5. Investigations into Simalytic Modeling

Section *4.2 Construction of a Simalytic Model* on page 100 presented a detailed analysis of a two server model example. This section expands the idea of a Simalytic Model and describes the results of models with additional servers. The presentation is similar to section *4.2.5 Implementation Example* on page 115, but without the detailed business discussion. For each of the following examples, an abbreviated description of the model creation process is shown. In these examples, the nodes are shown simply as servers S1 through S3, S4 or S8, as appropriate for each example. The transaction workloads are represented as T1 and T2, with T1 being the workload of interest[4]. The T2 workload needs to be included only to the extent it impacts the T1 workload and to understand what impact increasing the arrival rate of T1 has on T2 response times. The arrival rate of T2 is kept constant at 0.1 arrivals per second. Only the arrival rate of T1 is changed to represent growth in that workload. To keep this example simple, it is also assumed that all of the transactions that execute on a server use the same resources. This means that there is no difference on S1 between T1 transactions that route to S2 and those that don't. It also means there is no difference between the transactions that execute on S2 (i.e. a T1 transaction routed to S2 consumes the same resources on S2 as a T2 transaction initially

[4] The notation used to identify the transaction workloads in this chapter differs slightly from that used in the research results and models. The transaction workloads are identified as T1 and T2 so the reader can more easily distinguish them from the servers S1 and S2. However, the original research results identify these workloads as S1 transactions and S2 transactions because they are the transactions initially sent to the S1 and S2 servers, respectively. Therefore, throughout this chapter, the appendices and the raw data, the terms “T1”, “T1 transactions” and “S1 transactions” are interchangeable and the terms “T2”, “T2 transactions” and “S2 transactions” are interchangeable.

started on S2). Because this is a hypothetical client/server environment, there are no actual measurements. Therefore, the results of a pure simulation model of the environment are used to represent these measurements.

When designing the configurations to use for the examples, it quickly became apparent that there truly are an infinite number of possible combinations of workloads, servers, routings and static workload arrival rates (the constant value used for the arrival rate of all workloads other than T1, the workload of interest). Unfortunately, the results up to this point in the research were very consistent and did not show any anomalies to indicate any preferred configurations to investigate. Therefore, the configurations (the server service times, the routing percentages and the application topologies) were selected generally at random with some attempt to provide results that could be compared to the earlier MathCAD results. The simulation models were constructed first and then adjusted to provide a well formed response time curve within the arrival rate range of 0.01 to 5.0 arrivals per second with an objective of having the model near saturation at 5.0 arrivals per second. Once the desired configuration was established for each of these three examples, the Simalytic Models were constructed. In no case were any of the simulation models modified once work on the Simalytic Model for that example had begun.

The queuing theory models used for the servers are the same for all three models and are detailed in section *7.2 Appendix B: Queuing Theory Models* on page 164. Selected results from multiple runs of this model are shown in *Table 10 OpenQN Response Times* on page 167. The servers used in each of the models are those discussed in section *7.7 Appendix G: Tool Baseline Comparison Results* on page 215 and have service times of 0.1, 0.5, 1.0 and 1.5 seconds. The correlation between the OpenQN models and the

Simul8 models for these servers is quite good and is shown in *Figure 62 Multi-device Server Baseline Chart* on page 216.

The results of both the simulation models and the Simalytic Models for the three server, the four server and the eight server configurations are shown in *Table 6 Three Server Example Response Time Results* on page 134, *Table 7 Four Server Example Response Time Results* on page 138 and *Table 8 Eight Server Example Response Time Results* on page 142, respectively. Each data point is the average of ten trials for each arrival rate to reduce the impact of arrival distributions. The arrival rate refers to the arrival rate for T1. The arrival rate for T2 is kept at a constant value because it is not the workload of interest. Each trial was for 3600 simulation seconds (one simulation hour) with a 100 second warm-up period (300 seconds for the eight server model because of the increased complexity). The Simalytic Function was created using Microsoft's Visual Basic. For these models, it is a very simple function that calculates the cumulative average of the interarrival times for each workload and looks up the corresponding response time in a table. The Simalytic results track the simulation results in all three cases. The slight under predicting is consistent with the simple implementation of the Simalytic Function as discussed in *3.5.3 Validation of the Mathematical Foundation* on page 86. It is also discussed in section *4.2.5.5 Simalytic Model Example* on page 121 and in *Figure 31 Simalytic Function Comparison* on page 123.

The Simalytic Models for these examples are shown in section *7.3 Appendix C: Simalytic Models*, starting on page 168, as *Figure 56* on page 176, *Figure 57* on page 176 and *Figure 58* on page 177. The same Simalytic Function was used for all three examples and is shown in *Figure 55 Visual Basic Code for Simalytic Function* on page 170.

The replication factor for each server is set to 10,000 to avoid all queuing. Statistics were collected on the queue length for each server in each Simalytic Model to insure no additional queuing was introduced. The value of 10,000 was used after an occasional maximum queue length of one was found. All Simalytic Models in the section show a maximum queue length of zero for every server with this replication value.

Two methods were used to verify that these models are accurate representations of the applications. The average response times (over ten trials) for each model at the lowest arrival rate were compared to the calculated service times for that model and to the same results from the simulation models. The calculated service times, shown in *Table 5*, derived from the sum of the product of the service time and percentage of visiting transactions for each server. For example, the service time calculated for T1 in the three server model would be 100%*0.1+60%*1.0+(60%*70%)*1.5=1.33 (refer to *Figure 32 Three Server Configuration* on page 132 for the routing information). The simulation results are shown in *Table 6*, *Table 7* and *Table 8*. The Simalytic Model results are very close to the expected values in both comparisons (slight variations are due to the differences between the theoretical and actual routing percentages at such very low arrival rates), which shows these models are accurate representations of the applications.

	3 Server Model	4 Server Model	8 Server Model
T1 Transactions	1.33	1.55	2.57
T2 Transactions	2.50	2.25	3.63

Table 5 Calculated Service Times

Each of the following sections, *5.1 Three Server Example*, *5.2 Four Server Example* and *5.3 Eight Server Example*, are structured in the same way as section *4.2.5.1 Example Process Implementation* on page 117. In order to make each section readable on its own, much of the supporting text is repeated in each section. The significant differ-

ences are the references to the tables and figures and the analysis in the calibration step of the Simalytic Model phase. Once the reader is comfortable with the structure and flow of a section, the other sections can be reviewed quickly by referring just to the tables and figures and analysis.

5.1 Three Server Example

This three server example represents a simple application where all three servers are used by the T1 workload and only two are used by the T2 workload.

5.1.1 Three Server Example Workload Analysis

Identify: For this example, there are two transaction types: T1 and T2.

Document: Refer to *Figure 32* for complete information as to the topology of both workloads. This figure shows the routing of transactions out of each server as the percentage of the transactions that entered the server. The percentage for each transaction type will be 100%. A single arrow out of a server represents 100% of all transaction types.

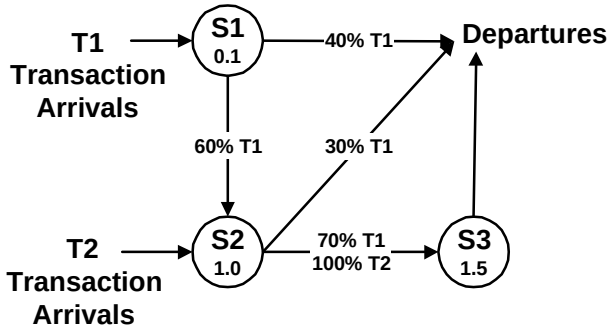

Figure 32 Three Server Configuration

Measure: Refer to the simulation results in *Table 6 Three Server Example Response Time Results* on page 134.

Correlate: The workload correlation is assumed.

5.1.2 Three Server Example Node Models

Build: The service times for the OpenQN model of each node (server) are shown in *Table 9 OpenQN Device Service Times* on page 166 (the server names reflect the service time for the server). This model uses servers S1, S2 and S3.

Calibrate: The model of each server is assumed to be calibrated for this example.

Run: The OpenQN models were run for each server.

Create: Partial results of the OpenQN model of each server are shown in *Table 10 OpenQN Response Times* on page 167.

5.1.3 Three Server Example Simulation Model

Build: The overall simulation model was built using Simul8 and is shown in *Figure 47 Three Server Simulation Model* on page 161. The service times are the same as those used in the OpenQN model shown in *Table 9 OpenQN Device Service Times* on page 166.

Set: This step was skipped for this example because it sets the simulation model service times for the next step, which is not required.

Calibrate: This step normally compares the simulation model to actual measurement response times. It is not required for this example because the simulation results are being used as the measurement data.

5.1.4 Three Server Example Simalytic Model

Create: The Simalytic Function shown in *Figure 55 Visual Basic Code for Simalytic Function* on page 170 was used for this example.

Replace: Replace the static service times. *Table 6 Three Server Example Response Time Results* shows the results of the model trials after the Simalytic Function was implemented. *Figure 56 Three Server Simalytic Model* on page 176 shows the Simalytic Model used for this example.

Calibrate: The results of the above model are compared to the pure simulation model results to calibrate the model as shown in *Figure 33 Three Server Comparison*. The lines **T1 Delta %** and **T2 Delta %** show how well the two modeling techniques correlate. The scale on the right side of the chart shows the objective of ±10% and all of the data points (except for the last one, 5.0 arrivals per second, which is close to model saturation) for both workloads are within the objective. The effects of the particular smoothing function implemented in the Simalytic Function are seen in the increased delta as the arrival rate increases. *Figure 31 Simalytic Function Comparison* on page 123 shows the impact of the function on the results correlation.

T1 Arrival Rate	Simulation T1 Transactions	Simulation T2 Transactions	Simalytic T1 Transactions	Simalytic T2 Transactions	T1 Delta %	T2 Delta %
0.01	1.31	2.58	1.37	2.58	4.1%	-0.2%
0.10	1.37	2.64	1.38	2.59	0.9%	-1.8%
0.20	1.41	2.65	1.38	2.60	-2.3%	-1.9%
0.50	1.50	2.82	1.45	2.75	-3.4%	-2.6%
1.00	1.64	3.12	1.57	3.01	-4.6%	-3.6%
2.00	2.04	3.98	1.98	3.82	-2.9%	-4.0%
3.33	3.24	6.49	3.10	6.11	-4.5%	-5.8%
3.70	4.02	8.06	3.79	7.56	-5.7%	-6.2%
4.00	5.21	10.70	4.77	9.61	-8.5%	-10.2%
5.00	88.09	199.72	45.42	91.43	-48.4%	-54.2%

Table 6 Three Server Example Response Time Results

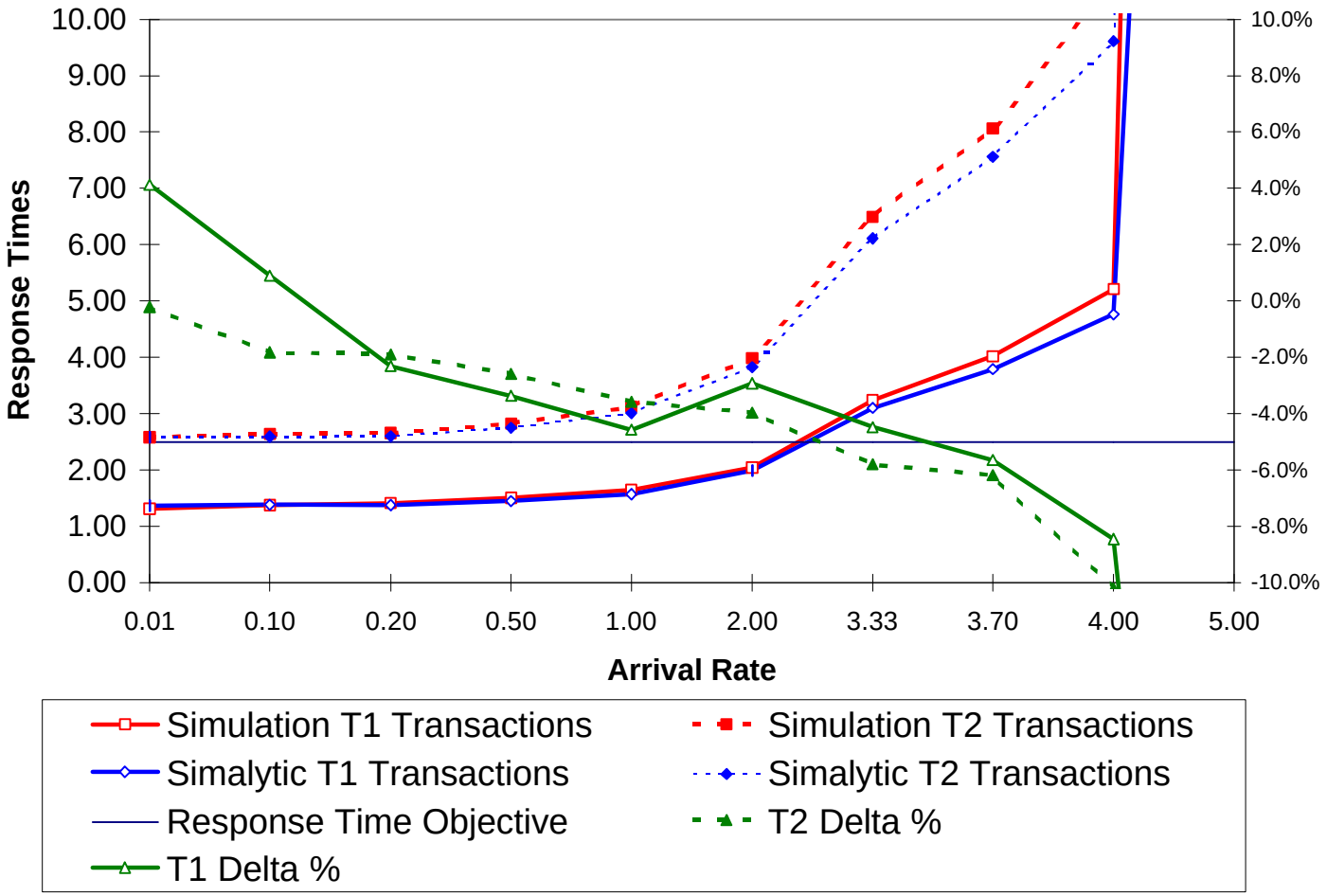

Figure 33 Three Server Comparison

5.2 Four Server Example

This four server example represents an application where all four servers are used by the T1 workload and three servers are used by the T2 workload.

5.2.1 Four Server Example Workload Analysis

Identify: For this example, there are two transaction types: T1 and T2.

Document: Refer to *Figure 34* for complete information as to the topology of both work-loads. This figure shows the routing of transactions out of each server as the percentage of the transactions that entered the server. The percentage for each transaction type will be 100%. A single arrow out of a server represents 100% of all transaction types.

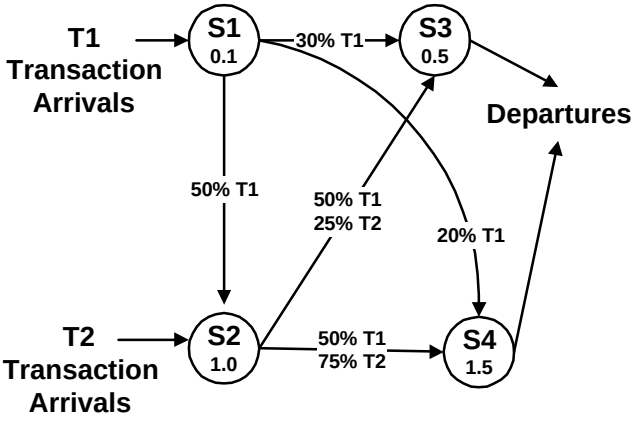

Figure 34 Four Server Configuration

Measure: Refer to the simulation results in *Table 7 Four Server Example Response Time Results* on page 138.

Correlate: The workload correlation is assumed.

5.2.2 Four Server Example Node Models

Build: The service times for the OpenQN model of each node (server) are shown in *Table 9 OpenQN Device Service Times* on page 166. This model uses servers S1, S2, S3 and S4.

Calibrate: The model of each server is assumed to be calibrated for this example.

Run: The OpenQN models were run for each server.

Create: Partial results of the OpenQN model of each server are shown in *Table 10 OpenQN Response Times* on page 167.

5.2.3 Four Server Example Simulation Model

Build: The overall simulation model was built using Simul8 and is shown in *Figure 48 Four Server Simulation Model* on page 162. The service times are the same as those used in the OpenQN model shown in *Table 9 OpenQN Device Service Times* on page 166.

Set: This step was skipped for this example because it sets the simulation model service times for the next step, which is not required.

Calibrate: This step normally compares the simulation model to actual measurement response times. It is not required for this example because the simulation results are being used as the measurement data.

5.2.4 Four Server Example Simalytic Model

Create: The Simalytic Function shown in *Figure 55 Visual Basic Code for Simalytic Function* on page 170 was used for this example.

Replace: Replace the static service times. *Table 7 Four Server Example Response Time Results* shows the results of the model trials after the Simalytic Function was implemented. *Figure 57 Four Server Simalytic Model* on page 176 shows the Simalytic Model used for this example.

Calibrate: The results of the above model are compared to the pure simulation model results to calibrate the model as shown in *Figure 35 Four Server Comparison*. The lines **T1 Delta %** and **T2 Delta %** show how well the two modeling techniques correlate. The scale on the right side of the chart shows a wider scale than the objective of ±10% because the last three data points are outside the objective. Except for the last data point (5.0 arrivals per second, which is close to model saturation) all of the data points for both workloads are either very close or within the objective. The effects of the particular smoothing function implemented in the Simalytic Function are seen in the increased delta as the arrival rate increases. *Figure 31 Simalytic Function Comparison* on page 123 shows the impact of the function on the results correlation.

T1 Arrival Rate	Simulation T1 Transactions	Simulation T2 Transactions	Simalytic T1 Transactions	Simalytic T2 Transactions	T1 Delta %	T2 Delta %
0.01	1.64	2.31	1.61	2.29	-1.9%	-0.5%
0.10	1.64	2.37	1.58	2.32	-3.4%	-2.0%
0.20	1.66	2.39	1.59	2.32	-4.3%	-3.0%
0.50	1.72	2.54	1.65	2.44	-3.8%	-4.0%
1.00	1.85	2.73	1.79	2.67	-3.4%	-2.3%
2.00	2.26	3.44	2.16	3.26	-4.5%	-5.2%
3.33	3.47	5.53	3.17	5.02	-8.7%	-9.3%
3.70	4.46	7.12	3.96	6.31	-11.4%	-11.4%
4.00	6.48	10.63	5.45	9.00	-16.0%	-15.4%
5.00	120.47	203.58	26.15	43.43	-78.3%	-78.7%

Table 7 Four Server Example Response Time Results

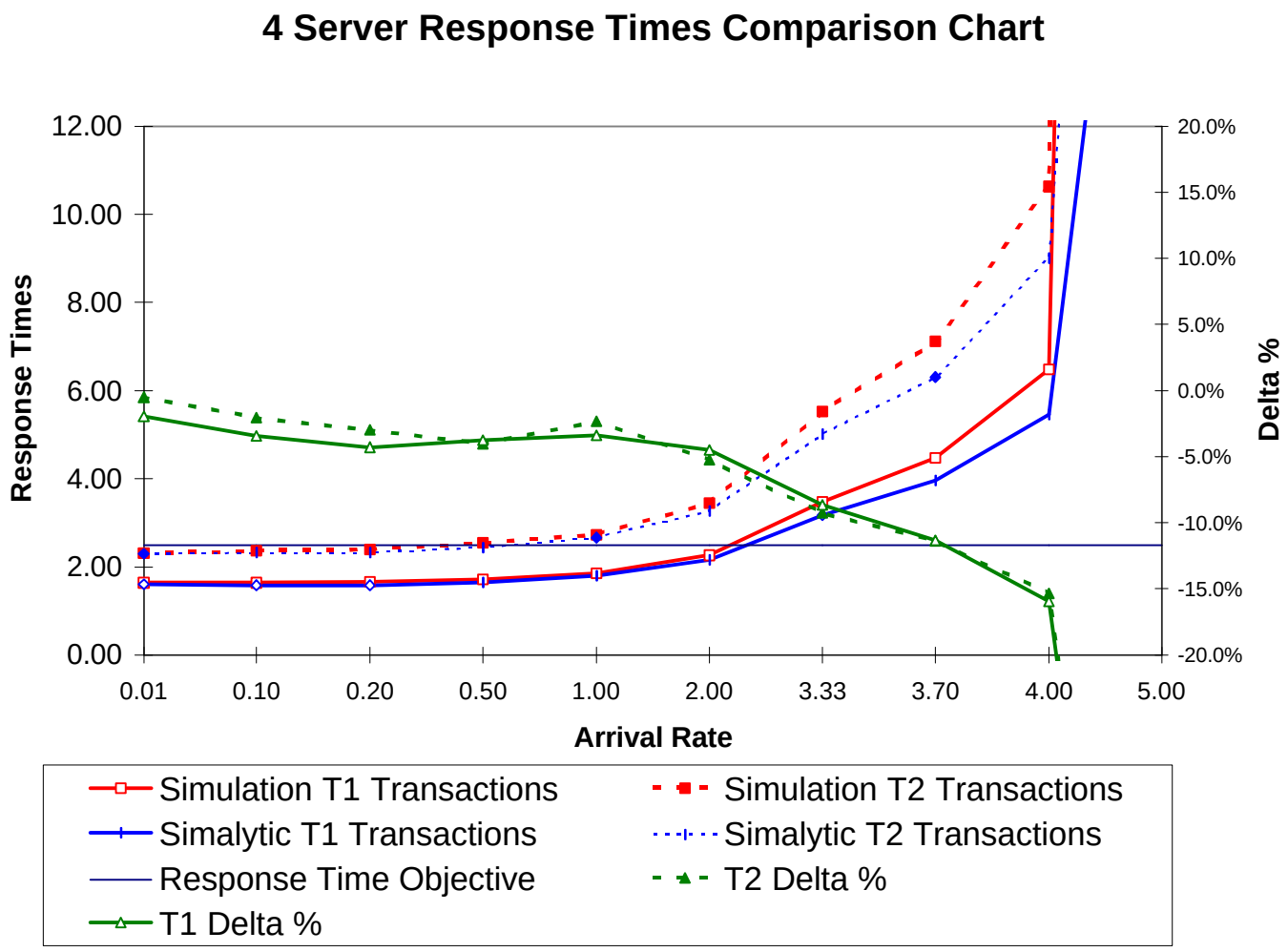

Figure 35 Four Server Comparison

5.3 Eight Server Example

This eight server example represents a complex application where transactions visit servers both routed and in series. Seven of the eight servers are used by the T1 workload and three servers are used by the T2 workload.

5.3.1 Eight Server Example Workload Analysis

Identify: For this example, there are two transaction types: T1 and T2.

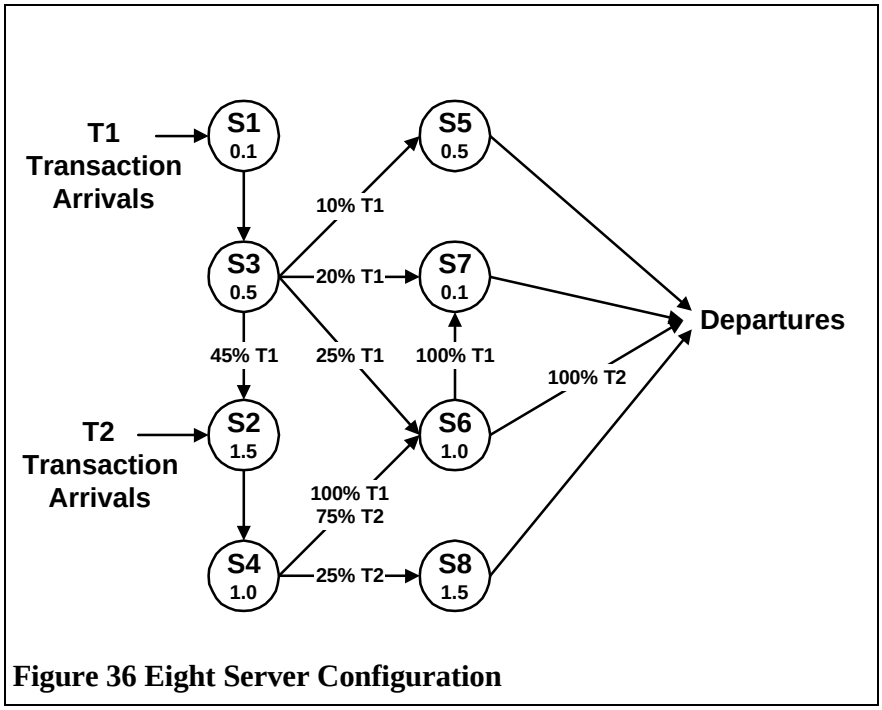

Figure 36 Eight Server Configuration

Document: Refer to *Figure 36* for complete information as to the topology of both workloads. This figure shows the routing of transactions out of each server as the percentage of the transactions that entered the server. The percentage for each transaction type will be 100%. A single arrow out of a server represents 100% of all transaction types.

Measure: Refer to the simulation results in *Table 8 Eight Server Example Response Time Results* on page 142

Correlate: The workload correlation is assumed.

5.3.2 Eight Server Example Node Models

Build: The service times for the OpenQN model of each node (server) are shown in *Table 9 OpenQN Device Service Times* on page 166. This model uses servers S1, S2, S3, S4, S5, S6, S7 and S8. In order to keep the OpenQN modeling data at a manageable level, four of these servers are assumed to have the same service times and to generate the same response time results as four other servers. Therefore, in this example, there are two servers with a service time of 0.1 seconds (S1 and S7), two servers with a service time of 0.5 seconds (S3 and S5), two servers with a service time of 1.0 seconds (S4 and S6), and two servers with a service time of 1.5 seconds (S2 and S8). This was done to keep this example simple enough to present in a reasonable manner. There is no indication that this simplification caused any bias in the model, either positive or negative. Additional benefits of this simplification are reduced construction effort of the simulation and Simalytic models as well as consistency with the earlier MathCAD results as discussed in section *7.7 Appendix G: Tool Baseline Comparison Results* on page 215.

Calibrate: The model of each server is assumed to be calibrated for this example.

Run: The OpenQN models were run for each server.

Create: Partial results of the OpenQN model of each server are shown in *Table 10 OpenQN Response Times* on page 167.

5.3.3 Eight Server Example Simulation Model

Build: The overall simulation model was built using Simul8 and is shown in *Figure 49 Eight Server Simulation Model* on page 163. The service times are the same as those

used on the OpenQN model shown in *Table 9 OpenQN Device Service Times* on page 166.

Set: This step was skipped for this example because it sets the simulation model service times for the next step, which is not required.

Calibrate: This step normally compares the simulation model to actual measurement response times. It is not required for this example because the simulation results are being used as the measurement data.

5.3.4 Eight Server Example Simalytic Model

Create: The Simalytic Function shown in *Figure 55 Visual Basic Code for Simalytic Function* on page 170 was used for this example.

Replace: Replace the static service times. *Table 8 Eight Server Example Response Time Results* shows the results of the model trials after the Simalytic Function was implemented. *Figure 58 Eight Server Simalytic Model* on page 177 shows the Simalytic Model used for this example.

T1 Arrival Rate	Simulation T1 Transactions	Simulation T2 Transactions	Simalytic T1 Transactions	Simalytic T2 Transactions	T1 Delta %	T2 Delta %
0.01	2.78	3.71	2.55	3.72	-8.0%	0.1%
0.10	2.66	3.78	2.64	3.73	-0.7%	-1.4%
0.20	2.71	3.83	2.64	3.75	-2.4%	-2.1%
0.50	2.83	4.01	2.73	3.94	-3.4%	-1.8%
1.00	3.10	4.41	3.00	4.32	-3.1%	-2.2%
2.00	3.87	5.44	3.63	5.24	-6.2%	-3.6%
3.33	6.37	9.26	5.92	8.63	-7.1%	-6.8%
3.70	8.38	12.44	7.40	11.07	-11.8%	-11.0%
4.00	12.90	19.78	11.20	17.30	-13.2%	-12.6%
5.00	184.20	346.42	55.12	86.37	-70.1%	-75.1%

Table 8 Eight Server Example Response Time Results

Calibrate: The results of the above model are compared to the pure simulation model results to calibrate the model as shown in *Figure 37 Eight Server Comparison*. The lines **T1 Delta %** and **T2 Delta %** show how well the two modeling techniques correlate. The scale on the right side of the chart shows a wider scale than the objective of ±10% because the last three data points are outside the objective. Except for the last data point (5.0 arrivals per second, which is close to model saturation) all of the data points for both workloads are either very close or within the objective. The effects of the particular smoothing function implemented in the Simalytic Function are seen in the increased delta as the arrival rate increases. *Figure 31 Simalytic Function Comparison* on page 123 shows the impact of the function on the results correlation.

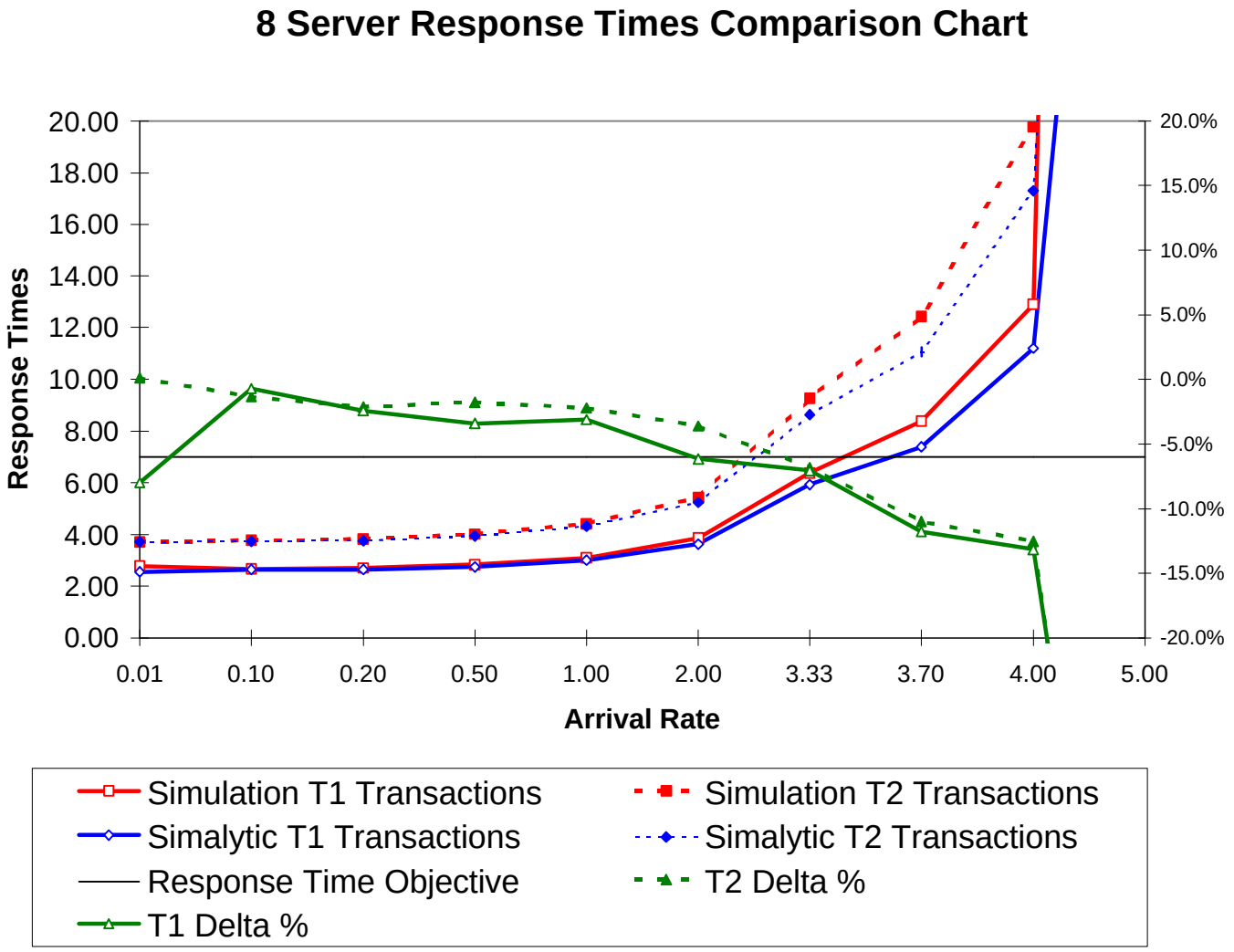

Figure 37 Eight Server Comparison

5.4 Multiple-Server Examples Summary

This section has presented three additional example models of applications with arbitrary topologies. It has shown how the results of the Simalytic Models track the results of the simulation models with the same degree of correlation as prior examples. The Simalytic Models are slightly under predicting, as discussed in earlier sections, because of the simple Simalytic Function used. Even with this under prediction, it is still appropriate to use the Simalytic Models because they are either within, or very close to, the objective of ±10% of the simulation results. When the Simalytic Model results are adjusted by the factor of 1.05 (the same as used in section *3.5.3 Validation of the Mathematical Foundation* on page 86), all of the data points are within the objective except for 4.0 arrivals per second in the four server example (which is very close at -11.8% for T1 and -11.1% for T2) and the saturation point (5.0 arrivals per second) in all three examples.

All three of these examples provide additional verification of the results of Simalytic Models as compared to simulation models of the same applications. The focus of this verification is that the results are consistently precise and accurate. The precision is evident by the clearly delineated predictions within a narrow range for the given input values for all three example models. The accuracy is evident by the predictions being consistently close to the simulation results. Also, additional techniques are available to improve the accuracy, but at the cost of increased complexity and effort. These model examples (three servers, four servers and eight servers), along with the two server example presented earlier, show consistently precise and accurate results across what is assumed to be the entire range of practical situations, and thus verify the predictive qualities of Simalytic Models.

6. Conclusion

The traditional view of planning the capacity of a system is evolving because of the desire to predict the performance of the application. Applications designed to exploit a client/server architecture greatly increase the complexity of both the computer system configurations and the applications themselves. Predicting the responsiveness of those more complex applications requires a more complex modeling methodology. But adding complexity to a modeling effort also adds time, effort and cost. There are many techniques and tools that are beginning to address this evolution, but none of them can provide the desired level of detail for every situation and every application.

Modeling an application at the enterprise level requires an understanding of the applications and measurements of the transaction response times. Different modeling techniques (simulation, analytic queuing theory or hybrid) and different modeling tools (platform-centric or general purpose) can be used to predict transaction response times for individual systems or servers. But none of these can be used alone to produce a detailed enterprise level model at a reasonable development cost.

By following the steps for implementing a Simalytic Model, the modeler can rapidly produce an application model at the level of detail needed to make business decisions. Combining different modeling techniques (simulation and analytic queuing theory) and different modeling tools (platform-centric and general purpose) allows rapid analysis with reusable components created with the most applicable tools to reduce the time, effort and cost of developing an enterprise application model. As more detailed results are required, more sophisticated tools can then be used to increase the understanding of critical sections

of the model. This level of analysis provides insight into the application's future performance that would not otherwise be available. Using the Simalytic Modeling Technique both improves the understanding of the application as well as identifies which systems require more detailed analysis. It protects the investment an organization has made in training and in the acquisition of existing tools. It allows the most appropriate tools to be used for each modeling effort. Capacity planning is still fundamental to business success. But just as application designs are moving away from single system solutions, modeling for capacity planning must move away from single system analysis and begin predicting the application across the enterprise.

6.1 Hypothesis

The hypothesis as stated in section *1.8.2 Hypothesis* on page 19 is: It is possible to develop a viable modeling methodology that will use a general purpose simulation modeling tool as an underlying framework and utilize the results of an analytic modeling tool to represent individual nodes or systems when predicting the capacity requirements of an application at the enterprise level.

6.2 Summary of Proof of Hypothesis

This hypothesis has been proven by the development of a hybrid modeling methodology, The Simalytic Enterprise Modeling Methodology. The proof includes the actual implementation of several models using the technique in addition to the development of the methodology and the process to implement the methodology. To accomplish this, the methodology was developed and presented at several levels.

The first level was that of basic mathematical formulae. The Simalytic Modeling formula is presented in section *3.3 Foundation* on page 61, which is derived from the simulation formula (*2.2.2 Simulation* on page 34) and the queuing theory formula (*2.2.1 Analytic Queuing Theory* on page 30). A transform function, the Simalytic Function, was developed to enable the results of the queuing theory formula to be used in the simulation formula, thus creating the Simalytic formula.

At the next level, these formulae were implemented using a mathematical analysis tool, MathCAD, to verify that the three formulae (simulation, queuing theory and Simalytic) produce acceptably similar results. The results, presented as graphs in section *7.5 Appendix E: MathCAD Formulae Results Charts* on page *180*, show a high degree of correlation between the three formulae. The results of the queuing theory formula were presented in surface plots to show that the relationships between the variables used in all three of the formulae (arrival rate, service time and response time) are a smooth curve across the values of interest and thus interpolation between sample results is reasonable.

The third level is the step-by-step process developed in section *4.2.4 Steps to Build a Simalytic Model* on page 102. Each step in the process was identified and discussed. The process to implement a Simalytic Model was shown to include the same steps that are required to implement a simulation model with the addition of those steps to implement the Simalytic Function.

The fourth level is the implementation of an example of a two server application, using the process developed in section *4.2.4*, with a commercial simulation tool for the Simalytic Model framework. Use of this tool was shown to be valid because the results of

this tool were shown to produce near identical results to the MathCAD implementation, as shown by the charts in section *7.5 Appendix E: MathCAD Formulae Results Charts* on page *180*. Section *4.2.5.1 Example Process Implementation* on page 117 presented the details of the step-by-step implementation and the results of both the simulation model and the Simalytic Model.

The last level is the set of experiments, presented in section *5 Investigations into Simalytic Modeling* on page 127, that extended the two server implementation from section *4.2.5.1* with additional servers. This level showed that the Simalytic Modeling Methodology is extensible and can be generalized to what is considered a large-scale client/server enterprise.

This multi-level approach, as presented in *Figure 38 Summary of Proof*, has shown a consistent progression from the mathematical foundation to the practical implementa-

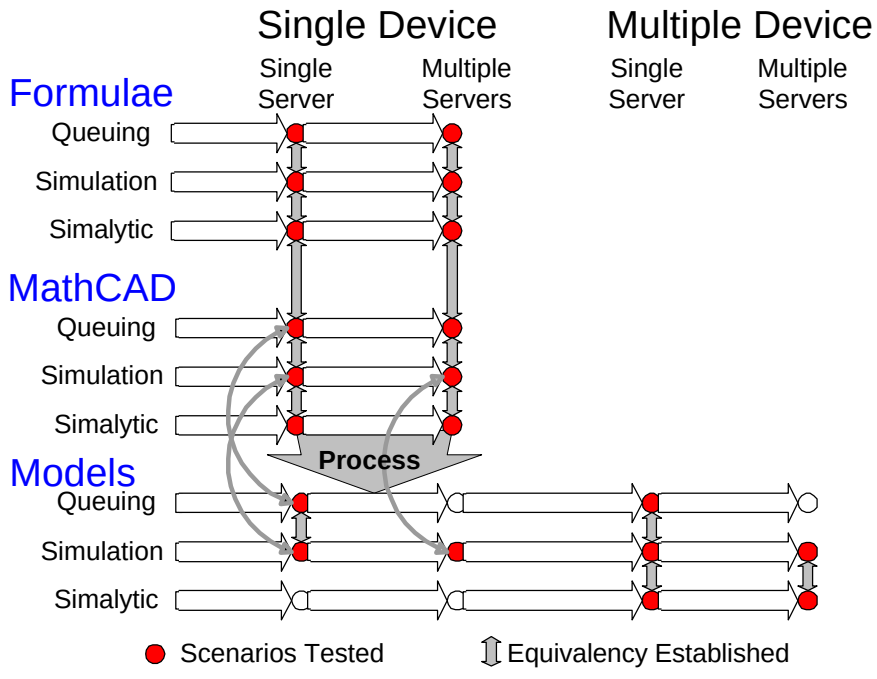

Figure 38 Summary of Proof

tion. At each level, the Simalytic Modeling Methodology has been compared to a simulation implementation. In each case, it has been shown to be a valid technique and the results have been verified.

6.2.1 Validation of Proof

Each level has shown the Simalytic Modeling Methodology valid by comparing the results of a Simalytic Model to the results of the same environment implemented with a simulation model. In all modeling cases, the Simalytic Model has been shown to produce equally acceptable enterprise model results within the bounds of reasonable usage. By verifying the consistency of the results across what is assumed to be the entire range of practical situations, Simalytic Modeling has been shown to be a valid technique for solving that set of problems.

6.2.2 Verification of Proof

At each level, the consistency of the results between the Simalytic Models and the simulation models has been verified by implementing the same environment using both techniques. Analysis of these results shows that the Simalytic Model returns clearly delineated (i.e. precise) predictions. These consistently precise results, that is, results within a narrow range for the given input values across what is assumed to be the entire range of practical situations, verify the predictive qualities of Simalytic Models.

6.3 Application

A practical implementation of the Simalytic Modeling Technique is to develop an interface between existing general purpose simulation tools such as SES/Workbench (SES), Qase (AST), Pro-Sim (MSI), ProModel (ProModel), CSIM (MS) or Simul8

(Visual) with existing platform-centric queuing theory tools such as Best/1 (BGS), Opti-Model (CA-Legent), CMF/MODEL (B&B), ISM/CP (ISM) or ATHENE (MSI). In addition, other tools could be used in place of the platform-centric analytic submodels, including design engineering models such as SPE*ED (PES; Smith 1995) and general analytic tools such as QSolver/1 (Menascé, Almeida, and Dowdy 1994) and CAPS (ACR). This partial list of tools is not intended to include all tools that could be used with this technique. This list represents some of the tools the author has had some level of experience with and believes should be usable in building a Simalytic Model. Some tools, such as QNAP II, Modline (SAS), Qase and SES/strategizer (SES), have some level of both simulation and analytic model solvers (Pooley 1995; Smith 1995). Both the list of simulation tools and the list of analytic tools are large enough to show that the Simalytic Modeling Technique is a general methodology with broad application and not a specialized implementation of a single tool.

6.4 Future Research

The two major areas for future research in Simalytic Modeling are tool integration and Simalytic Function refinement.

6.4.1 Tool Integration

The Simalytic Modeling Methodology presented here uses the results of a queuing theory modeling tool that has been run independently. The natural follow-on to this research would be to integrate the queuing theory tools directly into the simulation tools. However, one of the major advantages to Simalytic Modeling is that it allows the use of any combination of simulation and queuing theory tools. Integrating a single queuing the-

ory tool into a single simulation tool would remove this significant advantage. The solution to the apparent paradox is to extend the current research in exchanging model descriptions between tools (Smith and Williams 1995) by Dr. Connie Smith to include dynamic interaction during execution.

6.4.2 Simalytic Function Refinement

The work thus far in Simalytic Modeling has shown the Simalytic Function to be usable for capacity planning given a reasonable set of assumptions and expectations of accuracy. Furthermore, the accuracy of the resulting Simalytic Model is directly dependent on the Simalytic Function implementation. Additional research into different implementation techniques and the use of additional information available from each server in the simulation framework could improve the accuracy, expand the applicable problem set and address client/server situations neither identified nor anticipated in this paper.

6.5 Concluding Remarks

There are very few academic works that discuss more than just the fundamentals of capacity planning in general and, more specifically, modeling systems to predict future capacity requirements. The majority of work in this area of computer science has been related to commercial products that are either on the commercial market or were being developed by the authors during their academic studies. Examples are: the Buzen Estimator evolved into Best/1 (BGS) (from Dr. Jeff Buzen's dissertation); PEDAS (Goettge 1990) evolved into Qase (AST) and SPE*ED (PES) was developed by Dr. Connie Smith from the work presented in her book (Smith 1990). Even textbooks with a greater academic format, such as (Menascé, Almeida, and Dowdy 1994), have somewhat of a com-

mercial influence. Dr. Menascé sells a more robust version of the QSolver/1 tool included with the book. Even the more theoretical papers have a commercial influence because their authors are almost always affiliated with a company selling either a commercial modeling tool or computer hardware. Examples are: (Baker and Shih 1992) with an affiliation to the IBM T.J. Watson Research Center, and (Lipovich 1995) with an affiliation to BGS Systems, Inc (BGS). This discussion does not mean to imply that there is anything improper with any of these associations, either the ones above or any others. The point is that capacity planning has been viewed as an implementation or support issue, not as an area of research. The methodology, and associated technique, described in this dissertation is not only usable for capacity planners in industry, but also supported by an academic foundation and sound research. It is hoped that this dissertation will be used very cost-effectively in the practical world, as well as encourage additional research in capacity planning and performance analysis.

7. Appendices

7.1 Appendix A: Simulation Models

The following figures are screen images of the simulation models used throughout this paper to generate the baseline simulation results. They were implemented in Simul8 (Visual), which is a general purpose simulation tool. The symbols used are:

-  The small computer represents either a single server or the CPU (central processing unit) in a multi-device server model.
-  The floppy disk symbol represents any type of disk device. The fact that a floppy disk symbol was used is not meant to imply the servers use floppy disks, only that it is a more recognizable symbol and represents some type of disk.
-  The bin symbol, a partially full tub with an arrow pointing into it from the left, is the queue for whatever symbol is directly to its right. Simul8 requires a queue for every server to avoid transactions being lost if one arrives when the server is busy. Every server will have a queue although they are sometimes hidden in the image of the screen to reduce the complexity of the image.
-  The input symbol, a rectangle with an arrow, shows where transactions are created and controls the arrival distributions, which can be either internal (derived from a mathematical distribution) or external (from actual trace data).
-  The multi-block symbol represents a zero delay server and queue sub-model. The only function of this server is to provide a count of the transactions entering the

system for the reports. These servers do not add any delay to the transaction response times.

-  The user symbol, a person at a desk, combines the function of the input symbol and the multi-block symbol above to reduce the number of symbols on the more complex models. Functionally it is exactly the same as those two symbols.
-  The transactions complete symbol, a list with a check, collects statistics on each of the completed transactions.

A solid line connects each pair of servers to show the transactions flow. Although not a requirement of Simul8, in the models presented here the transaction flow is generally left to right and from top to bottom. It is always from the input symbol(s) to the transactions complete symbol(s). Transactions routing logic is not shown on the screen and is available only by opening the modification dialog for the given server.

The following figures, *Figure 39* through *Figure 49*, are screen images of the Simul8 models used throughout this paper, showing one, two, three, four and eight server models, both series and routing.

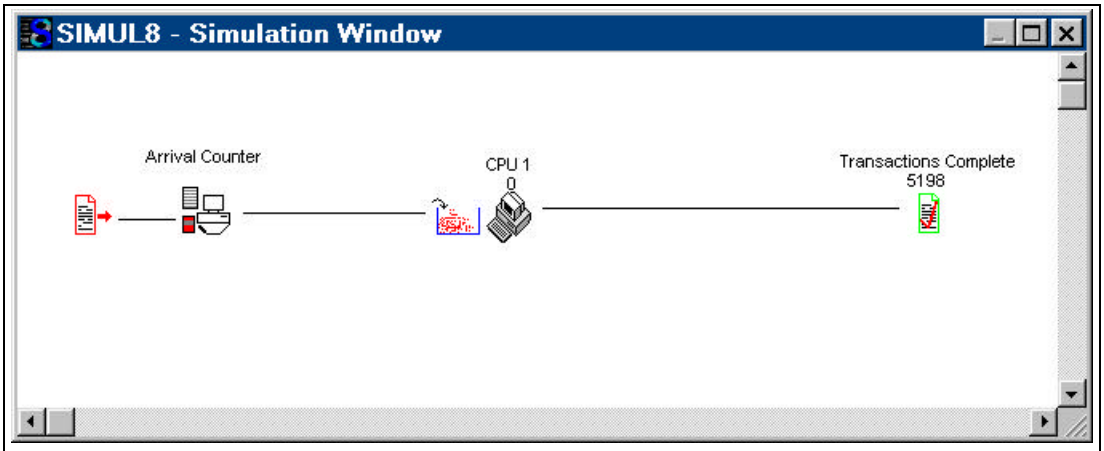

Figure 39 Simul8 Model S1SA.S8 - One Server

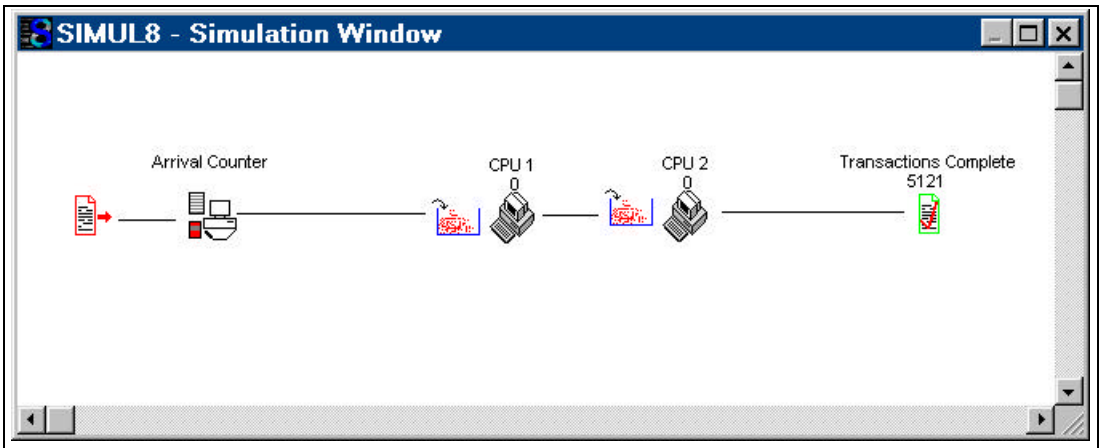

Figure 40 Simul8 Model S2SA.S8 - Two Servers in Series

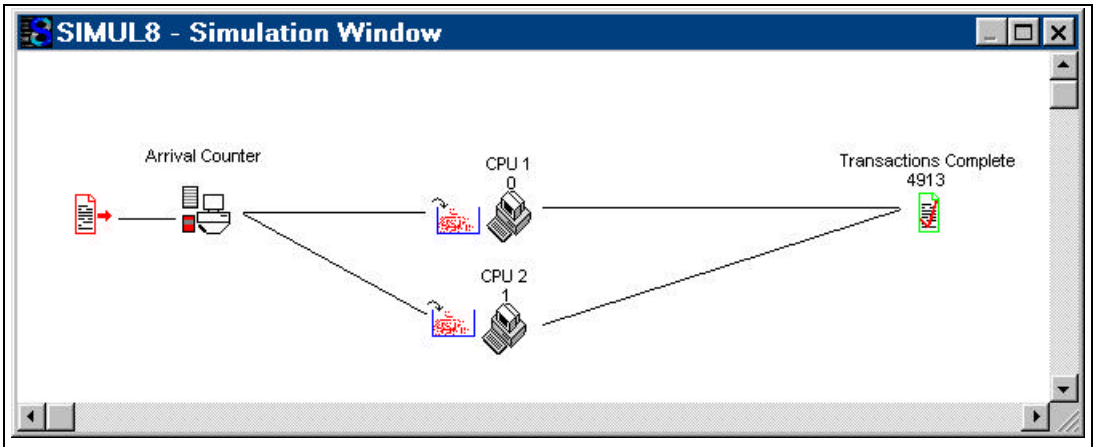

Figure 41 Simul8 Model S2PA.S8 - Two Servers Routed

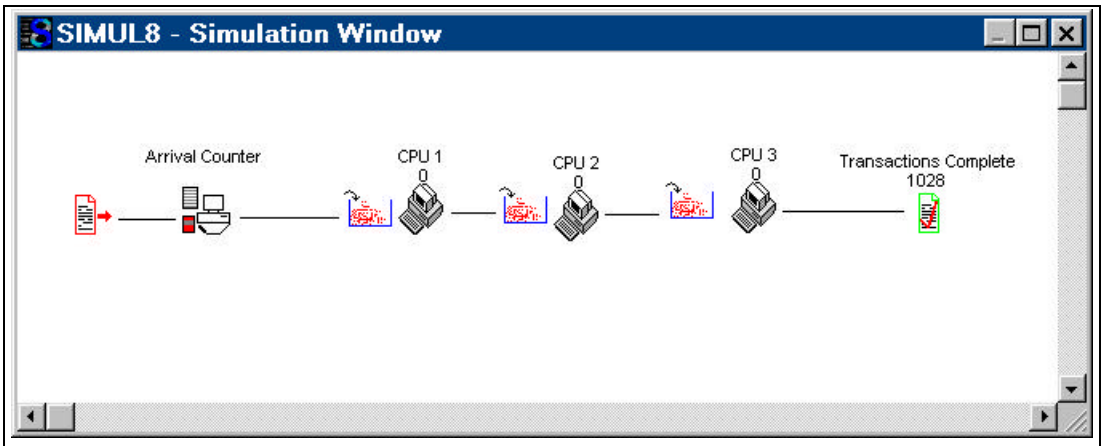

Figure 42 Simul8 Model S3SA.S8 - Three Servers in Series

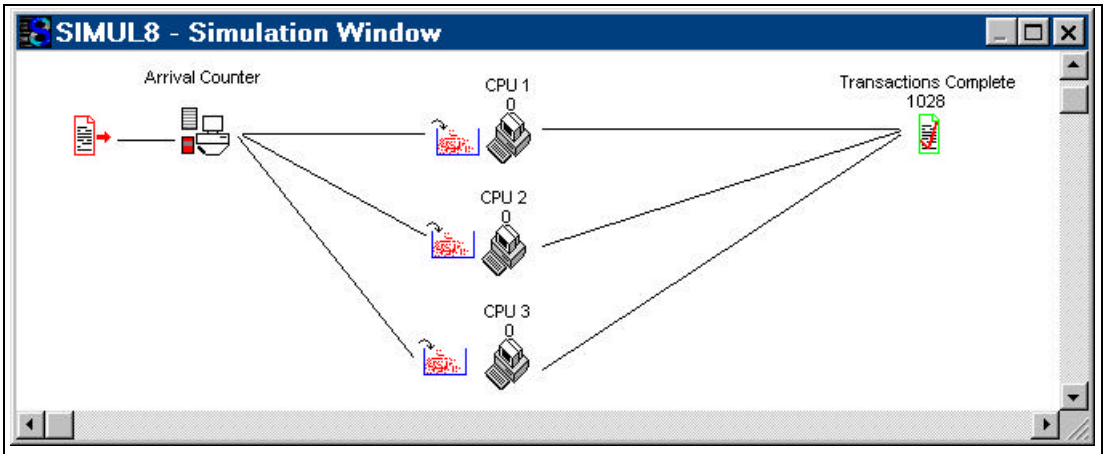

Figure 43 Simul8 Model S3PA.S8 - Three Servers Routed

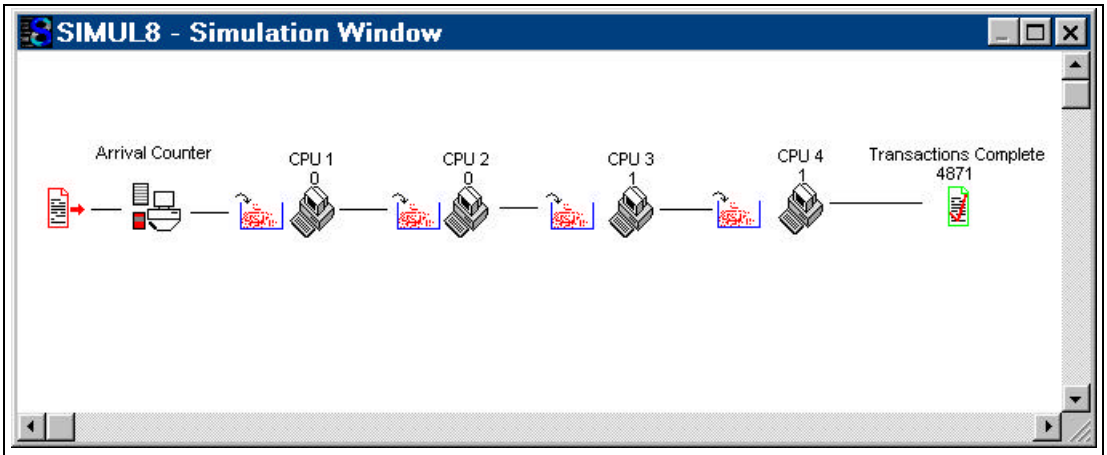

Figure 44 Simul8 Model S4SA.S8 - Four Servers in Series

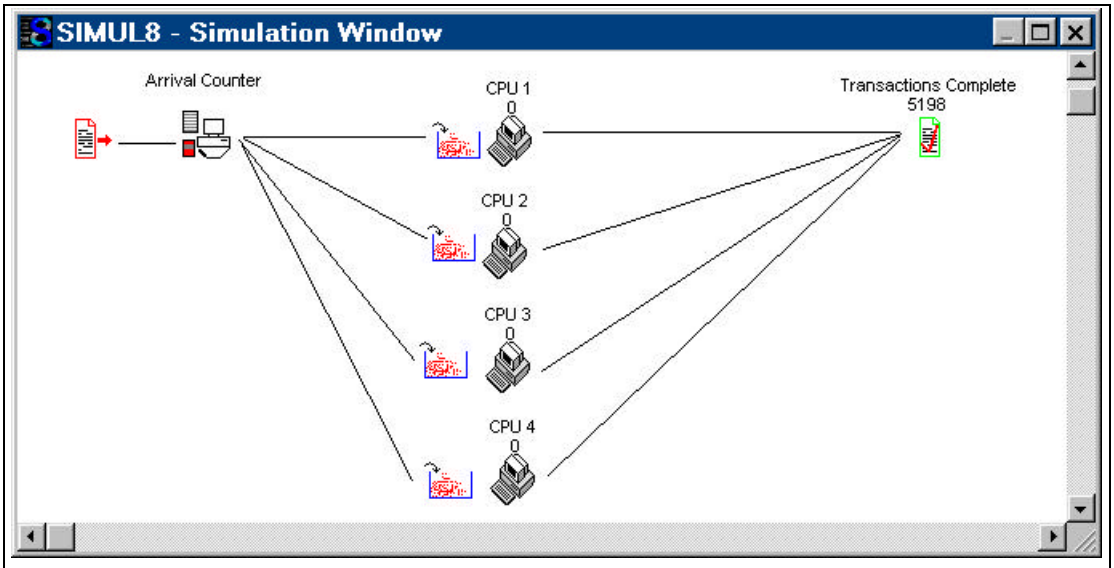

Figure 45 - Simul8 Model S4PA.S8 - Four Servers Routed

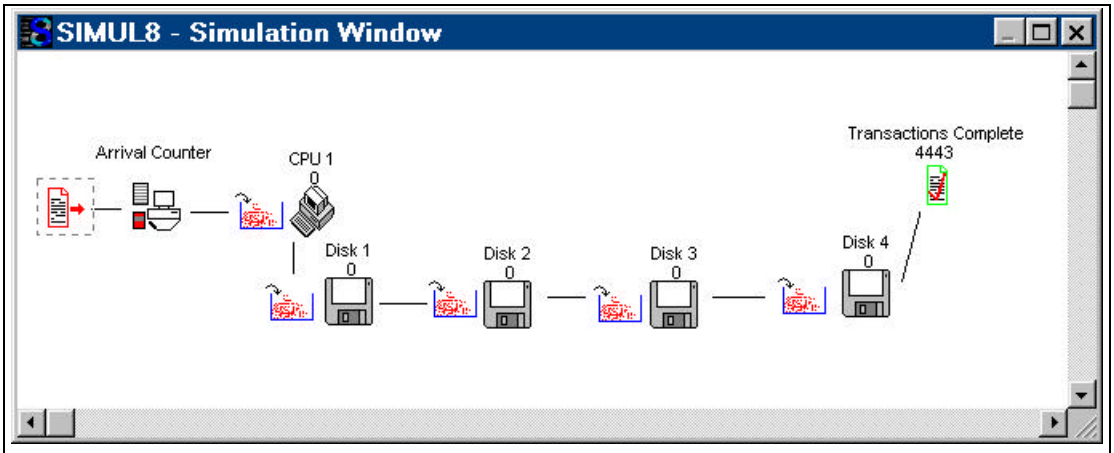

Figure 46 Simul8 Model MDxxa.S8 - Multi-device Server

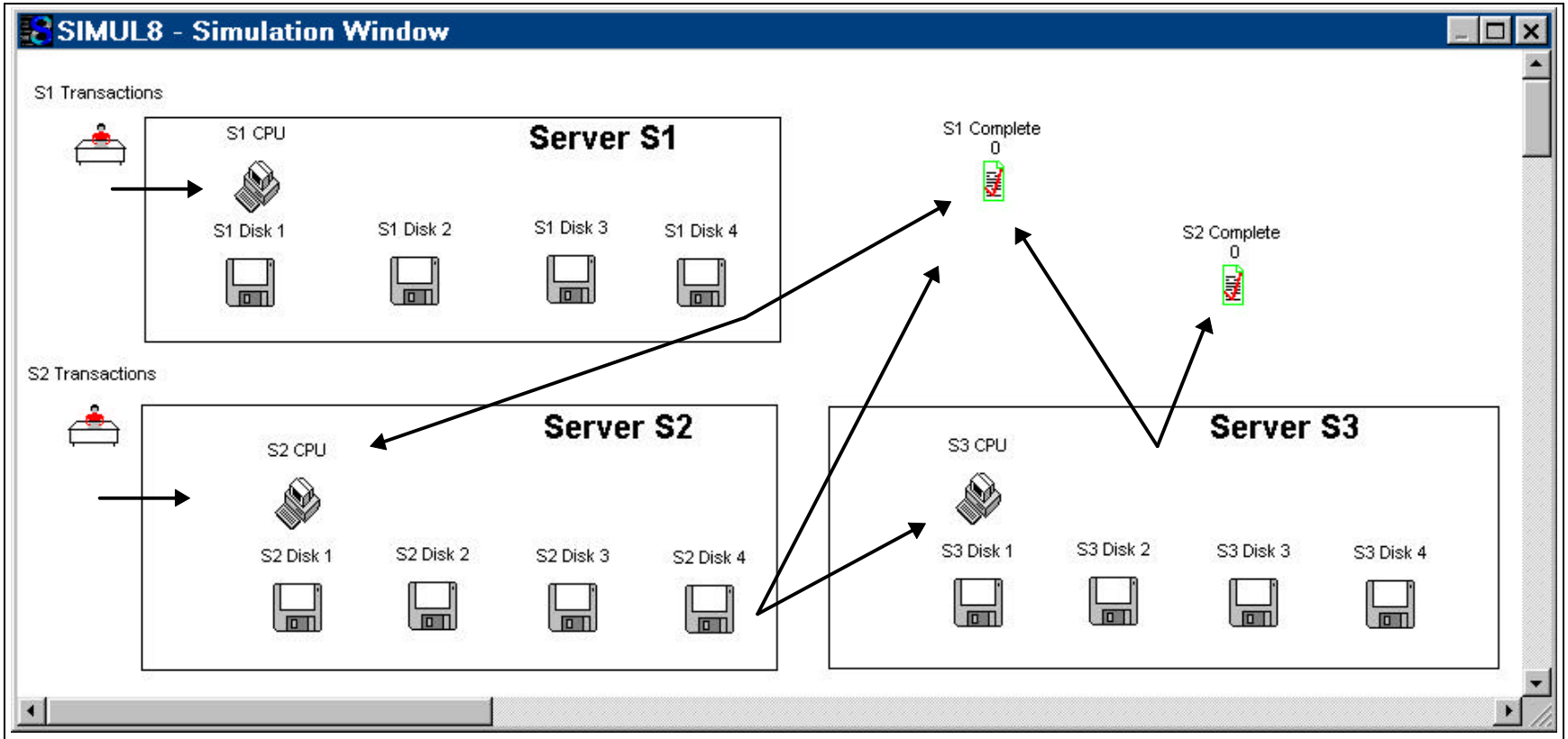

Figure 47 Three Server Simulation Model

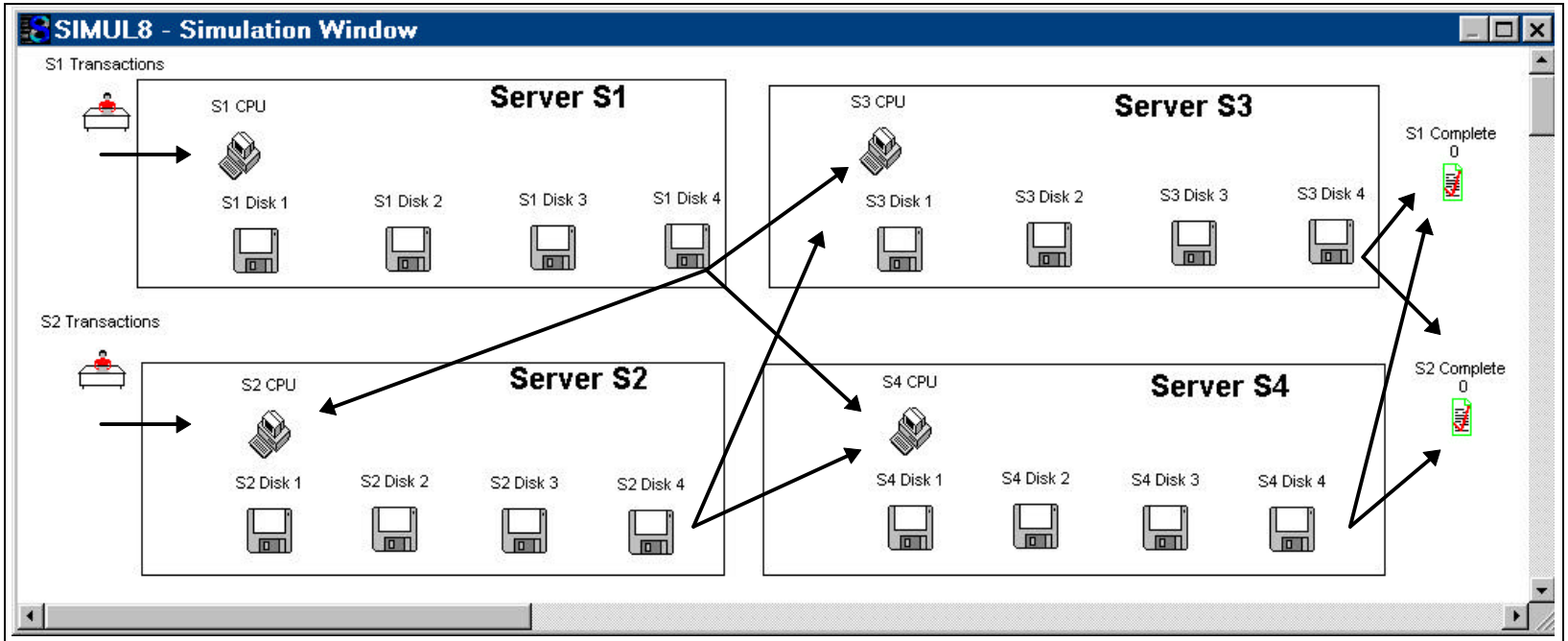

Figure 48 Four Server Simulation Model

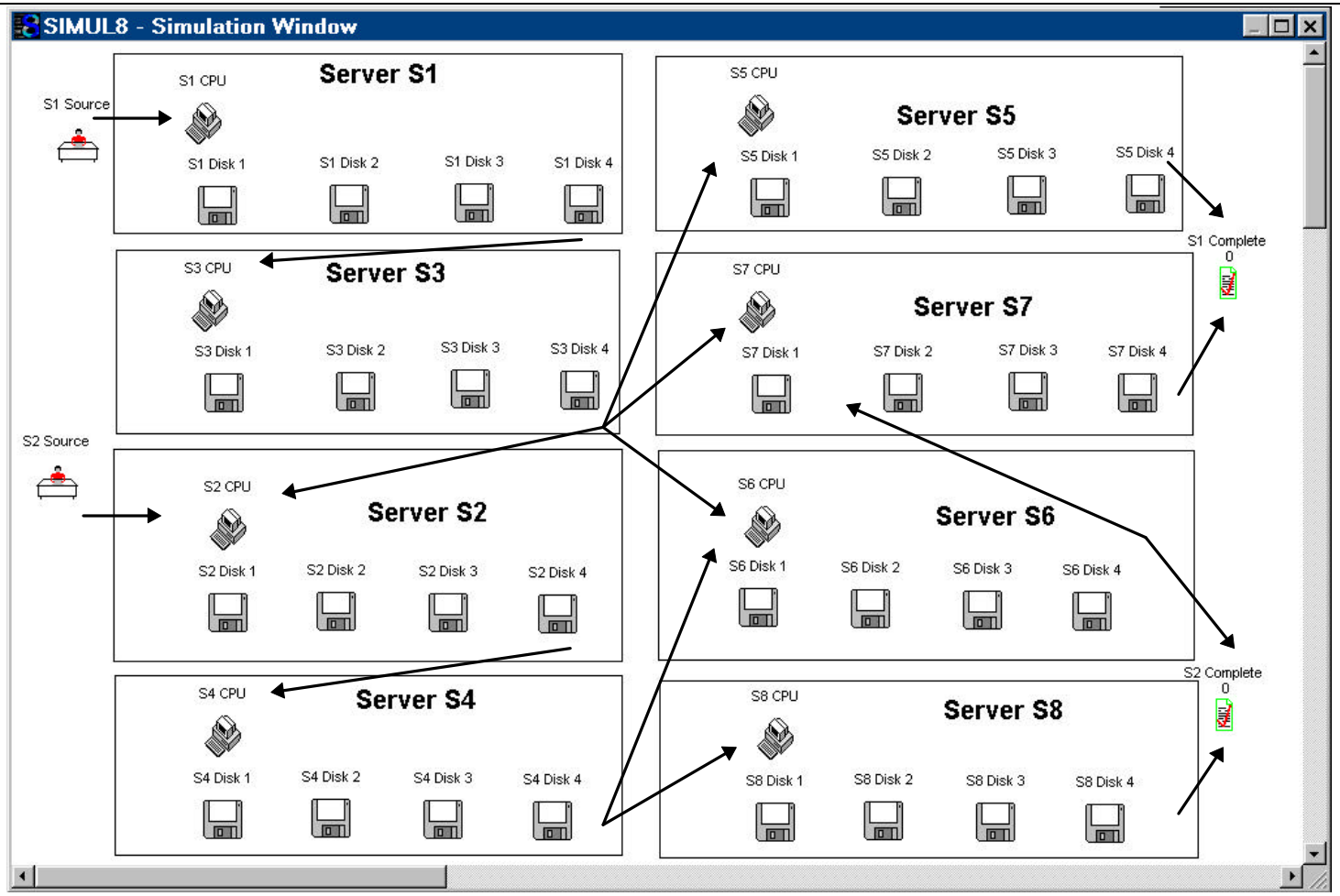

Figure 49 Eight Server Simulation Model

7.2 Appendix B: Queuing Theory Models

The queuing theory tool OpenQN was used to create the performance characteristics of the servers. This analytic modeling tool was specifically designed to model computer systems and included with Dr. Menascé's book (Menascé, Almeida, and Dowdy 1994). It is easy to use and generates results quickly. It is a simple Pascal program that reads an input file of workload parameters and produces a report.

Figure 50 and *Figure 51* show examples of the input file and output report for a single server with a single device. These are examples of the OpenQN models used to generate the results to compare to the MathCAD queuing theory results.

```
  1  1       devices(CPU) workloads(W1)
  0.5   Vector_N
  0 0   Device 1 type (LI): CPU
>>>> Service Demand Matrix
  0.1
```

Figure 50 OpenQN Input Example - Single Device Server

```
OpenQN - (c) Copr.  1994 D.  Menasce', V.  Almeida, and L.  Dowdy.
             All Rights Reserved.
This program comes with the book 'Capacity Planning and
Performance Modeling: from mainframes to client-server systems'
by Menasce, Almeida, and Dowdy, published by Prentice Hall.

>>>> Class  1 Throughput:   0.500000

>>>> Utilization of Device  1 :      5.000 %


Class  1 metrics:


>>>> Device Residence Times:

     Device  1 :   0.105263

>>>> Class  1 Response Time........:   0.105263
>>>> Class  1 Avg.  Number in System:   0.052632

>>>> Press Enter
```

Figure 51 OpenQN Output Example - Single Device Server

Figure 52 and *Figure 53* show examples of the input file and output report for a system (multi-device server) server with a CPU and four disk devices. These are examples of the OpenQN models used to generate the response time results to be implemented in the Simalytic Function.

```
  5  1       devices(CPU,D1,D2,D3,D4) workloads(W1)
  0.5   Vector_N
  0 0   Device 1 type (LI): CPU
  0 0   Device 2 type (LI): Disk1
  0 0   Device 3 type (LI): Disk2
  0 0   Device 4 type (LI): Disk3
  0 0   Device 5 type (LI): Disk4
>>>> Service Demand Matrix
  0.0050
  0.0200
  0.0250
  0.0200
  0.0300
```

Figure 52 OpenQN Input Example - Multiple Device Server

```
OpenQN - (c) Copr.  1994 D.  Menasce', V.  Almeida, and L.  Dowdy.
           All Rights Reserved.
This program comes with the book 'Capacity Planning and
Performance Modeling: from mainframes to client-server systems'
by Menasce, Almeida, and Dowdy, published by Prentice Hall.
>>>> Class  1 Throughput:   0.500000

>>>> Utilization of Device  1 :      0.250 %
>>>> Utilization of Device  2 :      7.500 %
>>>> Utilization of Device  3 :      8.750 %
>>>> Utilization of Device  4 :      4.000 %
>>>> Utilization of Device  5 :      4.500 %


Class  1 metrics:


>>>> Device Residence Times:

     Device  1 :   0.005013
     Device  2 :   0.162162
     Device  3 :   0.191781
     Device  4 :   0.083333
     Device  5 :   0.094241

>>>> Class  1 Response Time........:   0.536530
>>>> Class  1 Avg.  Number in System:   0.268265

>>>> Press Enter
```

Figure 53 OpenQN Output Example - Multiple Device Server

Table 9 shows the four server profiles used and lists the service times for each that were implemented with the OpenQN queuing theory models. The service times represent reasonable device times that are seen in real world systems. The total system service time for each server adds up to the server service time used in the single server models discussed in section *3.5.3 Validation of the Mathematical Foundation* on page 86. This allows the single results to be compared to the multi-device system results to illustrate the fallacy of using a single server node to model a complex system. The server names reflect the overall service time for each sever: **S01** has a service time of 0.1 seconds, **S05** has a service time of 0.5 seconds, **S10** has a service time of 1.0 seconds, and **S15** has a service time of 1.5 seconds.

	OpenQN Server Profiles											
OpenQN File	**S01**			**S05**			**S10**			**S15**		
	MIPS or I/O Time	Qty	Service Time	MIPS or I/O Time	Qty	Service Time	MIPS or I/O Time	Qty	Service Time	MIPS or I/O Time	Qty	Service Time
CPU (Instructions)	10	50000	0.0050	10	50000	0.0050	10	100000	0.0100	10	200000	0.0200
Disk 1 (I/O's)	0.010	2	0.0200	0.010	15	0.1500	0.010	25	0.2500	0.010	31	0.3100
Disk 2 (I/O's)	0.025	1	0.0250	0.025	7	0.1750	0.025	13	0.3250	0.025	20	0.5000
Disk 3 (I/O's)	0.020	1	0.0200	0.020	4	0.0800	0.020	14	0.2800	0.020	17	0.3400
Disk 4 (I/O's)	0.015	2	0.0300	0.015	6	0.0900	0.015	9	0.1350	0.015	22	0.3300
Total Service time			**0.1000**			**0.5000**			**1.0000**			**1.5000**

Table 9 OpenQN Device Service Times

Table 10 OpenQN Response Times on page 167 shows the OpenQN queuing theory model response times for the four server profiles in *Table 9*. Notice that these models were not run for each server for all of the arrival rates because there was no significant change in the response times.

Arrival Rate	S01	S05	S10	S15
00.01	0.100	0.501	1.003	1.506
00.05			1.013	1.529
00.08			1.022	
00.10	0.100	0.507	1.027	1.560
00.20				1.624
00.25			1.071	
00.30				1.695
00.40				1.773
00.50			1.154	1.859
00.60				1.954
00.70				2.061
00.75			1.252	2.119
00.80				2.181
00.90				2.319
01.00	0.102	0.579	1.370	2.477
01.10				2.663
01.20				2.885
01.25			1.514	
01.30				3.156
01.40				3.497
01.50			1.696	3.947
01.60				4.581
01.70				5.567
01.75			1.934	
01.80				7.411
01.90				12.620
01.95				22.739
01.98				52.816
01.99				102.843
02.00		0.694	2.260	
02.10			2.428	
02.20			2.627	
02.30			2.868	
02.40			3.166	
02.50			3.547	
02.60			4.059	

Arrival Rate	S01	S05	S10	S15
02.70			4.793	
02.80			5.968	
02.90			8.283	
03.00		0.875	15.987	
03.05			40.353	
03.10				
04.00		1.222		
05.00		2.302		
05.10		2.573		
05.20		2.938		
05.30		3.462		
05.40		4.292		
05.50		5.850		
05.60		10.019		
05.70		71.371		
05.75				
10.00	0.131			
15.00	0.157			
20.00	0.197			
25.00	0.272			
30.00	0.506			
31.00	0.651			
31.50	0.777			
31.75	0.868			
32.00	0.992			
32.25	1.171			
32.50	1.454			
32.75	1.974			
33.00	3.266			
33.10	4.555			
33.20	7.772			
33.25	12.274			
33.30	30.275			

Table 10 OpenQN Response Times

7.3 Appendix C: Simalytic Models

The Simalytic Models appear very similar to the single server simulation models. *Figure 54* shows the Simalytic Model for the Order Entry/Shipping example in section *4.2.5 Implementation Example* on page 115. The symbols used in the Simalytic Models are the same as those used for the simulation models described in section *7.1 Appendix A: Simulation Models* on page 154.

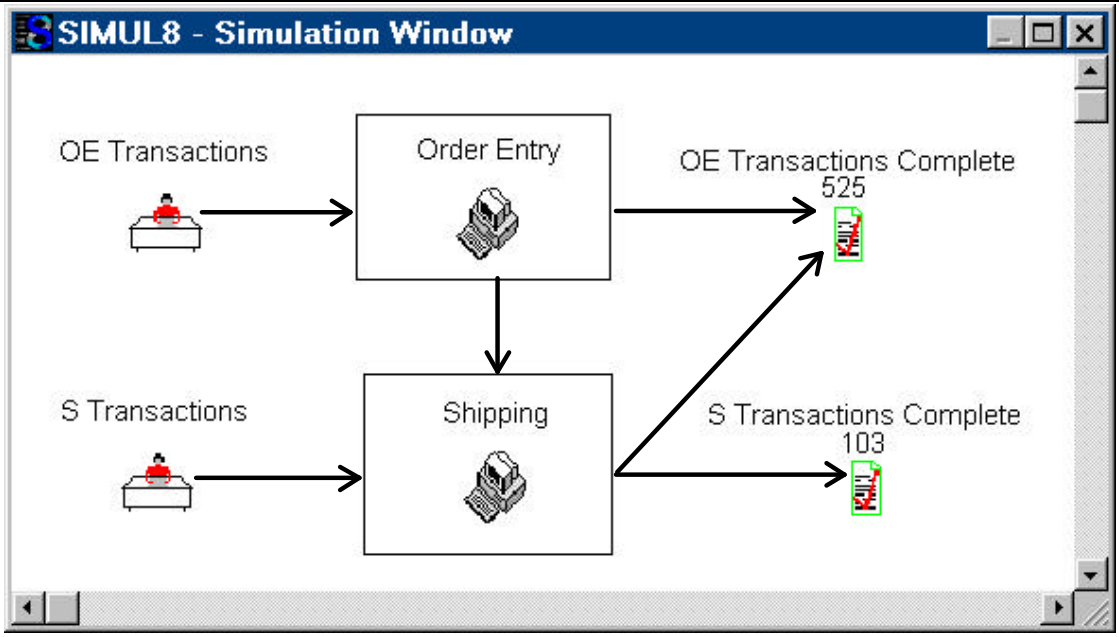

Figure 54 Simalytic Model Example - Two Servers

However, instead of each server having a static service time, the distribution name entered for the service time is actually used by the Visual Basic code to determine the service time value returned and used by the model. *Figure 55 Visual Basic Code for Simalytic Function* on pages 170 through 175 shows the Visual Basic code for the Simalytic Function in the model shown in *Figure 54*. The distribution name **DistribOE** is used by the Simalytic Function to identify transactions from the Order Entry server. The distribution

name **DistribS** is used by the Simalytic Function to identify transactions from Shipping. The distributions **DistribS01**, **DistribS05**, **DistribS10**, and **DistribS15** were used for server type S01, S05, S10 and S15 respectively.

```
Public CumIRTimeOE, CumIRTimeS As Single
Public CumIRTimeS01, CumIRTimeS05, CumIRTimeS10, CumIRTimeS15  As Single
Public CumIRTimeS01a, CumIRTimeS05a, CumIRTimeS10a, CumIRTimeS15a  As Single
Public OldTimeOE, OldTimeS As Single
Public OldTimeS01, OldTimeS05, OldTimeS10, OldTimeS15  As Single
Public OldTimeS01a, OldTimeS05a, OldTimeS10a, OldTimeS15a  As Single
Public NumOE, NumS As Single
Public NumS01, NumS05, NumS10, NumS15  As Single
Public NumS01a, NumS05a, NumS10a, NumS15a  As Single

Private Sub Form_LinkExecute(CmdStr As String, Cancel As Integer)

 If CmdStr = "RESET:" Then
   CumIRTimeOE = 0
   CumIRTimeS = 0
   CumIRTimeS01 = 0
   CumIRTimeS05 = 0
   CumIRTimeS10 = 0
   CumIRTimeS15 = 0
   CumIRTimeS01a = 0
   CumIRTimeS05a = 0
   CumIRTimeS10a = 0
   CumIRTimeS15a = 0

   OldTimeOE = 0
   OldTimeS = 0
   OldTimeS01 = 0
   OldTimeS05 = 0
   OldTimeS10 = 0
   OldTimeS15 = 0
   OldTimeS01a = 0
   OldTimeS05a = 0
   OldTimeS10a = 0
   OldTimeS15a = 0

   NumOE = 0
   NumS = 0
   NumS01 = 0
   NumS05 = 0
   NumS10 = 0
   NumS15 = 0
   NumS01a = 0
   NumS05a = 0
   NumS10a = 0
   NumS15a = 0
   End If

 If CmdStr = "WARM:" Then
   If NumOE > 0 Then AvgIRTimeOE = CumIRTimeOE / NumOE
   If NumS > 0 Then AvgIRTimeS = CumIRTimeS / NumS
   If NumS01 > 0 Then AvgIRTimeS01 = CumIRTimeS01 / NumS01
   If NumS05 > 0 Then AvgIRTimeS05 = CumIRTimeS05 / NumS05
   If NumS10 > 0 Then AvgIRTimeS10 = CumIRTimeS10 / NumS10
   If NumS15 > 0 Then AvgIRTimeS15 = CumIRTimeS15 / NumS15
   If NumS01a > 0 Then AvgIRTimeS01a = CumIRTimeS01a / NumS01a
   If NumS05a > 0 Then AvgIRTimeS05a = CumIRTimeS05a / NumS05a
   If NumS10a > 0 Then AvgIRTimeS10a = CumIRTimeS10a / NumS10a
   If NumS15a > 0 Then AvgIRTimeS15a = CumIRTimeS15a / NumS15a
   End If

  If CmdStr = "ENDRUN:" Then
   If NumOE > 0 Then AvgIRTimeOE = CumIRTimeOE / NumOE
   If NumS > 0 Then AvgIRTimeS = CumIRTimeS / NumS
   If NumS01 > 0 Then AvgIRTimeS01 = CumIRTimeS01 / NumS01
   If NumS05 > 0 Then AvgIRTimeS05 = CumIRTimeS05 / NumS05
   If NumS10 > 0 Then AvgIRTimeS10 = CumIRTimeS10 / NumS10
   If NumS15 > 0 Then AvgIRTimeS15 = CumIRTimeS15 / NumS15
   If NumS01a > 0 Then AvgIRTimeS01a = CumIRTimeS01a / NumS01a
   If NumS05a > 0 Then AvgIRTimeS05a = CumIRTimeS05a / NumS05a
   If NumS10a > 0 Then AvgIRTimeS10a = CumIRTimeS10a / NumS10a
   If NumS15a > 0 Then AvgIRTimeS15a = CumIRTimeS15a / NumS15a
   End If
```

Figure 55 Visual Basic Code for Simalytic Function

```
'DistribOE
If CmdStr = "DISTRIB: DistribOE" Then
  NewTimeOE = SimTime()
  IRTimeOE = NewTimeOE - OldTimeOE
  CumIRTimeOE = CumIRTimeOE + IRTimeOE
  NumOE = NumOE + 1
  AvgIRTimeOE = CumIRTimeOE / NumOE
     If AvgIRTimeOE <= 0.061 Then
        SRTimeOE = 6.06
     ElseIf AvgIRTimeOE <= 0.062 Then: SRTimeOE = 2.46
     ElseIf AvgIRTimeOE <= 0.063 Then: SRTimeOE = 1.15
     ElseIf AvgIRTimeOE <= 0.065 Then: SRTimeOE = 0.92
     ElseIf AvgIRTimeOE <= 0.066 Then: SRTimeOE = 0.76
     ElseIf AvgIRTimeOE <= 0.067 Then: SRTimeOE = 0.66
     ElseIf AvgIRTimeOE <= 0.071 Then: SRTimeOE = 0.43
     ElseIf AvgIRTimeOE <= 0.077 Then: SRTimeOE = 0.33
     ElseIf AvgIRTimeOE <= 0.083 Then: SRTimeOE = 0.27
     ElseIf AvgIRTimeOE <= 0.09 Then: SRTimeOE = 0.23
     ElseIf AvgIRTimeOE <= 0.1 Then: SRTimeOE = 0.2
     ElseIf AvgIRTimeOE <= 0.11 Then: SRTimeOE = 0.18
     ElseIf AvgIRTimeOE <= 0.13 Then: SRTimeOE = 0.16
     ElseIf AvgIRTimeOE <= 0.14 Then: SRTimeOE = 0.15
     ElseIf AvgIRTimeOE <= 0.17 Then: SRTimeOE = 0.14
     ElseIf AvgIRTimeOE <= 0.2 Then: SRTimeOE = 0.13
     ElseIf AvgIRTimeOE <= 0.25 Then: SRTimeOE = 0.12
     ElseIf AvgIRTimeOE <= 0.33 Then: SRTimeOE = 0.12
     ElseIf AvgIRTimeOE <= 0.5 Then: SRTimeOE = 0.11
     Else: SRTimeOE = 0.1
     End If
     SetDistribution "DistribOE", Val(SRTimeOE), 0, 0, 0, "FIXED"
  OldTimeOE = NewTimeOE
  End If

'DistribS
If CmdStr = "DISTRIB: DistribS" Then
  NewTimeS = SimTime()
  IRTimeS = NewTimeS - OldTimeS
  CumIRTimeS = CumIRTimeS + IRTimeS
  NumS = NumS + 1
  AvgIRTimeS = CumIRTimeS / NumS
     If AvgIRTimeS <= 0.71 Then
        SRTimeS = 38.21
     ElseIf AvgIRTimeS <= 0.74 Then: SRTimeS = 15.76
     ElseIf AvgIRTimeS <= 0.77 Then: SRTimeS = 10.65
     ElseIf AvgIRTimeS <= 0.8 Then: SRTimeS = 8.33
     ElseIf AvgIRTimeS <= 0.83 Then: SRTimeS = 6.98
     ElseIf AvgIRTimeS <= 0.87 Then: SRTimeS = 6.08
     ElseIf AvgIRTimeS <= 0.91 Then: SRTimeS = 5.43
     ElseIf AvgIRTimeS <= 0.95 Then: SRTimeS = 4.93
     ElseIf AvgIRTimeS <= 1 Then: SRTimeS = 4.54
     ElseIf AvgIRTimeS <= 1.33 Then: SRTimeS = 3.34
     ElseIf AvgIRTimeS <= 2 Then: SRTimeS = 2.7
     ElseIf AvgIRTimeS <= 4 Then: SRTimeS = 2.29
     ElseIf AvgIRTimeS <= 10 Then: SRTimeS = 2.11
     ElseIf AvgIRTimeS <= 100 Then: SRTimeS = 2.01
     Else: SRTimeS = 2
     End If
     SetDistribution "DistribS", Val(SRTimeS), 0, 0, 0, "FIXED"
  OldTimeS = NewTimeS
  End If
```

Figure 55 continued from previous page

```
'DistribS01
If CmdStr = "DISTRIB: DistribS01" Then
  NewTimeS01 = SimTime()
  IRTimeS01 = NewTimeS01 - OldTimeS01
  CumIRTimeS01 = CumIRTimeS01 + IRTimeS01
  NumS01 = NumS01 + 1
  AvgIRTimeS01 = CumIRTimeS01 / NumS01
     If AvgIRTimeS01 <= 0.03003 Then
        SRTimeS01 = 30.275
     ElseIf AvgIRTimeS01 <= 0.03008 Then: SRTimeS01 = 12.2735
     ElseIf AvgIRTimeS01 <= 0.03012 Then: SRTimeS01 = 7.7721
     ElseIf AvgIRTimeS01 <= 0.03021 Then: SRTimeS01 = 4.555
     ElseIf AvgIRTimeS01 <= 0.0303 Then: SRTimeS01 = 3.2665
     ElseIf AvgIRTimeS01 <= 0.03053 Then: SRTimeS01 = 1.9741
     ElseIf AvgIRTimeS01 <= 0.03077 Then: SRTimeS01 = 1.4536
     ElseIf AvgIRTimeS01 <= 0.03101 Then: SRTimeS01 = 1.1707
     ElseIf AvgIRTimeS01 <= 0.03125 Then: SRTimeS01 = 0.9921
     ElseIf AvgIRTimeS01 <= 0.0315 Then: SRTimeS01 = 0.8683
     ElseIf AvgIRTimeS01 <= 0.03175 Then: SRTimeS01 = 0.7771
     ElseIf AvgIRTimeS01 <= 0.03226 Then: SRTimeS01 = 0.6509
     ElseIf AvgIRTimeS01 <= 0.03333 Then: SRTimeS01 = 0.5059
     ElseIf AvgIRTimeS01 <= 0.04 Then: SRTimeS01 = 0.2724
     ElseIf AvgIRTimeS01 <= 0.05 Then: SRTimeS01 = 0.1972
     ElseIf AvgIRTimeS01 <= 0.06667 Then: SRTimeS01 = 0.1571
     ElseIf AvgIRTimeS01 <= 0.1 Then: SRTimeS01 = 0.1315
     ElseIf AvgIRTimeS01 <= 1# Then: SRTimeS01 = 0.1024
     ElseIf AvgIRTimeS01 <= 10# Then: SRTimeS01 = 0.1002
     ElseIf AvgIRTimeS01 <= 100# Then: SRTimeS01 = 0.1
     Else: SRTimeS01 = 0.1
     End If
     SetDistribution "DistribS01", Val(SRTimeS01), 0, 0, 0, "FIXED"
  OldTimeS01 = NewTimeS01
  End If

'DistribS01a (same as DistribS01 but allows for second server with same response.)
If CmdStr = "DISTRIB: DistribS01a" Then
  NewTimeS01a = SimTime()
  IRTimeS01a = NewTimeS01a - OldTimeS01a
  CumIRTimeS01a = CumIRTimeS01a + IRTimeS01a
  NumS01a = NumS01a + 1
  AvgIRTimeS01a = CumIRTimeS01a / NumS01a
     If AvgIRTimeS01a <= 0.03003 Then
        SRTimeS01a = 30.275
     ElseIf AvgIRTimeS01a <= 0.03008 Then: SRTimeS01a = 12.2735
     ElseIf AvgIRTimeS01a <= 0.03012 Then: SRTimeS01a = 7.7721
     ElseIf AvgIRTimeS01a <= 0.03021 Then: SRTimeS01a = 4.555
     ElseIf AvgIRTimeS01a <= 0.0303 Then: SRTimeS01a = 3.2665
     ElseIf AvgIRTimeS01a <= 0.03053 Then: SRTimeS01a = 1.9741
     ElseIf AvgIRTimeS01a <= 0.03077 Then: SRTimeS01a = 1.4536
     ElseIf AvgIRTimeS01a <= 0.03101 Then: SRTimeS01a = 1.1707
     ElseIf AvgIRTimeS01a <= 0.03125 Then: SRTimeS01a = 0.9921
     ElseIf AvgIRTimeS01a <= 0.0315 Then: SRTimeS01a = 0.8683
     ElseIf AvgIRTimeS01a <= 0.03175 Then: SRTimeS01a = 0.7771
     ElseIf AvgIRTimeS01a <= 0.03226 Then: SRTimeS01a = 0.6509
     ElseIf AvgIRTimeS01a <= 0.03333 Then: SRTimeS01a = 0.5059
     ElseIf AvgIRTimeS01a <= 0.04 Then: SRTimeS01a = 0.2724
     ElseIf AvgIRTimeS01a <= 0.05 Then: SRTimeS01a = 0.1972
     ElseIf AvgIRTimeS01a <= 0.06667 Then: SRTimeS01a = 0.1571
     ElseIf AvgIRTimeS01a <= 0.1 Then: SRTimeS01a = 0.1315
     ElseIf AvgIRTimeS01a <= 1# Then: SRTimeS01a = 0.1024
     ElseIf AvgIRTimeS01a <= 10# Then: SRTimeS01a = 0.1002
     ElseIf AvgIRTimeS01a <= 100# Then: SRTimeS01a = 0.1
     Else: SRTimeS01a = 0.1
     End If
     SetDistribution "DistribS01a", Val(SRTimeS01a), 0, 0, 0, "FIXED"
  OldTimeS01a = NewTimeS01a
  End If
```

Figure 55 continued from previous page

```
'DistribS05
If CmdStr = "DISTRIB: DistribS05" Then
  NewTimeS05 = SimTime()
  IRTimeS05 = NewTimeS05 - OldTimeS05
  CumIRTimeS05 = CumIRTimeS05 + IRTimeS05
  NumS05 = NumS05 + 1
  AvgIRTimeS05 = CumIRTimeS05 / NumS05
     If AvgIRTimeS05 <= 0.175 Then
        SRTimeS05 = 71.371
     ElseIf AvgIRTimeS05 <= 0.179 Then: SRTimeS05 = 10.019
     ElseIf AvgIRTimeS05 <= 0.182 Then: SRTimeS05 = 5.85
     ElseIf AvgIRTimeS05 <= 0.185 Then: SRTimeS05 = 4.292
     ElseIf AvgIRTimeS05 <= 0.189 Then: SRTimeS05 = 3.462
     ElseIf AvgIRTimeS05 <= 0.192 Then: SRTimeS05 = 2.938
     ElseIf AvgIRTimeS05 <= 0.196 Then: SRTimeS05 = 2.573
     ElseIf AvgIRTimeS05 <= 0.2 Then: SRTimeS05 = 2.302
     ElseIf AvgIRTimeS05 <= 0.25 Then: SRTimeS05 = 1.222
     ElseIf AvgIRTimeS05 <= 0.333 Then: SRTimeS05 = 0.875
     ElseIf AvgIRTimeS05 <= 0.5 Then: SRTimeS05 = 0.694
     ElseIf AvgIRTimeS05 <= 1# Then: SRTimeS05 = 0.579
     ElseIf AvgIRTimeS05 <= 10# Then: SRTimeS05 = 0.507
     ElseIf AvgIRTimeS05 <= 100# Then: SRTimeS05 = 0.501
     Else: SRTimeS05 = 0.5
     End If
     SetDistribution "DistribS05", Val(SRTimeS05), 0, 0, 0, "FIXED"
  OldTimeS05 = NewTimeS05
  End If

 'DistribS05a (same as DistribS05 but allows for second server with same response.)
If CmdStr = "DISTRIB: DistribS05a" Then
  NewTimeS05a = SimTime()
  IRTimeS05a = NewTimeS05a - OldTimeS05a
  CumIRTimeS05a = CumIRTimeS05a + IRTimeS05a
  NumS05a = NumS05a + 1
  AvgIRTimeS05a = CumIRTimeS05a / NumS05a
     If AvgIRTimeS05a <= 0.175 Then
        SRTimeS05a = 71.371
     ElseIf AvgIRTimeS05a <= 0.179 Then: SRTimeS05a = 10.019
     ElseIf AvgIRTimeS05a <= 0.182 Then: SRTimeS05a = 5.85
     ElseIf AvgIRTimeS05a <= 0.185 Then: SRTimeS05a = 4.292
     ElseIf AvgIRTimeS05a <= 0.189 Then: SRTimeS05a = 3.462
     ElseIf AvgIRTimeS05a <= 0.192 Then: SRTimeS05a = 2.938
     ElseIf AvgIRTimeS05a <= 0.196 Then: SRTimeS05a = 2.573
     ElseIf AvgIRTimeS05a <= 0.2 Then: SRTimeS05a = 2.302
     ElseIf AvgIRTimeS05a <= 0.25 Then: SRTimeS05a = 1.222
     ElseIf AvgIRTimeS05a <= 0.333 Then: SRTimeS05a = 0.875
     ElseIf AvgIRTimeS05a <= 0.5 Then: SRTimeS05a = 0.694
     ElseIf AvgIRTimeS05a <= 1# Then: SRTimeS05a = 0.579
     ElseIf AvgIRTimeS05a <= 10# Then: SRTimeS05a = 0.507
     ElseIf AvgIRTimeS05a <= 100# Then: SRTimeS05a = 0.501
     Else: SRTimeS05a = 0.5
     End If
     SetDistribution "DistribS05a", Val(SRTimeS05a), 0, 0, 0, "FIXED"
  OldTimeS05a = NewTimeS05a
  End If
```

Figure 55 continued from previous page

```
'DistribS10
If CmdStr = "DISTRIB: DistribS10" Then
  NewTimeS10 = SimTime()
  IRTimeS10 = NewTimeS10 - OldTimeS10
  CumIRTimeS10 = CumIRTimeS10 + IRTimeS10
  NumS10 = NumS10 + 1
  AvgIRTimeS10 = CumIRTimeS10 / NumS10
     If AvgIRTimeS10 <= 0.328 Then
        SRTimeS10 = 40.353
     ElseIf AvgIRTimeS10 <= 0.333 Then: SRTimeS10 = 15.987
     ElseIf AvgIRTimeS10 <= 0.345 Then: SRTimeS10 = 8.283
     ElseIf AvgIRTimeS10 <= 0.357 Then: SRTimeS10 = 5.968
     ElseIf AvgIRTimeS10 <= 0.37 Then: SRTimeS10 = 4.793
     ElseIf AvgIRTimeS10 <= 0.385 Then: SRTimeS10 = 4.059
     ElseIf AvgIRTimeS10 <= 0.4 Then: SRTimeS10 = 3.547
     ElseIf AvgIRTimeS10 <= 0.417 Then: SRTimeS10 = 3.166
     ElseIf AvgIRTimeS10 <= 0.435 Then: SRTimeS10 = 2.868
     ElseIf AvgIRTimeS10 <= 0.455 Then: SRTimeS10 = 2.627
     ElseIf AvgIRTimeS10 <= 0.476 Then: SRTimeS10 = 2.428
     ElseIf AvgIRTimeS10 <= 0.5 Then: SRTimeS10 = 2.26
     ElseIf AvgIRTimeS10 <= 0.571 Then: SRTimeS10 = 1.934
     ElseIf AvgIRTimeS10 <= 0.667 Then: SRTimeS10 = 1.696
     ElseIf AvgIRTimeS10 <= 0.8 Then: SRTimeS10 = 1.514
     ElseIf AvgIRTimeS10 <= 1# Then: SRTimeS10 = 1.37
     ElseIf AvgIRTimeS10 <= 1.333 Then: SRTimeS10 = 1.252
     ElseIf AvgIRTimeS10 <= 2# Then: SRTimeS10 = 1.154
     ElseIf AvgIRTimeS10 <= 4# Then: SRTimeS10 = 1.071
     ElseIf AvgIRTimeS10 <= 10# Then: SRTimeS10 = 1.027
     ElseIf AvgIRTimeS10 <= 12.5 Then: SRTimeS10 = 1.022
     ElseIf AvgIRTimeS10 <= 20# Then: SRTimeS10 = 1.013
     ElseIf AvgIRTimeS10 <= 100# Then: SRTimeS10 = 1.003
     Else: SRTimeS10 = 1#
     End If
     SetDistribution "DistribS10", Val(SRTimeS10), 0, 0, 0, "FIXED"
  OldTimeS10 = NewTimeS10
  End If

'DistribS10a (same as DistribS10 but allows for second server with same response.)
If CmdStr = "DISTRIB: DistribS10a" Then
  NewTimeS10a = SimTime()
  IRTimeS10a = NewTimeS10a - OldTimeS10a
  CumIRTimeS10a = CumIRTimeS10a + IRTimeS10a
  NumS10a = NumS10a + 1
  AvgIRTimeS10a = CumIRTimeS10a / NumS10a
     If AvgIRTimeS10a <= 0.328 Then
        SRTimeS10a = 40.353
     ElseIf AvgIRTimeS10a <= 0.333 Then: SRTimeS10a = 15.987
     ElseIf AvgIRTimeS10a <= 0.345 Then: SRTimeS10a = 8.283
     ElseIf AvgIRTimeS10a <= 0.357 Then: SRTimeS10a = 5.968
     ElseIf AvgIRTimeS10a <= 0.37 Then: SRTimeS10a = 4.793
     ElseIf AvgIRTimeS10a <= 0.385 Then: SRTimeS10a = 4.059
     ElseIf AvgIRTimeS10a <= 0.4 Then: SRTimeS10a = 3.547
     ElseIf AvgIRTimeS10a <= 0.417 Then: SRTimeS10a = 3.166
     ElseIf AvgIRTimeS10a <= 0.435 Then: SRTimeS10a = 2.868
     ElseIf AvgIRTimeS10a <= 0.455 Then: SRTimeS10a = 2.627
     ElseIf AvgIRTimeS10a <= 0.476 Then: SRTimeS10a = 2.428
     ElseIf AvgIRTimeS10a <= 0.5 Then: SRTimeS10a = 2.26
     ElseIf AvgIRTimeS10a <= 0.571 Then: SRTimeS10a = 1.934
     ElseIf AvgIRTimeS10a <= 0.667 Then: SRTimeS10a = 1.696
     ElseIf AvgIRTimeS10a <= 0.8 Then: SRTimeS10a = 1.514
     ElseIf AvgIRTimeS10a <= 1# Then: SRTimeS10a = 1.37
     ElseIf AvgIRTimeS10a <= 1.333 Then: SRTimeS10a = 1.252
     ElseIf AvgIRTimeS10a <= 2# Then: SRTimeS10a = 1.154
     ElseIf AvgIRTimeS10a <= 4# Then: SRTimeS10a = 1.071
     ElseIf AvgIRTimeS10a <= 10# Then: SRTimeS10a = 1.027
     ElseIf AvgIRTimeS10a <= 12.5 Then: SRTimeS10a = 1.022
     ElseIf AvgIRTimeS10a <= 20# Then: SRTimeS10a = 1.013
     ElseIf AvgIRTimeS10a <= 100# Then: SRTimeS10a = 1.003
     Else: SRTimeS10a = 1#
     End If
     SetDistribution "DistribS10a", Val(SRTimeS10a), 0, 0, 0, "FIXED"
  OldTimeS10a = NewTimeS10a
  End If
```

Figure 55 continued from previous page

```
 'DistribS15
 If CmdStr = "DISTRIB: DistribS15" Then
   NewTimeS15 = SimTime()
   IRTimeS15 = NewTimeS15 - OldTimeS15
   CumIRTimeS15 = CumIRTimeS15 + IRTimeS15
   NumS15 = NumS15 + 1
   AvgIRTimeS15 = CumIRTimeS15 / NumS15
      If AvgIRTimeS15 <= 0.505 Then
         SRTimeS15 = 52.816
      ElseIf AvgIRTimeS15 <= 0.513 Then: SRTimeS15 = 22.739
      ElseIf AvgIRTimeS15 <= 0.526 Then: SRTimeS15 = 12.62
      ElseIf AvgIRTimeS15 <= 0.556 Then: SRTimeS15 = 7.411
      ElseIf AvgIRTimeS15 <= 0.588 Then: SRTimeS15 = 5.567
      ElseIf AvgIRTimeS15 <= 0.625 Then: SRTimeS15 = 4.581
      ElseIf AvgIRTimeS15 <= 0.667 Then: SRTimeS15 = 3.947
      ElseIf AvgIRTimeS15 <= 0.714 Then: SRTimeS15 = 3.497
      ElseIf AvgIRTimeS15 <= 0.769 Then: SRTimeS15 = 3.156
      ElseIf AvgIRTimeS15 <= 0.833 Then: SRTimeS15 = 2.885
      ElseIf AvgIRTimeS15 <= 0.909 Then: SRTimeS15 = 2.663
      ElseIf AvgIRTimeS15 <= 1# Then: SRTimeS15 = 2.477
      ElseIf AvgIRTimeS15 <= 1.111 Then: SRTimeS15 = 2.319
      ElseIf AvgIRTimeS15 <= 1.25 Then: SRTimeS15 = 2.181
      ElseIf AvgIRTimeS15 <= 1.333 Then: SRTimeS15 = 2.119
      ElseIf AvgIRTimeS15 <= 1.429 Then: SRTimeS15 = 2.061
      ElseIf AvgIRTimeS15 <= 1.667 Then: SRTimeS15 = 1.954
      ElseIf AvgIRTimeS15 <= 2# Then: SRTimeS15 = 1.859
      ElseIf AvgIRTimeS15 <= 2.5 Then: SRTimeS15 = 1.773
      ElseIf AvgIRTimeS15 <= 3.333 Then: SRTimeS15 = 1.695
      ElseIf AvgIRTimeS15 <= 5# Then: SRTimeS15 = 1.624
      ElseIf AvgIRTimeS15 <= 10# Then: SRTimeS15 = 1.56
      ElseIf AvgIRTimeS15 <= 20# Then: SRTimeS15 = 1.529
      ElseIf AvgIRTimeS15 <= 100# Then: SRTimeS15 = 1.506
      Else: SRTimeS15 = 1.5
      End If
      SetDistribution "DistribS15", Val(SRTimeS15), 0, 0, 0, "FIXED"
   OldTimeS15 = NewTimeS15
   End If

 'DistribS15a same as DistribS15 but allows for second server with same response.)
 If CmdStr = "DISTRIB: DistribS15a" Then
   NewTimeS15a = SimTime()
   IRTimeS15a = NewTimeS15a - OldTimeS15a
   CumIRTimeS15a = CumIRTimeS15a + IRTimeS15a
   NumS15a = NumS15a + 1
   AvgIRTimeS15a = CumIRTimeS15a / NumS15a
      If AvgIRTimeS15a <= 0.505 Then
         SRTimeS15a = 52.816
      ElseIf AvgIRTimeS15a <= 0.513 Then: SRTimeS15a = 22.739
      ElseIf AvgIRTimeS15a <= 0.526 Then: SRTimeS15a = 12.62
      ElseIf AvgIRTimeS15a <= 0.556 Then: SRTimeS15a = 7.411
      ElseIf AvgIRTimeS15a <= 0.588 Then: SRTimeS15a = 5.567
      ElseIf AvgIRTimeS15a <= 0.625 Then: SRTimeS15a = 4.581
      ElseIf AvgIRTimeS15a <= 0.667 Then: SRTimeS15a = 3.947
      ElseIf AvgIRTimeS15a <= 0.714 Then: SRTimeS15a = 3.497
      ElseIf AvgIRTimeS15a <= 0.769 Then: SRTimeS15a = 3.156
      ElseIf AvgIRTimeS15a <= 0.833 Then: SRTimeS15a = 2.885
      ElseIf AvgIRTimeS15a <= 0.909 Then: SRTimeS15a = 2.663
      ElseIf AvgIRTimeS15a <= 1# Then: SRTimeS15a = 2.477
      ElseIf AvgIRTimeS15a <= 1.111 Then: SRTimeS15a = 2.319
      ElseIf AvgIRTimeS15a <= 1.25 Then: SRTimeS15a = 2.181
      ElseIf AvgIRTimeS15a <= 1.333 Then: SRTimeS15a = 2.119
      ElseIf AvgIRTimeS15a <= 1.429 Then: SRTimeS15a = 2.061
      ElseIf AvgIRTimeS15a <= 1.667 Then: SRTimeS15a = 1.954
      ElseIf AvgIRTimeS15a <= 2# Then: SRTimeS15a = 1.859
      ElseIf AvgIRTimeS15a <= 2.5 Then: SRTimeS15a = 1.773
      ElseIf AvgIRTimeS15a <= 3.333 Then: SRTimeS15a = 1.695
      ElseIf AvgIRTimeS15a <= 5# Then: SRTimeS15a = 1.624
      ElseIf AvgIRTimeS15a <= 10# Then: SRTimeS15a = 1.56
      ElseIf AvgIRTimeS15a <= 20# Then: SRTimeS15a = 1.529
      ElseIf AvgIRTimeS15a <= 100# Then: SRTimeS15a = 1.506
      Else: SRTimeS15a = 1.5
      End If
      SetDistribution "DistribS15a", Val(SRTimeS15a), 0, 0, 0, "FIXED"
   OldTimeS15a = NewTimeS15a
   End If
S8_Signal_Done
End Sub
```

Figure 55 continued from previous page

Figure 56 Three Server Simalytic Model below, *Figure 57 Four Server Simalytic Model* below, and *Figure 58 Eight Server Simalytic Model* on page 177 show the Simalytic Models used in the examples in section *5 Investigations into Simalytic Modeling* on page 127. Each server uses the Simalytic Function in Figure 55 to implement the service time for that server.

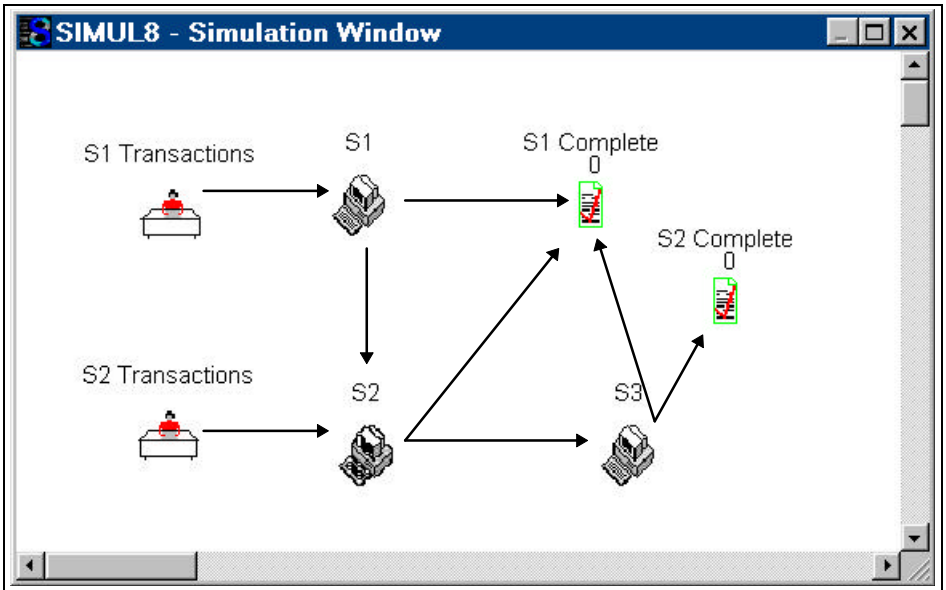

Figure 56 Three Server Simalytic Model

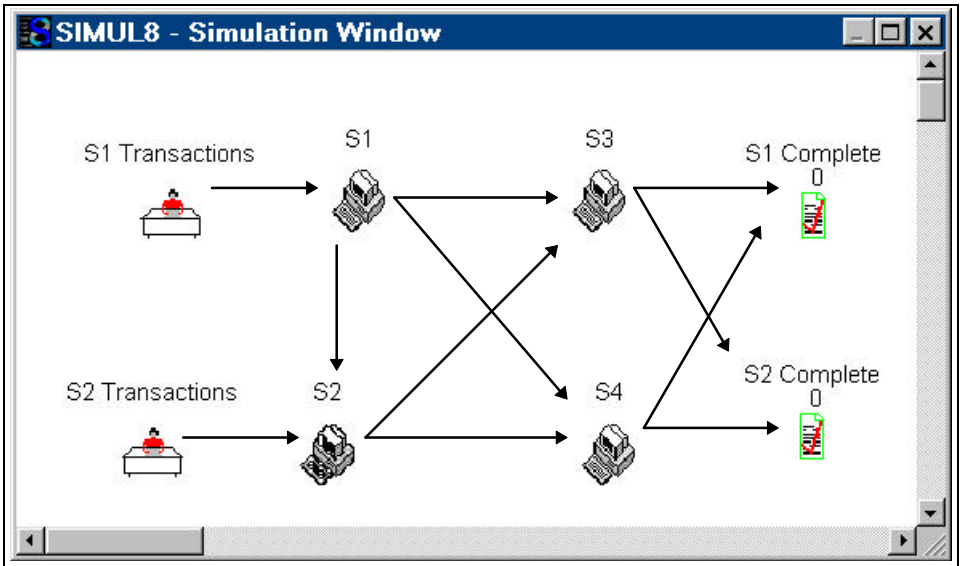

Figure 57 Four Server Simalytic Model

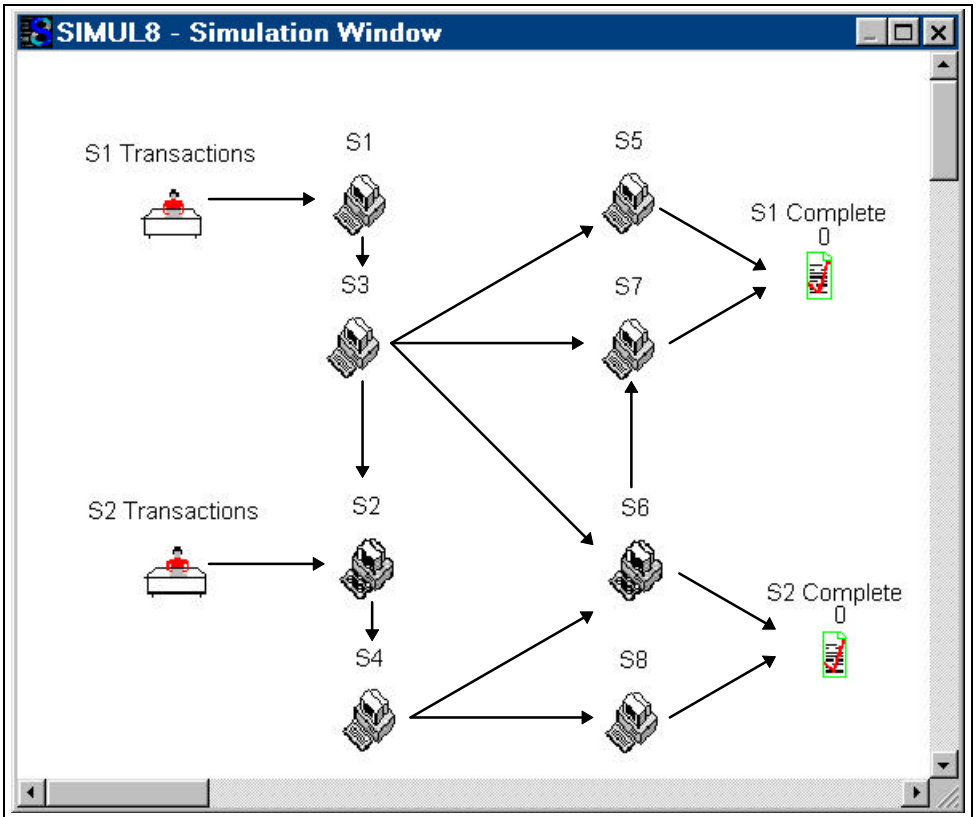

Figure 58 Eight Server Simalytic Model

7.4 Appendix D: MathCAD Queuing Formulae

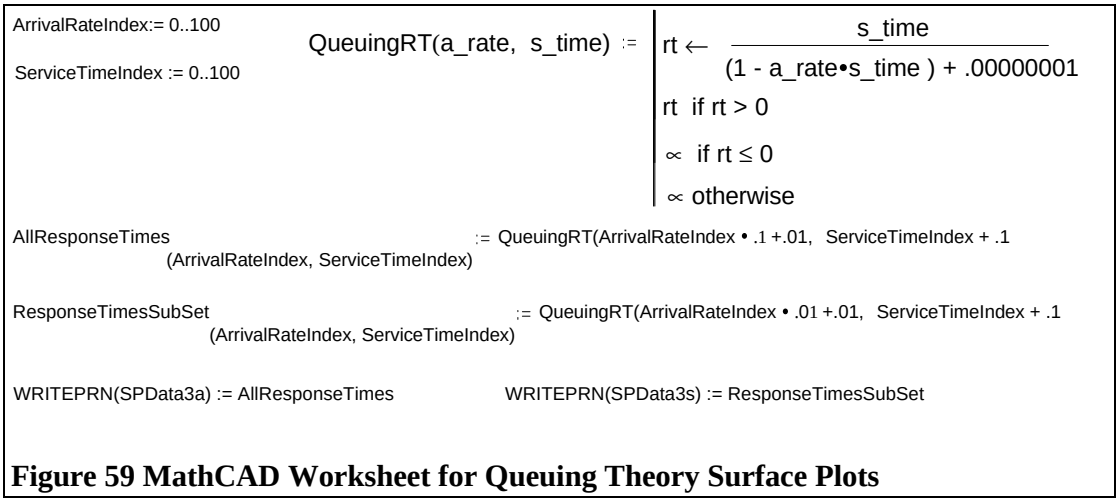

Figure 59 MathCAD Worksheet for Queuing Theory Surface Plots

Figure 59 MathCAD Worksheet for Queuing Theory Surface Plots presents the MathCAD worksheet that defines the queuing theory function used to calculate the response times for the surface plots in section *3.5.3 Validation of the Mathematical Foundation* on page 86. This worksheet also shows the MathCAD statements to use the queuing theory function to generate data for both of the surface plots in section *3.5.3*. The **WRITEPRN** statements write the array on the right side of the assignment symbol (**:=**) to the file name provided as a parameter. For readers not familiar with the MathCAD program from MathSoft (MathSoft 1995), additional information is available in the user's guide for the current version of MathCAD.

The QueuingRT function returns a response time value when called with two parameters, arrival rate and service time. The function defined in *Figure 59* was used for all MathCAD queuing theory results. Both the service time and the arrival rate are a series of values calculated from the respective indices. The function is defined to return only positive response times. The calculated time goes negative when the server is saturated and

the function returns infinity to show this. Because of MathCAD rounding of intermediate values, a small value is added to avoid a divide by zero error.