Performance Analysis of Client-Server Systems with Aggressive Server Replies

Greg Franks C. M. Woodside

Department of Systems and Computer Engineering
Carleton University
Ottawa, Ontario, Canada, K1S 5B6

Abstract

Client-Server systems generally use the send-receive-reply messaging paradigm for inter-process communication. In the most general case, the server process can be structured to continue to execute after replying to the client. Analytic models for closed queueing networks have only recently appeared, but they suffer from poor accuracy in some instances. A new approach is presented here which improves the accuracy of previous approaches, integrates into the standard Mean Value Analysis algorithms for layered queueing models, and broadens the scope of systems that can be modelled. Two examples are given showing both the effects of aggressive replies, and the improvements in approximation accuracy.

1 Introduction

The send-receive-reply pattern of communication is widely used in distributed systems, often using an RPC (remote procedure call) service. This imposes waiting on the sender or client in the interaction, which can be reduced by sending the reply as early as possible – what we will call here "aggressive" replies. To perform an aggressive reply, any server operations which can be done later are postponed until after replying; the deferred work is termed a "second phase of service" [19, 10]. This will give performance advantages when the client executes on a different processor from the server, since it lets the two processes proceed in parallel. Examples of second-phase operations include delayed writes to stable storage in database systems, logging of non-critical data, and deallocation of resources [4, page 131]. Smith and Williams [17] give an example of real-time system performance engineering of a design which includes a second phase. Many existing distributed operating systems support second phase execution, including Ameoba [18], Chorus [16], V [2], and Sun RPC [4]. Second phase service is also directly supported in the programming language Ada [1].

The two phases of service are illustrated in Figure 1. The time interval from the instant of the receive at the server to the time of the reply is described as phase one; the server execution after the reply, is phase two. From the standpoint of the customers (clients), the server's phase two is a vacation – a client cannot receive service during this period. When a client makes a request while the vacation caused by its own previous service request is still underway, we call it "overtaking" (see Figure 2).

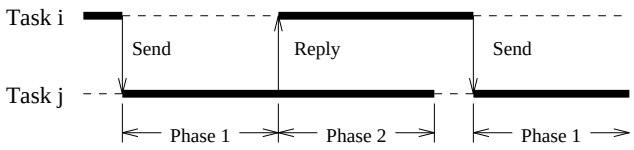

Figure 1: Non-overtaking event

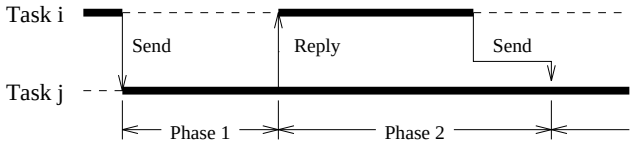

Figure 2: Overtaking event

This kind of system has been thoroughly analyzed for open systems as type GNENP (General Non-Exhaustive Service, Non-Preemptive), SV (Single Vacation) [5, §2]. An early version of the model, the "walking server" [7, Chapter 2] has been used to model drum-based memories [9] and other computer devices, but in general there has been little work on closed models apart from [10]. The special form of closed model called a "Rendezvous Network" [20, 21, 12] or a Layered Queueing Network [14, 15, 13] has second phases and the references given contain a simple analysis of their effects. Both approaches used essentially the same approximation, which for this paper will be called the "old overtaking approximation".

This work was stimulated by finding that the old overtaking algorithm gives large approximation errors in certain important cases. These tend to occur in deeply layered systems, which occur in information

systems and which have been described, for instance, by Febish and Sarna [6]. Even in a large system, the middle layers may have a server with just a few clients, in which both the clients and the server have important second phases. The old approximation ignored the phases of the clients while the new approximation takes these into account. The present paper is intended to give insight into the whole overtaking phenomenon, which has not been much studied but which will be more important in the future, then to show where second phases and aggressive replies give performance advantages, and finally to present a much better approximation. In the test cases the new overtaking approximation reduced the peak errors in throughput from the order of 50% to less than 6%. The evaluation of the new approximation is confined to layered systems, but they depend on solving a series of queueing networks, so the approximation should be equally good in closed queueing networks with second phases, in general.

2 Performance Enhancement through Aggressive Server Replies

This section will show how system throughput improves by using second phases at servers. Two cases are considered, a very simple system consisting of a set of clients calling a single server, and a more complex case where multiple two-phase clients call three two-phase servers.

2.1 Single Server, Single Phase Client

The client-server system system shown in Figure 3 will be used to show the potential performance improvements achieved by using aggressive replies. The large parallelograms in the figure represent tasks, the parallelogram labeled `entry` is a service entry or port on the server task that accepts messages from clients. All tasks run on unique processors. The numeric label on the client task represents its service time. The numeric label on the arc "`request`" represents the number of visits to the server per cycle of the client, or visit ratio, and the label on the entry `entry` represents the service time per request.

The system was studied by varying the visit ratio from the client to the server, the number of clients and the fraction of phase two service. Varying the number of visits varied the demand at the server so that the second phase effect on throughput could be studied at various server utilizations. The total service time at the clients and the server were fixed at the values shown in the figure. However, the ratio of phase-one to phase-two service was varied from 100% (a product-form central server system) to 0% (a synchronous send as is employed in transputers [8]).

Figure 4 shows the results for the different configurations and were obtained by solving the underlying Markov chain for the entire system using GreatSPN [3]. Points on the graphs represent equivalent demand from the clients; demand increases from left to right. For all cases, a second phase of service improves throughput (hence lowers response time), especially when the server is moderately loaded. At very high and very low server loading, phase two service does not contribute substantially to improved performance.

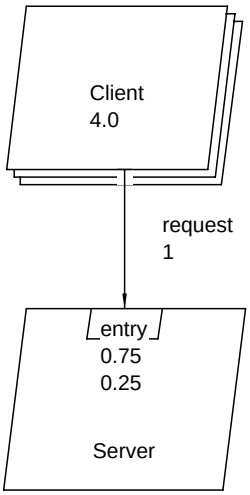

Figure 3: Simple two-level client-server system.

2.2 Multiple Servers, Two Phase Clients

The client-server system shown in Figure 5 is used to demonstrate the effect of aggressive replies when sending to multiple servers and when the client itself has more than one phase. This system was also solved for various levels of visits from the client and for various numbers of clients. This configuration can arise in deeply nested client-server systems where the task '`dispatcher`' receives requests by way of '`c0`' from higher-level clients.

For this system, the number of clients, the ratio of phase-one to total service time, and the visit ratios were varied, although the ratio of requests to the three servers was fixed. The total service time at the clients and servers was fixed at the values shown in the figure.

The results of the performance experiments are shown in Figure 6 with the results for all but the 10 customer case being exact. The results for the 10 customer case were generated using simulation due to state space problems with the Markovian solver. The simulation results were generated with 95% confidence intervals of $\pm 1\%$. For most cases, the throughput is improved with increasing amounts of phase-two service.

Two curious effects are apparent in the graphs. First, for small numbers of clients, the curves turn under at the right hand end. That is, the utilization at the servers drops as the visit ratio increase. The client throughput also drops. Second, the system with 100% of its service in phase two may have lower utilization and throughput than the systems with smaller amounts of phase two service for the same request rate. Both effects are caused by the overtaking phenomenon shown in Figure 2. As the visit ratio from the dispatcher task increases there is a larger probability that the task will overtake itself, thus increasing

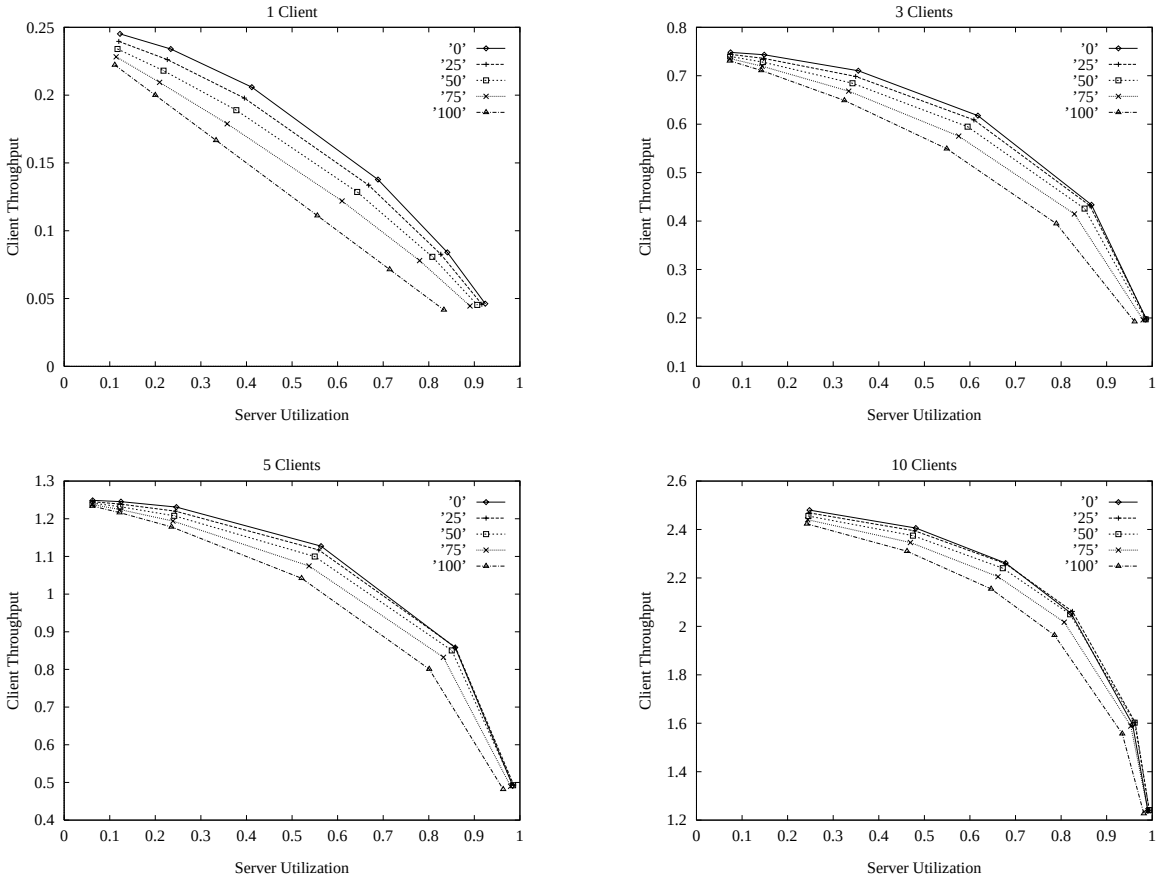

Figure 4: Throughput at client. The curves represent the ratio of phase-one to total service time, with '100' being a product-form FCFS server.

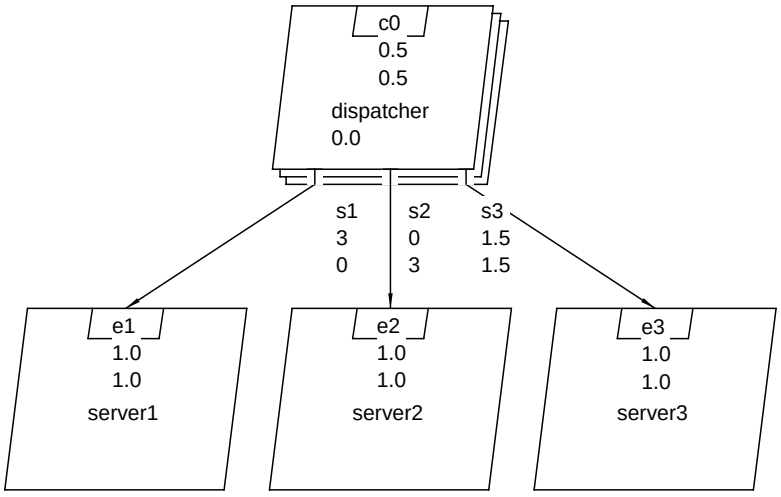

Figure 5: Complex Client-Server system.

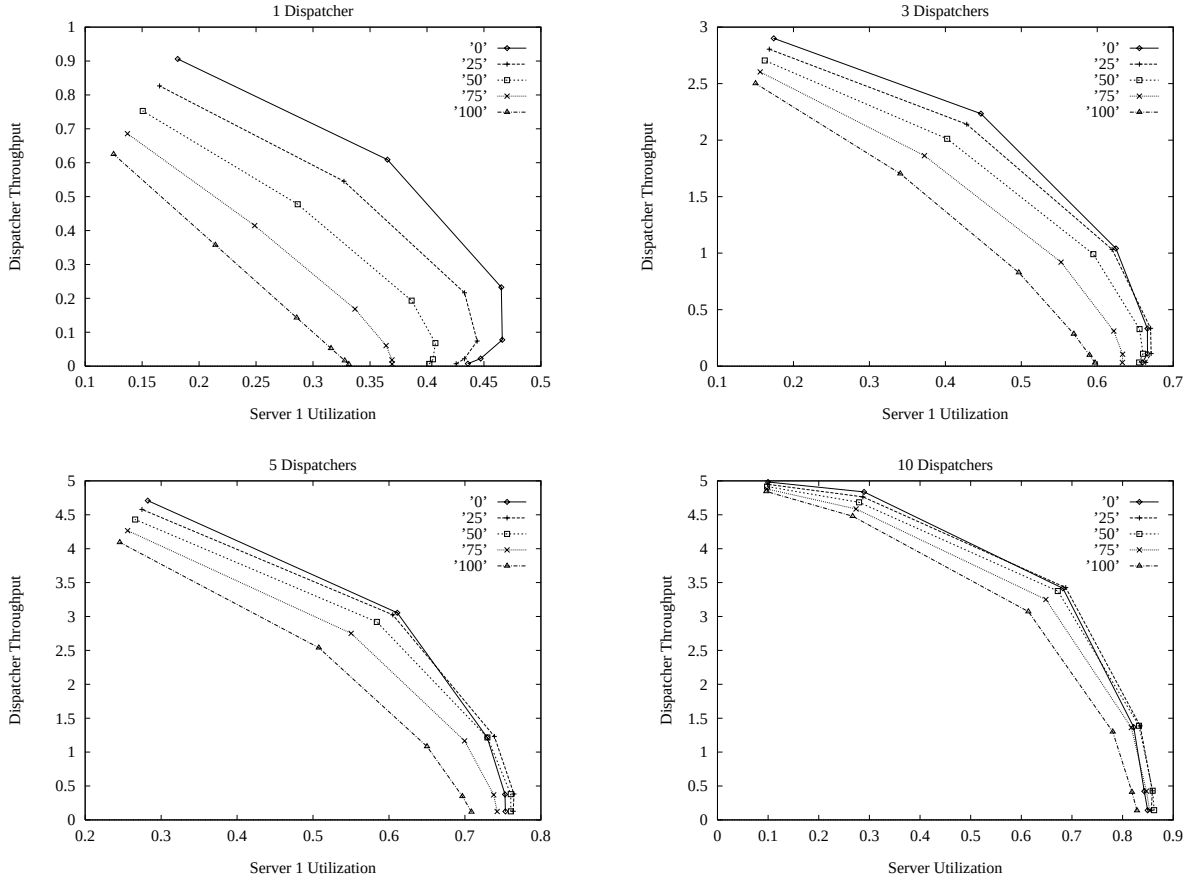

Figure 6: Throughput at client. The curves represent the ratio of phase-one to total service time, with '100' being a product-form FCFS server.

the waiting time and decreasing the throughput. However, the overtaking effect is stable; utilization drops as the demand increases.

3 Solution

The results from the previous section demonstrate that aggressive replies during a remote procedure call improves throughput and shortens response times even for heavily loaded servers. This section will give the new approximation for computing the performance effects.

3.1 Analysis of Overtaking

This section summarizes the previous overtaking approximation analysis, and shows its weaknesses, before introducing a more complete approximation. The argument centers on the mean-value equation for the mean total delay $W_m(N)$ of a customer arriving at a queueing station labeled m

$$W_m(N) = s_m + s_m L_m(N-1) \qquad (1)$$

where s_m is the delay for the customer's own service, and $s_m L_m(N-1)$ is the delay for other customers present at the arrival instant. For a two phase server, the second term is unchanged, but the first term is replace by two new terms as follows:

$$W_m = s_{m1} + \Pr\{\mathrm{OT}_m\} s_{m2} + s_m L_m(N-1) \qquad (2)$$

where:

- s_{m1} is the mean delay for service, which is just the phase-one service time, and
- $\Pr\{\mathrm{OT}_m\} s_{m2}$ is the probability of the customer arriving during the phase-two vacation caused by its own previous visit to the server multiplied by the phase-two service time.

This expression will be approximate, since phase-two systems do not satisfy the product-form assumptions, and the overtaking term also assumes exponentially-distributed phase-two times, to make the remaining phase-two time equal to its mean.

After a departure of a customer, the probability of it overtaking depends on the outcome of a race between the completion of phase two (mean delay of s_{m2}) and the return of the customer. We will assume that

the return delay has a mean rate of τ_m and is also exponentially distributed for solution convenience. The probability of the return event winning the race and causing an overtaking event is:

$$\Pr\{\mathrm{OT}_m\} = \frac{\tau_m^{-1}}{(\tau_m^{-1} + s_m^{-1})}$$

which simplifies to

$$\Pr\{\mathrm{OT}_m\} = \frac{s_{m2}}{\tau_m + s_{m2}}$$

The mean return time can be calculated if we know the mean arrival rate of requests from this particular customer, λ_m, and the mean total delay at the server, W_m, as

$$\tau_m = \frac{1}{\lambda_m} - W_m$$

giving:

$$\Pr\{\mathrm{OT}_m\} = \frac{s_{m2}}{s_{m2} + 1/\lambda_m - W_m} \qquad (3)$$

Equations (2) and (3) make up the old overtaking approximation.

3.1.1 Error with Two-Phase clients

Now we consider some client tasks which themselves have second phases. They might be servers to higher-level tasks in a deeply layered system, as discussed in the Introduction. Figure 5 shown earlier is an example in which this approximation behaves quite badly when the server has a significant fraction of its execution in the second phase. The principle reason for the large error is that (3) does not consider the variation in the rate of arrivals for different sending phase of the client. Solving (3) with the model in Figure 5 where $\lambda_m = 0.204$ and $W_m = 0.589$ results in $\Pr\{\mathrm{OT}\} = 0.231$. However, the actual overtaking probability to `server1` and `server2` is 0.535 and to `server3`, 0.354.

Figure 7 shows the percent relative error[1] in throughput using (2) and (3) for the example system when the proportion of phase-one to phase-two service time was varied from 0 to 100% for a variety of clients. Very large errors occur even for moderate amounts of phase 2 service.

3.2 The New Approximation

The new approximation begins with a state space for the client i and the server j, following a departure of the client from the server, shown in Figure 8. The server is in phase 2 at the time of the departure; the chain terminates when either the server finishes its second phase, or the client makes another request.

The states are named with a two digit combination (pk) where p is the phase of the client i and k takes the values of 0 to 3. Since we again assume that the phase-two duration has an exponential distribution,

[1] Relative error $= \frac{\text{exact}-\text{estimate}}{\text{exact}} \times 100$.

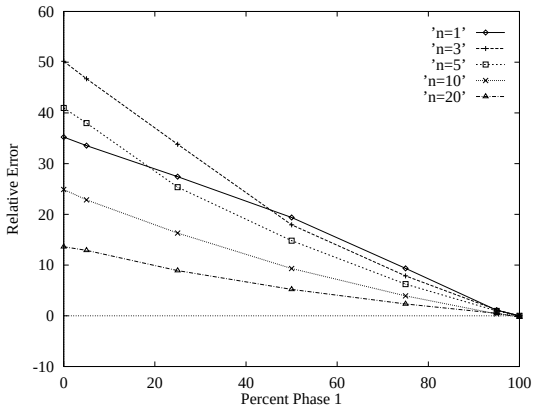

Figure 7: Percent Relative Error for the multiple servers, two-phase client system using the old overtaking approximation. The request rate and service times used correspond to the values in the Figure 5.

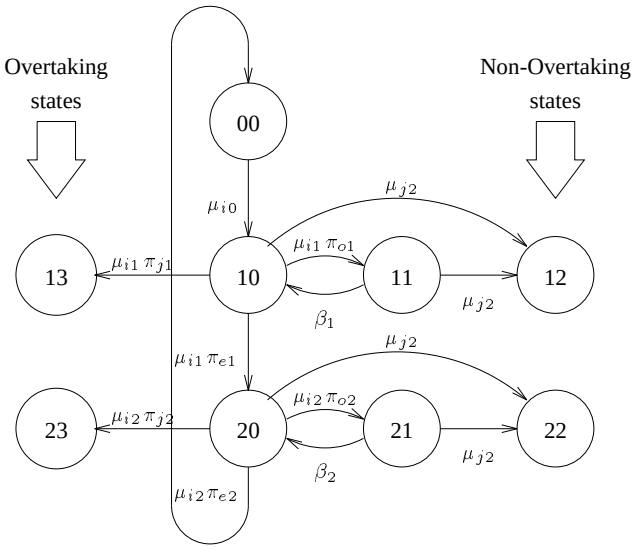

Figure 8: Continuous time Markov chain for the new overtaking calculation. States 10, and 20 represent the phase of the client while it is executing, states 13, and 23 represent overtakes, states 11 and 21 represent time blocked at other servers, states 12 and 22 represent non-overtakes, and state 00 represent idle time by the client..

the server j imposes a rate $\mu_{j2} = s_{m2}^{-1}$ of transition from every state to a state where it has completed without an overtake. This takes the state from 10 to 13, for example.

The client i may go through several changes before a potential overtaking event. It can complete phase 1 and enter phase 2; it can then become idle waiting for the request to reach it, before reaching phase 1 again. We will say that the client is in phase 0 when it is idle. Further, during each phase p the client task may be executing on its own (in state $p1$) or blocked waiting for some other server (in state $p2$). At the end of a period executing on its own it may request the particular server that is being analyzed for overtaking, and this gives a transition from state $p1$ to an "overtaken" state $p3$. This is a continuous-time Markov chain (CTMC) which can be easily solved, if the times in states are assumed to be exponentially distributed, and this is assumed in the following.

State $p1$ and its associated transitions serves an important purpose. The overtaking probability is heavily dependent on the race between the completion of phase-two, and the next request from the client to the server. If the time spent calling other servers is merged into the rate parameter μ_{i1}, i.e., by removing state $p1$, the race may be skewed significantly in favour of the server. Consequently, the overtaking probability may also be heavily underestimated.

To provide the parameters to this Markov chain, we need rates calculated from the remainder of the performance model. In the iterative scheme used in solving layered systems these rates are available from the previous iteration, or in the initialization of the calculation. As well as μ_{j2}, the rate of server's phase 2 execution, we require:

μ_{ip} = the mean execution rate of the client i in phase p between requests to servers,

β_p = the mean delay of the client i when blocked on a server other than server j in phase p during a remote procedure call.

μ_{i0} = the inverse of the mean idle time of the client i,

π_{ep} = the probability that, at the end of an execution-interval of the client i, the phase ends rather than a new request being made to a server,

π_{op} = the probability that, at the end of an execution-interval of the client i, a request is made to a server other than server j, and

π_{jp} = the probability that, at the end of an execution-interval of the client i, a request is made to the particular server j.

Given the values, we can determine the probability of terminating the chain in states 13 and 23, which give overtaking, or in states 12 and 22, which do not, by solving the first-passage equations for the Markov chain [11].

A closed form solution for overtaking is given in the Appendix, as follows. First, it finds $\Pr(S_p|S_r)$, which is the probability of overtaking on a request from phase 'p' of the client while the server is still processing a request from phase 'r'.

$$\Pr(S_p|S_r) = \frac{c_p \prod_{y=r}^{p-1} \frac{b_y}{1-d_y}}{1 - \left(b_p \prod_{x \neq p} \frac{b_x}{1-d_x} + d_p \right)} \quad (4)$$

$$b_p = \frac{1}{1 + Y_{ip} + \mu_j s_{ip}}$$

$$c_p = \frac{y_{ijp}}{1 + Y_{ip} + \mu_j s_{ip}}$$

$$d_p = \frac{Y_{ip} - y_{ijp}}{1 + Y_{ip} + \mu_j s_{ip}} \cdot \frac{\mu_{kp}}{\mu_{kp} + \mu_j}$$

$$Y_{ip} = \sum_k y_{ikp}$$

$$\mu_{kp} = \frac{1}{\sum_k w_{ikp}}$$

$$\mu_j = \frac{\lambda_{ij}}{\sum_{p=2}^{n} \lambda_{ijp} s_{jp}}$$

where:

y_{ijp} is the mean number of calls from client i to server j during phase p of i,

w_{ijp} is the mean waiting time for the client i to the server j during phase p, and

λ_{ij} is the mean throughput from the client i to the server j during phase p.

The final overtaking probability for client i calling server j is then found using:

$$\Pr\{\mathrm{OT}(i,j)\} = \sum_{p=1}^{n} \sum_{r=0}^{n} \frac{\lambda_i y_{ijr}}{\lambda_{ij}} \Pr(S_p|S_r) \quad (5)$$

3.2.1 Waiting Time Calculation

For mean value analysis, the queue length L_m:

$$L_m = v_m W_m \frac{N}{\sum_i (v_i W_i(N)) + Z} \quad (6)$$

where v_i is the number of visits to station i and Z is the think time. The W_m term in (6) for the two-phase server is calculated using the phase-one service time plus overtaking. However, the server is, in fact, busy for both the phase-one and phase-two service time intervals. During the next step of mean value analysis, i.e. when the population is increased by one customer, the value of L_m is taken to be $L_m(N-1)$ in (2). This value is not correct because of the unaccounted for second phase service time while the server

is busy and the client is not overtaking. The difference $s_{m2} - \Pr\{\mathrm{OT}_m\}s_{m2}$ must be added back into the station to fully account for the vacation of the server.

The $L_m(N-1)$ term in (2) represents both the queue at the server plus its utilization:

$$L_m(N-1) = L_m^*(N-1) + U_m(N-1) \qquad (7)$$

The $U_m(N-1)$ term itself can be split into two components, the phase-one and phase-two utilization, of which, the phase-two utilization includes the overtaking component. Therefore, to correct the $L_m(N-1)$ for the server's utilization when the client does not overtake, the following term must be added to (7):

$$(1 - \Pr\{\mathrm{OT}_m\})U_{m2}(N-1) \qquad (8)$$

Equation (2) becomes:

$$\begin{aligned} W_m &= s_{m1} + \Pr\{\mathrm{OT}_m\}s_{m2} + s_m L_m(N-1) \\ &\quad + (1 - \Pr\{\mathrm{OT}_m\})U_{m2}(N-1) \end{aligned} \qquad (9)$$

3.3 Generalizations to the Markov Chain

The Markov chain shown in Figure 8 can be extended to increase its generality and accuracy in a number of ways. The two considered in this paper extend the chain to handle more than two client and two server phases.

3.3.1 Multi-phase Clients

Additional client phases can be added to the Markov chain by adding the subchain shown in Figure 9 for each additional phase of the client. The subchains are connected together from state $p0$ to state $(p+1)0$ with transition of rate $\mu_{ip}\pi_{ep}$. The final state $p0$ is connected to state 00.

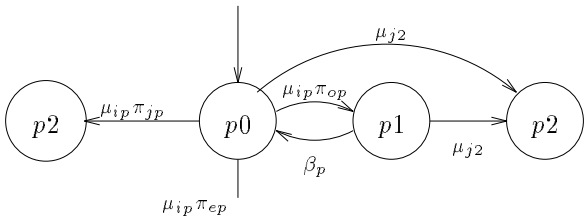

Figure 9: Subchain for extending the Markov chain Figure 8 for multiple client phases.

3.3.2 Multi-phase Servers

Additional server phases can be accommodated by modifying the subchain for a phase of the client (Figure 9) in the manner shown in Figure 10. States are now named with a three digit combination ijk where i is the phase of the client, j is the phase of the server and k takes the values of 0 to 2. State 10 in the original chain in Figure 8 becomes state 120 in the new chain, which similar changes for for states 11 and 13. States 110, 111 and 113 do not exist because the server is never in phase 1 for this analysis.

State 13 in the original chain denoted the completion of phase 2 for the server without overtaking; this state is renamed to 130 with outgoing transitions for the server completing phase 3, the client sending to the server, and the client sending to some other server. State 140 becomes the new non-overtaking terminal state for a three-phase server.

3.3.3 Other extensions

The transient Markov chain used for this approximation assumes exponential time distributions for all of the rate parameters. The accuracy could be further improved by breaking down the server's phase into a set of stages which represent its own execution and blocking times. However, the state space would rapidly explode, so the addition is not considered further here.

4 Examples

The two examples presented earlier in section 2 will be used to show the improvements in accuracy using the new approximation.

4.1 Single Server, Single Phase Client

Figure 11 contains eight graphs, all of which plot the relative error in client throughput for the four cases in Section 2.1. The four graphs on the left side of the figure plot the relative error for the old overtaking approximation, while the four graphs on the right show the results for the new version. For the old approximation, the error is proportional to the ratio of second-phase service present at the server. The error peaks when the server is moderately utilized, which is also where the proportion of second-phase service has the greatest performance improvement effect. The new overtaking approximation is better than the old method for all cases, but exhibits errors which are roughly equivalent to the old approximation in cases where the server has almost all phase-two service and serves only a few clients. All of the examples exhibit the highest error when the server is heavily utilized.

4.2 Multiple Servers, Two Phase Sends

Figure 12 shows relative error in throughput for the system in Section 2.2 and using the same parameters. The approximations were compared against exact results except for the 10-customer case where simulation was used. The graphs show that the relative error for the new algorithm is much better than the old algorithm, especially when the server is heavily utilized. Furthermore, error for the new overtaking approximation tends towards zero for all four cases when the demand at the server is high whereas the old approximation only exhibits this effect when there are a large number of customers.

5 Conclusions

Client-server systems can benefit from the use of aggressive replies at the server because the clients in the system are blocked for shorter periods. Servers which

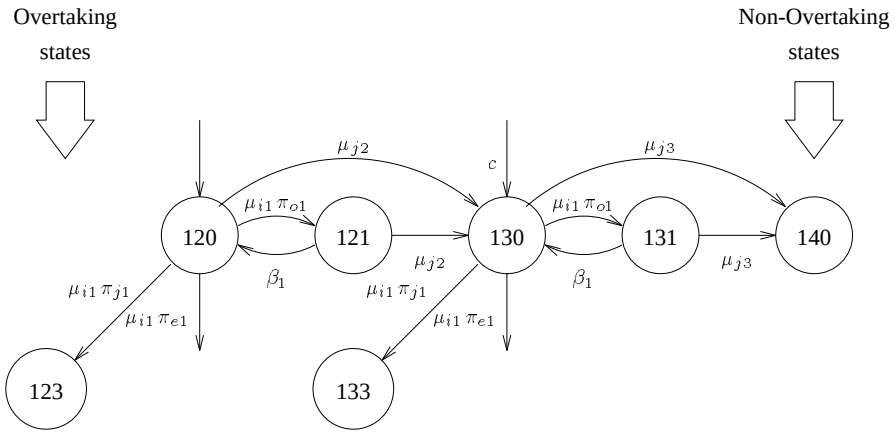

Figure 10: A single client phase subchain from Figure 8 extended for three server phases.

use aggressive replies complicate analytic modeling because the corresponding queueing network is no longer product form. Accurate models are necessary because the two-phase servers can have counter-intuitive characteristics when clients overtake execution in phase-two caused by their own earlier requests.

This paper presents a new algorithm for estimating the waiting time at two-phase servers which is, in most cases much more accurate than older methods, and never any worse. In some important cases where the server is heavily utilized, the accuracy is improved by an order of magnitude. The new approximation uses a simple closed-form expressions which is efficient for incorporation in an iterative algorithm for solving performance models of deeply layered client-server systems.

Acknowledgments

This research was funded by the Telecommunications Research Institute of Ontario (TRIO) and the Natural Science and Engineering Research Council of Canada (NSERC). Helpful discussions with Dorina Petriu are gratefully acknowledged. She and Shikharesh Majumdar, John Neilson and Jerome Rolia all have contributed to the overall model in which this calculation is used.

A Solution for Overtaking Probability

The overtaking probabilities, $\Pr(S_p|S_r)$, are found using the jump chain shown in Figure 13 which is based on the continuous-time Markov chain in Figure 8. The states are named with the two digit combination pk where p represents the phase of the client, $k = 4$ represents overtake events, and $k = 3$ non-overtaking events. The starting states are 00, 10 through $p0$ which correspond to the identically named states in the continuous Markov chain.

The rates for the transitions q are as follow:

q_{p0} = the probability that some other server k replies to client i before server j completes its phase-two service.

q_{p1} = the probability that client i completes its own execution interval before server j completes its phase two service.

q_{p2} = the probability that client i calls some other server other than server j.

q_{p3} = the probability that server j completes its own phase-two service before the client i finishes its execution interval.

q_{p4} = the probability that client i calls server j and overtakes.

q_{p5} = the probability that server j completes its phase two service before some other server k replies to client i.

q_{p6} = the probability that client i completes phase p and goes to phase $p+1$.

The rates are found using the following equations:

$$
\begin{aligned}
q_{p0} &= \frac{\mu_{kp}}{\mu_{kp}+\mu_j} \\
q_{p1} &= \frac{\mu_{ip}}{\mu_{ip}+\mu_j} \\
q_{p2} &= \frac{Y_{ip}-y_{ijp}}{Y_{ip}+1} \\
q_{p3} &= \frac{\mu_j}{\mu_{kp}+\mu_j} \\
q_{p4} &= \frac{y_{ijp}}{Y_{ip}+1} \\
q_{p5} &= \frac{\mu_j}{\mu_{ip}+\mu_j}
\end{aligned}
$$

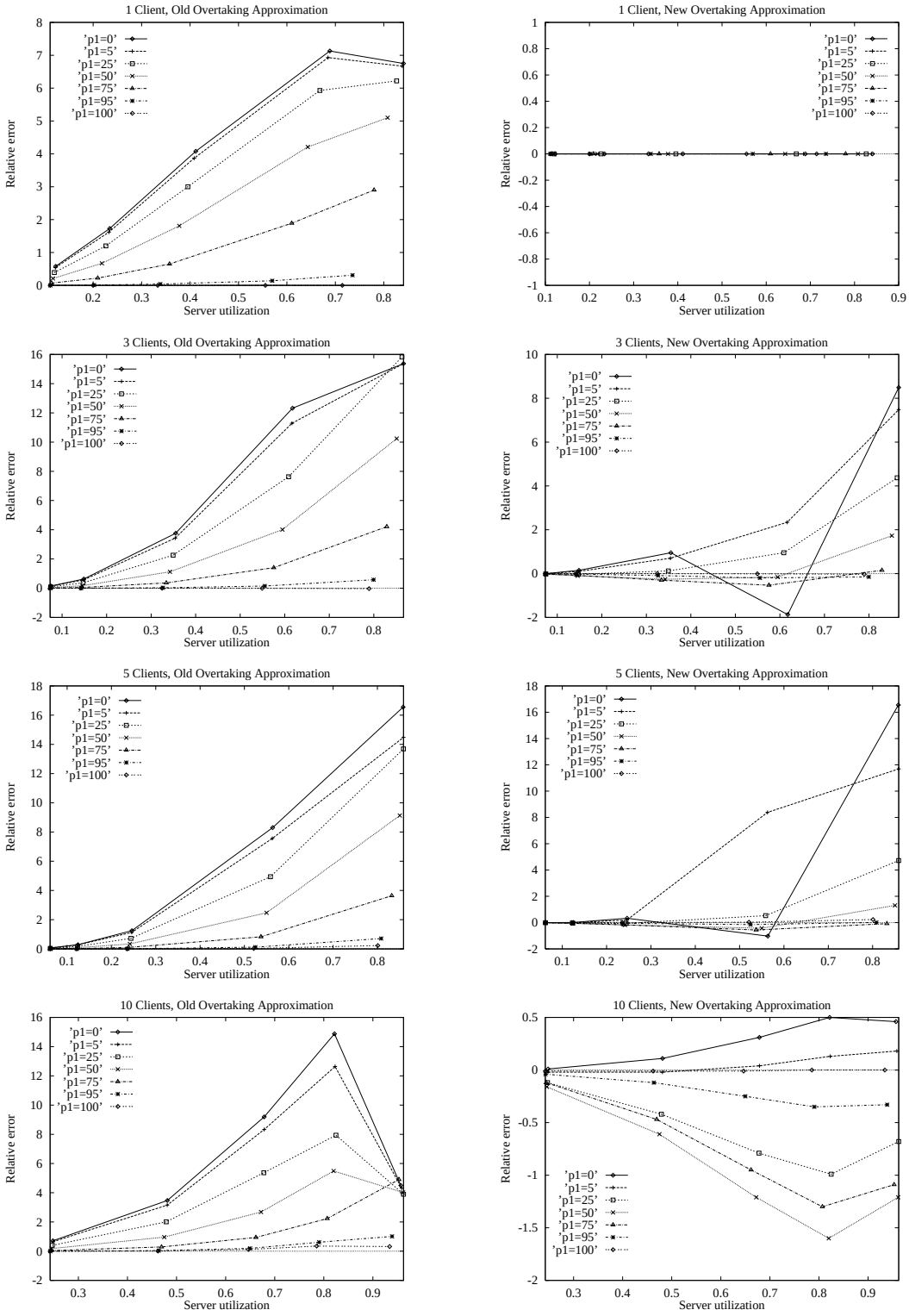

Figure 11: Relative error versus server utilization for the Single-Server, Single Phase Client system in §2.1. Curves on the graphs represent the percentage of phase 1 service time with 'p1=100' being a product-form queueing network.

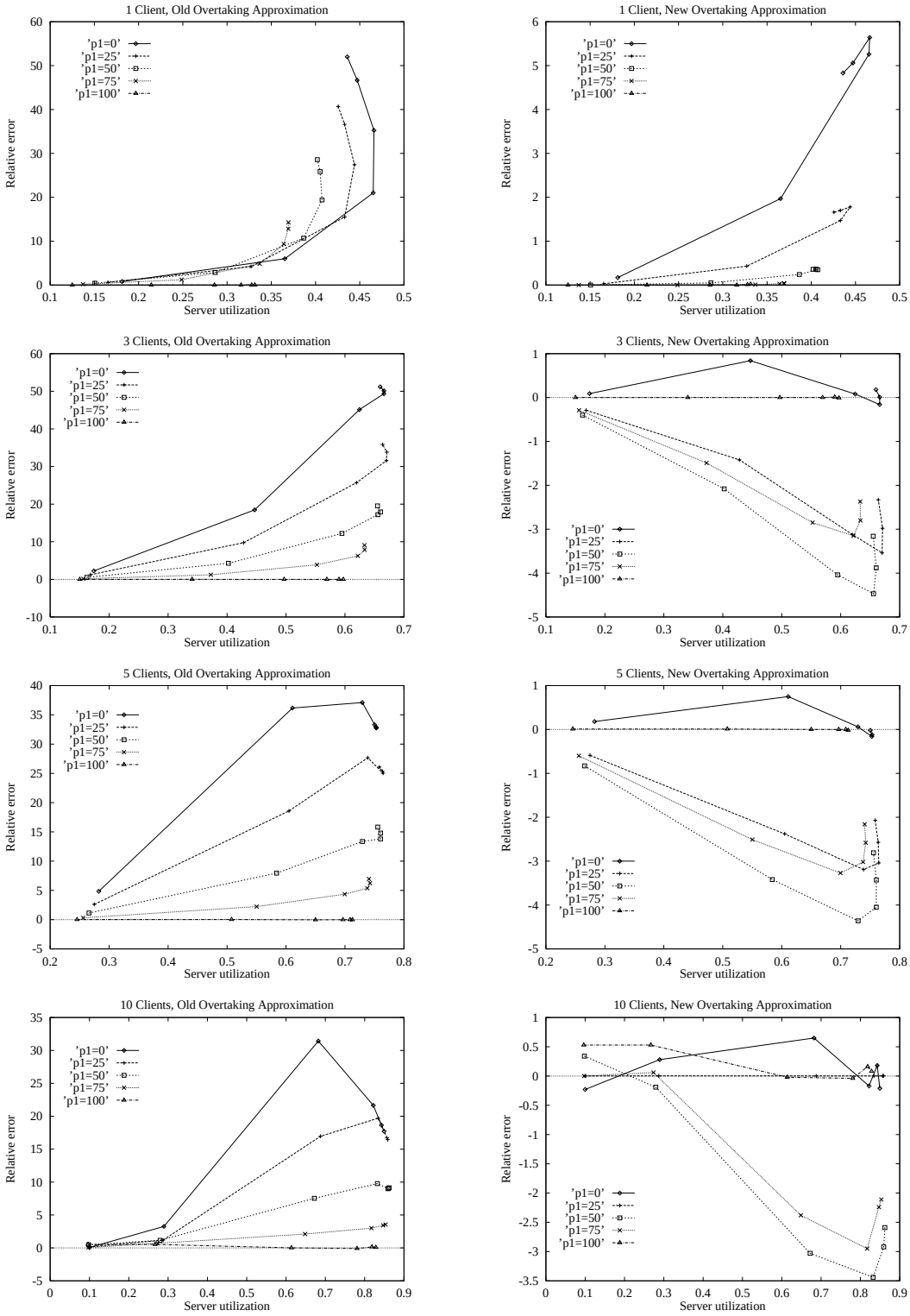

Figure 12: Relative error versus server utilization for the Three Server, Two-Phase Client system in §2.2. Curves on the graphs represent the percentage of phase 1 service time with 'p1=100' being a product-form queueing network.

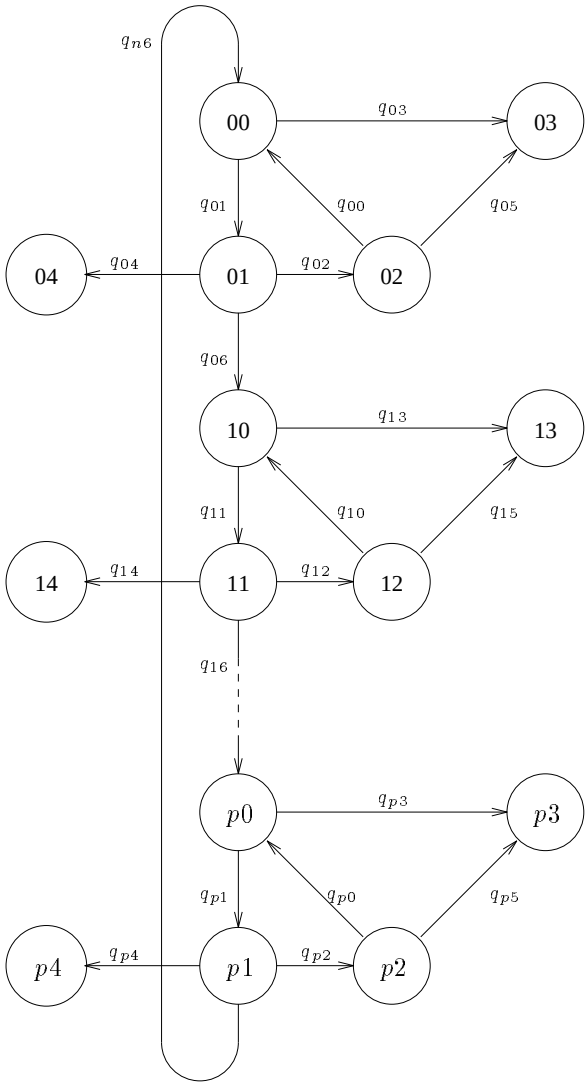

Figure 13: Jump chain for finding overtaking probabilities. The states are named with the two digit combination pk where p represents the phase of the client, $k = 4$ represents overtake events, and $k = 3$ non-overtaking events.

$$q_{p6} = \frac{1}{Y_{ip} + 1}$$

Next, the terms b_p, c_p and d_p which make up the solution, (4), are found using:

$$\begin{aligned} b_p &= q_{p1}q_{p6} \\ c_p &= q_{p1}q_{p4} \\ d_p &= q_{p0}q_{p1}q_{p2} \\ a_p &= q_{p5} + q_{p1}q_{p2}q_{p3} \end{aligned} \tag{10}$$

Equation (10) is used to find $\Pr(S_{p4}|S_{r0})$, the probability no overtaking occurs when the client makes a request from phase p after a reply to a request from phase r. This probability is used as the starting probability for overtaking during phase three of the server when the server has more than two phases. To find $\Pr(S_{p4}|S_{r0})$, replace c_p with a_p in (4).

References

[1] *The Programming Language Ada: Reference Manual*, volume 155 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1983.

[2] D. R. Cheriton. The V distributed system. *Communications of the ACM*, 31(3):314–333, March 1988.

[3] G. Chiola. A graphical Petri net tool for performance analysis. In Serge Fdida and Guy Pujolle, editors, *Modelling Techniques and Performance Evaluation*. Elsevier Science, Amsterdam, March 1987.

[4] John R. Corbin. *The Art of Distributed Applications: Programming Techniques for Remote Procedure Calls*. Springer-Verlag, New York, 1991.

[5] Bharat Doshi. Single server queues with vacations. In Hideaki Takagi, editor, *Stochastic Analysis of Computer and Communication Systems*, pages 217–265. North Holland, Amsterdam, 1990.

[6] G. J. Febish and D. E. Y. Sarna. Building three-tier client-server business solutions. White paper, Object Soft. Corp., Englewood, NJ, 1995.

[7] E. Gelenbe and Mitrani I. *Analysis and Synthesis of Computer Systems*. Computer Science and Applied Mathematics. Academic Press, Toronto, 1980.

[8] M. Homewood, D. May, D. Shepherd, and R. Shepherd. The IMS T800 transputer. *IEEE Micro*, 7(5):10–26, October 1987.

[9] Stephen S. Lavenberg, editor. *Computer Performance Modeling Handbook*, volume 4 of *Notes and Reports in Computer Science and Applied Mathematics*. Academic Press, Toronto, ON, 1982.

[10] Louis-Marie Le Ny and C. Murray Woodside. Performance modelling of queues with rendezvous service. Technical Report 941, Institut National de Recherche en Informatique et en Automatique (INRIA), Domaine de Voluceau, Rocquencourt, B.P.105, 78153 Le Chesnay Cedex, France, December 1988.

[11] Emanuel Parzen. *Stochastic Processes*. Holden-Day, San Francisco, 1962.

[12] Dorina C. Petriu. Approximate mean value analysis of client–server systems with multi-class requests. In *Proceedings of the 1994 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems.*, pages 77–86, Nashville, TN, U.S.A., May 1994. A.C.M. SIGMETRICS.

[13] J. A. Rolia and K. C. Sevcik. The method of layers. Submitted for publication in *IEEE Transactions on Software Engineering*, 1994.

[14] Jerome Alexander Rolia. Performance estimates for systems with software servers: The lazy boss method. In Ignacio Casas, editor, *VIII SCCC International Conference On Computer Science*, pages 25–43, Santiago, Chile, July 1988. Chilean Computer Science Society.

[15] Jerome Alexander Rolia. *Predicting the Performance of Software Systems*. PhD thesis, Univerisity of Toronto, Toronto, Ontario, Canada. M5S 1A1, January 1992.

[16] M. Rozier, V. Abrossimov, F. Armand, I. Boule, M. Gien, M Guillemont, F. Herrmann, C. Kaiser, S. Langlois, P. Léonard, and W. Neuhauser. Overview of the chorus distributed operating systems. Technical Report CS/TR-90-25, Chorus Systèmes, February 1991.

[17] Connie U. Smith and Lloyd G. Williams. Software performance engineering: A case study including performance comparison with design alternatives. *IEEE Transactions on Software Engineering*, 19(12):720–741, July 1993.

[18] Andrew S. Tanenbaum, Robbert Renesse, Staveren Hans, Gregory J. Sharp, Sape J. Mullender, Jansen Jack, and Guido Rossum. Experiences with the Amoeba distributed operating system. *Communications of the ACM*, 33(12):46–64, December 1990.

[19] C. M. Woodside, E. Neron, E. D.-S. Ho, and B. Mondoux. An "active server" model for the performance of parallel programs written using rendezvous. *Journal of Systems and Software*, pages 844–848, 1986.

[20] C. Murray Woodside. Throughput calculation for basic stochastic rendezvous networks. *Performance Evaluation*, 9:143–160, 1989.

[21] C. Murray Woodside, John E. Neilson, Dorina C. Petriu, and Shikharesh Majumdar. The stochastic rendezvous network model for performance of synchronous client-server-like distributed software. *IEEE Transactions on Computers*, 44(8):20–34, August 1995.