8. Compute downtime minutes per day as shown in Equation (18).

Using the approximate model for computing, the steady-state user-perceived availability for the example system in Section 5 is 99.9999 % for the first 10 hours, 96.9861 % for the next 6 hours, and 99.9943 % for the next 8 hours. The downtime estimate is 326.16 minutes per month. Thus, for the given failure, repair, transaction arrival, and transaction service rates, this approximation is very good.

6.1 Examples

The approximation allows us to answer what-if questions quickly, and therefore allows for efficient evaluation of design alternatives. Thus, for example, in Section 5, we determined from the user-perceived downtime measure, that five servers were not enough to meet the maximum one-hour downtime requirement. Using the approximate method, we can carry out the same calculation for $N = 6$ servers. The downtime in this case turns out to be 27.9 minutes per month, which meets the requirement. Therefore, we conclude that we need six servers to satisfy the downtime requirement.

As another example, consider the same system, but with a different load pattern. We consider a very sharp peak hour load on the system : let us assume that $\lambda_1 = 600$ transactions per hour from 0 to 10 hours, $\lambda_2 = 5500$ from 10 to 14 hours, and $\lambda_3 = 600$ again from 14 hours to 24 hours. For this system, with the same performance-availability requirements, we would like to determine the number of servers necessary to again meet a maximum one hour per month downtime requirement. Using the approximate method, we determine the downtime for $N = 5, 6$ and $N = 7$ servers. The following table shows the downtime estimates in minutes per month using the approximate model and the implicit model for these three cases.

N	User Perceived	Implicit Model
5	3436.8	7200
6	506.4	43
7	40.4	0.15

Table 2. Comparison of User-Perceived Downtime *vs* Implicit Model Downtime in minutes

Thus we need 7 servers to satisfy the downtime requirement in this case. Note that unlike other cases, when we have 5 servers, the implicit model downtime estimate is larger than the user-perceived downtime estimate. This is so because the steady-state probability of missing the deadline when the load is 5500 transactions per hour, and there are 5 servers, is 0.62 which is less than 0.8. Thus according to the implicit model, with 5 servers the system cannot be considered available, and therefore, the system is unavailable for 4 hours, which is 7200 minutes per month. The downtime in the rest of the day when there is low load is almost negligible. The user-perceived availability in the four-hour peak period is, on the other hand, 52.3 %. In this case, this measure is allowing for the random occurrence that the number of transactions in the system is low, even in the peak period, and when the user may perceive the system to be "available". Therefore, user-perceived downtime is smaller than the implicit model downtime, in this case.

7 Summary and Conclusions

In this paper, we presented a formal definition of availability for transaction processing systems for which an informal notion of system failure due to degraded system performance already existed. The definition is simple, explicit and is a natural representation of the users's performance/availability expectation of a typical transaction processing system. We presented a Markov model that can be used to compute the availability as defined in this paper. The

Markov model was solved numerically, using existing tools. We also presented an approximate method of computing this user-perceived availability.

We conclude that this measure more appropriately reflects availability as actually perceived by the user and should be the measure of choice for systems for which the user perceived availability is important.

Acknowledgements

The author would like to thank Paul Reeser for many insightful comments, and Gagan L. Choudhury for reviewing this paper.

References

[1] G. Ciardo, A. Blakemore, P. F. Chimento, and K. S. Trivedi. Automated generation and analysis of Markov reward models using stochastic reward nets. In A. Freidman and J. W. Miller, editors, *Linear Algebra, Markov Chains, and Queueing Models, IMA Volumes in Mathematics and its Applications*, volume 48, pages 145–191. Springer-Verlag, Heidelberg, 1993.

[2] G. Ciardo, J. Muppala, and K. Trivedi. SPNP: Stochastic Petri net package. In *Proc. Int. Conf. on Petri Nets and Performance Models*, pages 142–150, Kyoto, Japan, Dec. 1989.

[3] R. B. Cooper. *Introduction to Queueing Theory*. CEEPress Books, Washingtion, D.C., U.S.A., 3rd edition, 1990.

[4] Y. Levy and P. E. Wirth. A unifying approach to performance and reliability objectives. In M. Bonatti, editor, *Teletraffic Science for New Cost-Effective Systems, Networks and Services, ITC-12*, pages 1173–1179, Amsterdam, 1989. Elsevier Science Publishers, B. V. (North-Holland).

[5] B. Melamed and M. Yadin. Numerical computation of sojourn-time distributions in queueing networks. *J. ACM.*, 31(4):839–854, Oct. 1984.

[6] J. K. Muppala, K. S. Trivedi, V. Mainkar, and V. Kulkarni. Numerical computation of response time distributions using stochastic reward nets. *Annals of Operations Research: Special Issue on Queuing Networks*, 48:155–184, 1994.

[7] J. K. Muppala, S. P. Woolet, and K. S. Trivedi. Real-time performance in the presence of failures. *IEEE Computer*, pages 37–47, May 1991.

[8] K. G. Shin and C. M. Krishna. New performance measures for design and evaluation of real-time multiprocessors. *Computer Systems Science and Engineering*, 1(4):179–192, Oct. 1986.

[9] K. G. Shin, C. M. Krishna, and Y.-H. Lee. Optimal dynamic control of resources in a distributed system. *IEEE Transactions on Software Engineering*, 15(10):1188–1198, Oct. 1989.

[10] J. A. Stankovic and K. Ramamritham. *Hard Real-Time Systems*. IEEE Computer Society Press, Los Angeles, CA, 1988.

[11] R. W. Wolff. Poisson arrivals see time averages. *Oper. Res.*, 30(2), Mar.-Apr. 1982.