A Simulation Approach for Network Operations Performance Studies

Arbee L.P. Chen, E. Jane Cameron*, George F. Shuttleworth*, E. Carey Anderson*

Bell Communications Research, Inc.
444 Hoes Lane, Piscataway, NJ 08854
*435 South St., Morristown, NJ 07960

ABSTRACT

This paper describes a simulation approach to study the performance issues of the Telephone Network Operations Process. For the initial phase of the study, the Customer Trouble Reporting Process was selected. This process specifies interactions between many components of an operations system. A queuing network model was designed to model this process. Assumptions about the size of a maintenance center, the trouble rate, and the trouble report distribution over time of day were made to simulate existing operations. Other assumptions on the probabilities of trouble types, the processing time incurred at each module, and the work hours of a repair technician (RT) were also made to provide a set of input parameters for the simulation.

The average receipt-to-close time of a customer trouble as a function of the number of RTs and data access time was the first issue studied. From the results of the simulation, the smallest number of RTs required to meet the repair commitment time can be determined, and the effect of different data access times can be understood. The other issue studied was the workload distribution. Each module has associated queues. Work that is to be done by a module is held in a queue until it is acted upon. Bottlenecks in the system can be identified by studying the changes in the length of the queues over the simulation time.

The Bellcore-developed system *IC** was used to implement the simulation, to display queuing effects and transaction flows, and to produce a pictorial representation of the simulation model. A library of C routines was built to support the simulation. These routines generate troubles based on a table of trouble arrivals, manipulate queues, manage multiple repair technicians operations, maintain a system clock, and write data for statistical analysis. The S system, which was developed at Bell Laboratories, was used to generate graphical plots.

This work provides a feasibility demonstration of the simulation modeling approach. Future work on the performance analysis for more complex operations processes and data management strategies is addressed.

1. INTRODUCTION

The Network Operations Process encompasses a set of operations system (OS) functions directly concerned with the day-to-day operation of Bellcore Client Companies' networks. Examples include the following operations functions [1] [2] (see Figure 1):

- Installation, which involves implementing all types of network circuits, transmission paths and/or systems, and subscriber services.

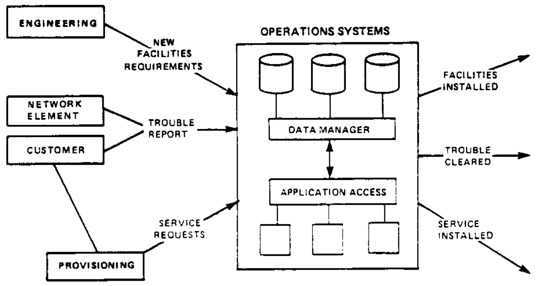

Figure 1. Network Operations Process Flows.

- Network monitoring operations, which involve continuous surveillance of the network to detect conditions requiring intervention and/or repair activity so as to improve or preserve network service levels.
- Maintenance, which involves acceptance of network trouble indications and customer trouble reports, analyzing them in order to localize underlying trouble conditions, and making repairs as needed. The major source for the network trouble indications is the network monitoring operations.

In order to systematically study the performance issues of the Network Operations Process for different operations systems architectural alternatives and data management strategies, a simulation modeling approach has been used. The viability of this simulation modeling approach was first established by modeling an existing operation process whose parameters are well known. Once this feasibility has been demonstrated, the extensions of the modeling process into more complex target environments can follow.

This paper describes the effort to establish the viability of this approach by modeling the Customer Trouble Reporting Process for POTS-like circuits. This process specifies interactions between many components of an operations system. The process flow was modeled and implemented in the Bellcore-developed system [3] IC*. The IC* system provides an environment for simulation, analysis and display of complex systems. Assumptions for the simulation were made from well established parameters of this process. They were also changed in the simulation to study system performance.

This paper is organized into five sections. Section 2 describes the Customer Trouble Reporting Process and the simulation model that represents that process. Features of the IC* system and the

CH2611-2/88/0000/0105$01.00 © 1988 IEEE

implementation of the model in IC* are discussed in Section 3. Section 4 presents the simulation results and their analysis. Finally, Section 5 concludes this work and provides future directions of this modeling effort.

2. MODEL OF A NETWORK OPERATIONS PROCESS

Figure 2 depicts the flow of the Customer Trouble Reporting Process.

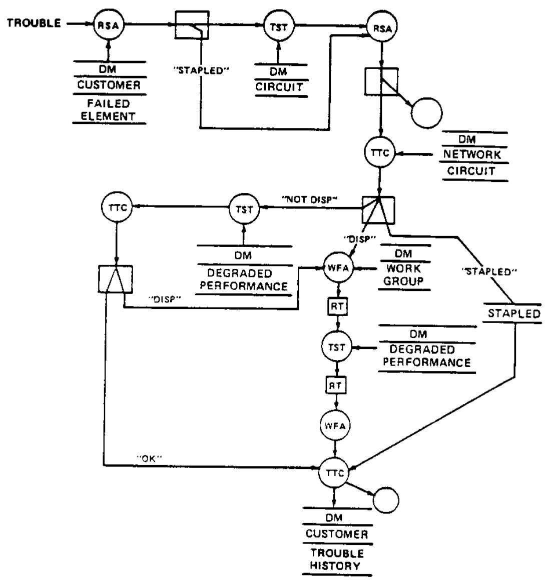

Figure 2. Customer Trouble Reporting Process Flow.

2.1 Functional Components, Data, and Queries

The process specifies interactions between the following functional components of an OS.

- the 'repair service answering' (RSA) module - responsible for supporting the trouble taking task;
- the 'testing' (TST) module - responsible for coordinating automatic/manual test requests and trouble isolation;
- the 'trouble ticket control' (TTC) module - responsible for trouble screening and trouble ticket generation, control, and tracking;
- the 'work force administration' (WFA) module - responsible for work force assignments and control;
- the 'data manager' (DM) module - responsible for managing data access, control, and cataloging.
- the 'repair technician' (RT) - responsible for restoring and testing service/circuit.

The types of data [1] required to process a trouble include:

- CUSTOMER data used to identify a customer and to describe a customer's services;
- CIRCUIT data that describes circuit layouts including test access points;
- NETWORK data used to describe the network topology;
- WORK FORCE data related to work force characteristics including *Work Group*;
- PERFORMANCE data associated with customer services and network facilities including *Failed Element* and *Degraded Performance*;
- TROUBLE HISTORY data that details history of a customer's circuit or service.

Requests for data by functional modules may involve simple or complex queries. An example of a simple query would be accessing a customer record using a circuit identifier or customer name as an access key. An example of a complex query would be determining the status of a facility element used by a customer circuit.

2.2 The Process Flow

The process flow can be described as follows.

A customer initiated trouble will enter the system through the RSA module, typically via a customer telephone call. Besides trouble description data provided by the customer, CUSTOMER and Failed Element information is retrieved by the DM. If the circuit is part of a network failure, the trouble will be statused "STAPLED" and associated with other related troubles; if not, the circuit will be tested and test information passed to RSA. Based on the test information, the customer can either choose to cancel the trouble or request that a trouble ticket be generated.

TTC generates a trouble ticket and statuses the trouble "DISP(ATCHED)" or "NOT DISP(ATCHED)." A "STAPLED" trouble is normally attached to some single trouble that has a "DISP" status.

If the trouble is dispatched, it will be assigned to the proper work group by WFA, dispatched to a repair technician (RT), repaired, post-repair tested, and closed. A "NOT DISP" trouble will require more detailed testing and trouble isolation. If this process sufficiently isolates the trouble within the network, the TTC will status the trouble "DISP" and it will be handled by the flow described above. Otherwise, the trouble will be statused "OK." As the trouble makes a last pass through TTC, CUSTOMER and TROUBLE HISTORY data will be updated and the trouble closed.

2.3 The Simulation Model

As shown in Figure 3, the simulation model has modules RSA, TTC, WFA, DM, TST, a number of RTs and channels between them for communications. There are two separate channels between two modules, one for each direction of communication. They are represented by a double-headed arrow in the figure for clarity.

Each of the modules is associated with two queues, while RTs have no queues. One of the two queues has a higher priority from which jobs will be retrieved and processed first. After a *request module* sends a job to and requests a service of a *server module* , the request module takes another job from one of its queues to process. The returning jobs from a server module will be placed in the high priority queue of the request module. To reduce the call-waiting time (defined as the time period a customer waits over the phone when reporting a trouble), jobs will also be placed in the high priority queue of any server module during this period. All trouble reports enter the system from the input queue at RSA. There are also output queues (which are not shown in the figure) associated with RSA and TTC. These queues basically serve as a storage facility to store jobs which have been processed. Processed jobs are those: that are cancelled by customers after a quick test; or that have gone through screening processes and either no troubles have been found or a repair process has been completed. A special queue

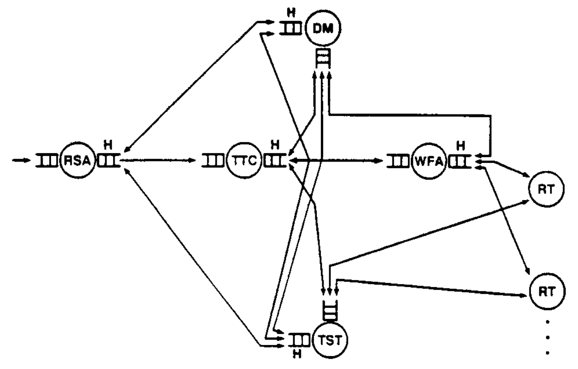

Figure 3. The Simulation Model.

(not shown in the figure) is also associated with TTC, which stores stapled jobs. When a job is completed, the stapled jobs have to be checked to see if they are related to the completed job. If they are, then these stapled jobs are also considered completed. WFA also has a special queue which is not shown in the figure and will be further described in Section 4.3.

3. IMPLEMENTATION OF THE SIMULATION MODEL

3.1 The IC* System

The IC* project [4] is an effort to create an environment for the design and development of complex systems. It is based on a new model of parallel computation that gives a precise mathematical meaning to parallel coordinated computation. The project includes the development of the C&E (Cause and Effect) programming language, a development environment, and a parallel machine. A brief description of the C&E language and the parts of the development environment used in creating this simulation follows.

3.1.1 The C&E language

A specification in C&E typically consists of several capsules and a list of connections between the state variables in the different capsules. A capsule consists of a declaration of state variables and a list of fragments. The fragments specify how the values of the state variables change in time. Each fragment consists of a cause, which is a Boolean expression in the state variables, and a list of possible effects. The semantics of the fragments are such that when the cause expression is true, then one of the effects is chosen to take place. Each effect has a probability, a time delay and a list of changes. The effect's probability is the probability that the effect will be chosen when the fragment is triggered.

The connections give static time invariant relationships between the state variables. Connections give the specifier a very flexible way to combine many small capsules into a large specification.

*3.1.2 Name-Handler -- N**

Name-Handler can be viewed as an internal "database management" system, which coordinates the access of names for the rest of the environment. The N* system manages the names of objects, variables, behaviors, invariants, etc., in a way that mirrors the structure of the system component being described. It keeps relevant information within "easy reach".

Because of the modularity of the C&E language, the simulation was created and then it was embedded in display programs which add many *filters* to transform the state variables of the simulation in meaningful ways for the I* system to display. The semantics of C&E and the capabilities of N* allow this form of program development and display.

*3.1.3 Filters -- F**

An observation filter is a C&E program whose input is a group of the simulation program's variables and whose output is a meaningful transformation of their values. For example, a filter may output the value true when an event such as a job splitting occurs. Because of the inherent parallelism of the model, observation filters can be written to monitor but not alter the normal logic of the program.

*3.1.4 Infoscope -- I**

Infoscope, consisting of Infosight and Infosound, is a display utility using both audio and visual icons to depict "events" defined by the observation filters. Infosight contains a library of visual gauges. There are a wide variety of gauges. For example, text may be displayed using alphanumeric characters, magnitude may be displayed using a rectangle that changes shape, and context may be displayed by an icon that changes color. The gauges may be placed at will on the screen, thus related gauges can be placed close together; or if values are to "flow" through several gauges, then these gauges can be placed to create this effect. Visual gauges are good for displaying information which requires close attention, and for displaying many different types of information at one time. Infosound contains a library of audio gauges and a sound editor. The library contains musical, voice and sound effects gauges. Audio gauges are good for monitoring information that requires sustained but not close attention, and for emphasizing the beginning or end of a particular event.

*3.1.5 Documentation -- D**

Documentation in the form of a layout of the C&E program (or part of the program) can be produced automatically from the C&E source code. D* produces drawings that are very close to those used to design the simulation. This shows that the source code closely models the specification of the simulation.

3.2 Implementation of the Model in the IC* System

3.2.1 IC specification*

Each component in the model was specified as a C&E capsule. A component can be a job generator, a queue, a module, an RT or a channel. A CLOCK maintaining the system time was also specified for a time reference by each component. The interaction between the components was expressed by connections between the state variables of the different capsules. For example, 'RSA.**Time** = CLOCK.**Time**' connects the state variable **Time** at RSA and CLOCK. Whenever a **Time** is referenced in RSA, its value is taken from the **Time** in CLOCK.

A job is described by a **JobId**, a **TimeIn** and a **TimeOut** for recording job times in the system, a **JobType** for distinguishing among a stapled job, a dispatched job, and a nondispatched job, a **DataAccess** for indicating a read or write data operation, and seven other attributes for server modules to identify the request module of the job. **JobType, DataAccess** and the request module identifier uniquely determine the operation to be performed at a server module.

Jobs are generated according to certain distribution and put in the system input queue. A queue is represented by a **QueueId** and a **QueueLength**. A module gets a job from one of its associated queues and processes it based on some probabilistic conditions. For example, RSA could send the job to TTC if the job can be stapled after a checking from DM; or it could send the job to TST for a quick test. Different delay factors were associated with different conditions to represent delays for data access, repair and testing. The values of probabilities and delays can be changed to reflect different operations environments.

The channels were implemented using a simple send-and-receive protocol. The whole job was sent as a message across network for inter-module communication. Two separate channels were specified between two modules, one for each direction. A delay was associated with each channel to represent the transmission delay.

A connection file connected all these capsules to form the system. RT capsule was replicated to represent multiple repair technicians. Their interactions with WFA and ITS were carefully handled by a locking mechanism to prevent possible synchronization problems. Both **TimeOuts** and **QueueLengths** are changed in time in the simulation. Statistical data can be calculated from these changes for system performance analysis as will be discussed in the following section.

3.2.2 C function library

Several groups of C functions were necessary for the simulation, they are:

- A collection of input/output routines to handle data files. In addition to the visual displays, the simulation created data files for analysis. These routines created these files as the simulation ran, and from these files created the files needed by the S system [5] to produce the graphical representation of the data gained by the simulation.
- A collection of routines for manipulating queues. These included implementations of queues as linked lists of jobs and as buffers of time-stamped jobs.
- A collection of routines for manipulating data used in the filters. These are the routines to transform data for the infoscope format, and for other types of manipulation of data to be displayed.

3.2.3 IC display*

There are two main displays, a display showing how jobs flow through the system, and a display showing the status of each module at each unit of time.

The screen for the job flow display is shown in Figure 4. The title appears at the top of the screen. The time of day is displayed just below the title. The integer, representing the job identification number of the job being traced, is shown in the lower left hand corner. Just below the job identification number is the time elapsed since the job entered the system. In the lower right hand corner, the total number of RTs is displayed just above the data access time parameter. In the center of the screen is a display of the system. The modules: RSA, DM, TTC, TST, and WFA are displayed as spots that change color: bright pink indicates that the job is at that module; blue indicates that the module is processing a job, but not the one being traced; and grey indicates that the module is not busy. The queues are represented by thermometer gauges. The background is pink if the traced job is in the queue, and grey otherwise. A blue rectangle grows and shrinks to indicate the number of jobs in the queue. The queues are placed next to the appropriate modules. The RTs are handled by a single large spot, which is pink if the traced job is being processed by a RT, blue if at least one RT is working on a job and no RT is working on the job being traced, and grey if no RT is working on any job. If the RT icon is pink, then the index of the RT processing the traced job is displayed below the RT icon.

The display emphasizing the work being done by each module at each unit of time is very similar to the job flow display. Again the title is placed at the top of the screen, with the time of day just below it. The modules: RSA, DM, TTC, TST, and WFA are depicted as spots, which are pink if they are busy and grey otherwise. However, the job identification number of the job being processed by each node is displayed also. The queues are displayed as rectangles which are grey if they are empty and red otherwise. A number indicating the size, i.e., the number of jobs in the queue, is displayed in this rectangle. The number of busy RTs is displayed in the RT spot. It is pink if at least one RT is busy, and grey otherwise. Again the number of RTs and the data access time is displayed in the lower right hand corner.

3.2.4 IC documentation*

The graphical representation produced by the D* system from the C&E source code is very similar to Figure 3. This also provides a way to validate the correctness of the source code specification.

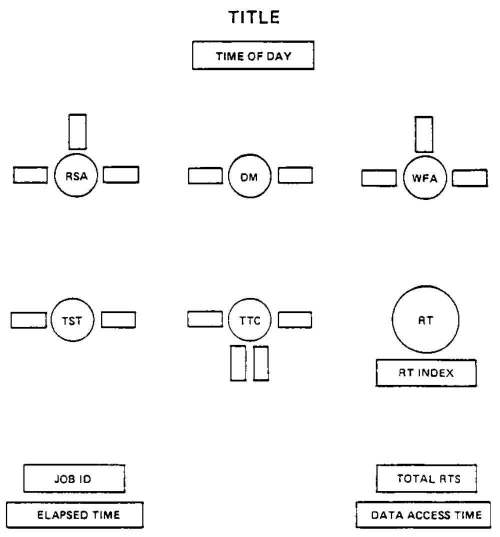

Figure 4. IC* Display of The Simulation.

4. SIMULATION RESULTS AND ANALYSIS

4.1 Assumptions of the Current Process Being Modeled

Customer trouble reporting for POTS-like service is currently supported by the Loop Maintenance Operations System (LMOS). In this process, the RSA module is a component of LMOS. The TST module is the Mechanized Loop Testing (MLT) system that is automatically accessed by LMOS to perform a set of analog tests on the circuit reported by the customer to be in trouble. If, after testing, the customer wishes the process continued, the TTC module uses LMOS to establish the trouble ticket. If the trouble is dispatchable, LMOS as the WFA assigns it to the work list of an appropriate RT, who makes the repair, tests that the trouble is actually fixed, and closes out the trouble.

If the trouble is not dispatchable, it is referred to a Maintenance Analyzer (MA) for manual sectionalization and trouble isolation. Part of the MA's actions will be to collect all possible data about a trouble using the TST module that may include generation of another test. Following this analysis, which is modeled as the TTC process, the trouble will either be considered dispatchable or OK. If it is dispatchable, it then follows the path for dispatchable troubles outlined above. Otherwise, the customer is contacted to verify that the trouble is now OK or to get details for further analysis. This process continues until the trouble is resolved, either to a dispatchable state or to the OK state.

In order to validate the model, some assumptions have to be made about a "typical" maintenance center. The numbers chosen are not expected to match exactly any actual operation, but should be representative of a normal operation. The assumptions follow.

The maintenance center was assumed to serve 250,000 lines and have eight RSA attendants. The trouble rate per line per year was assumed to be 0.65. Therefore, there would be 624 troubles received per working day. Their assumed distribution over time of day is shown in Table 1.

Time	*Troubles Arrived*	*Troubles Handled By Each RSA Attendent*
7:30 - 8:00	16	2
8:00 - 9:00	48	6
9:00 - 10:00	72	9
10:00 - 11:00	72	9
11:00 - 12:00	64	8
12:00 - 1:00	56	7
1:00 - 2:00	56	7
2:00 - 3:00	56	7
3:00 - 4:00	64	8
4:00 - 5:00	56	7
5:00 - 6:00	32	4
6:00 - 7:00	32	4
Total	624	78

Table 1. Trouble Arrival Distribution in a Day.

Table 2 shows other assumptions made to run the simulation.

Module	*Category*	*Count*
RSA	%Stapled Jobs	5
RSA	%Troubles Closed Through Negotiation	14
TTC	%Dispatchable after Initial Screening	47
TTC	%Dispatchable after Trouble Investigation	28
TTC	%Test-OK	11
RSA	Take Customer Trouble Report	45 Sec
TST	MLT Test Time	45 Sec
RSA	Conclude Customer Contact	30 Sec
TTC	Generate Trouble Ticket	50 Sec
TST	Gather Data To Analyze Trouble	180 Sec
TTC	(Manual) Trouble Investigation	300 Sec
RT	Outside Repair Time	90 Min
RT	Inside Repair Time	30 Min

Table 2. Assumptions on Probabilities and Processing Times for The Simulation.

The work hours for RTs were assumed to be from 8:00 am to 4:00 pm. The average repair time for inside jobs and outside jobs were assumed to be 30 min and 90 min, respectively. Therefore, WFA assigns jobs to inside RTs from 8:00 am to 3:30 pm, and to outside RTs from 8:00 am to 2:30 pm. The ratio between the numbers of inside RTs and outside RTs was assumed to be 1:4 and was always held constant.

4.2 Receipt-to-Close Time Analysis

Receipt-to-close time is the elapsed time from the receipt of a trouble until the trouble is closed. The simulation ran for varying numbers of repair technicians and different data access times to observe the receipt-to-close time. Data access time is the time to read or write data stores and was assumed to be constant for each run. It varied between 10 seconds and 20 seconds in the simulations to reflect different data management strategies. Based on these time variants, the number of repair technicians was varied by 8, 11, 14, 17 and 20. Since there is only one RSA attendant in the current simulation model, the numbers of RTs have to be multiplied by eight to get the total number for the maintenance center modeled. Among these repair technicians, one fourth of them were assigned as inside technicians and the others outside technicians.

The simulation was run for a three work-day period in order to reach a steady state. The data collected were from the third day, which represents a regular day of operations. All the completed repair jobs were stored at queue QTO, associated with their receipt-to-close time. From this, the average receipt-to-close time could be calculated. The ratio of the completed jobs which include repaired jobs, test-OK jobs, and CPE (Customer Premise Equipment) jobs and the total incoming jobs was also calculated. Based on these data, Figure 5 shows the average receipt-to-close time and the out-jobs/in-jobs ratio versus the number of repair technicians for the simulation under the assumption of a 10 second data access time. Notice that the ratio can be greater than 1 because the jobs left over from the last day are processed first. Therefore, the total number of completed jobs could be more than the number of jobs received for a day.

From Figure 5, the smallest number of repair technicians required to meet the repair commitment time can be determined by observing the intersection of the average receipt-to-close time curve with the horizontal line representing the repair commitment time.

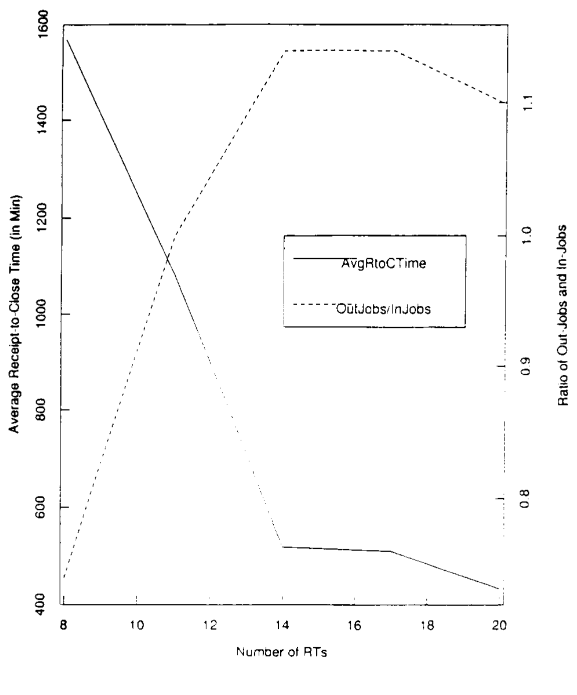

Figure 5. Average Receipt-to-Close Time and Ratio of Out-Jobs and In-Jobs Versus Number of RTs.

The comparison of the effect of different data access times is shown in Figure 6. Every module accesses the DM at least once. Although the data access time was just increased by 10 seconds, the average receipt-to-close time was raised significantly due to the queuing delay caused by this increase.

For the simulation run with 10 second data access time and 14 repair technicians, Table 3 shows a summary by job type of the number of completed jobs and their receipt-to-close time range and average receipt-to-close time.

Job Type	*Total Number*	*Time Range* (in Min)	*Average Time* (in Min)
CPE	13	2.07 - 5.42	2.61
Test-OK	6	12.38 - 24.58	14.45
Inside	11	41.2 - 1083.73	381.51
Outside	47	99.75 - 1133.75	550.82

Table 3. Total Number and Response Time for Different Completed Jobs.

The receipt-to-close times for inside and outside jobs have a wide range. This is due to the work hours of an RT. Troubles arrive until 7:00 pm, while WFA assigns jobs to inside RT by 3:30 pm and to outside RT by 2:30 pm. Therefore, the jobs that have to be carried over to the next day have a much longer receipt-to-close time.

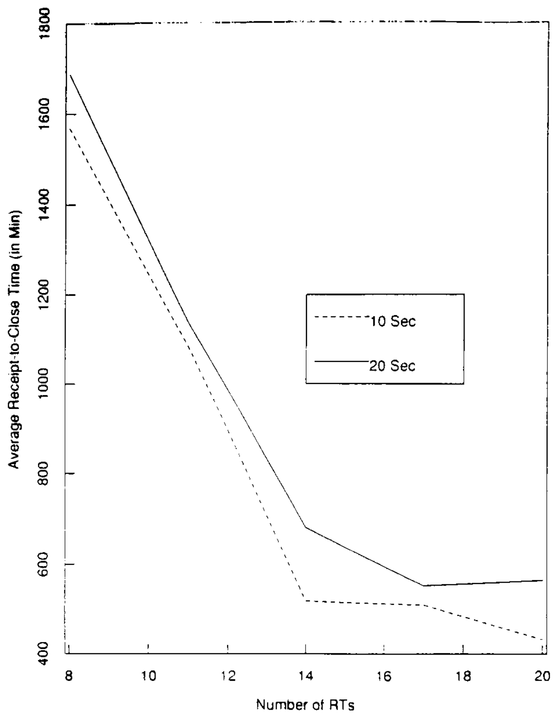

Figure 6. Comparison of Average Receipt-to-Close Time Versus Number of RTs for The Simulation Runs with 10 Second and 20 Second Data Access Times.

4.3 Workload Analysis

An examination of all the queues in the system indicates that four of them show more queue length variability than the other. They are the three queues associated with WFA, and the QD, which is associated with the DM. One would expect that since WFA is the module to dispatch jobs to RTs, it would be a bottleneck in the system. This fact is validated from the queuing effect at WFA. Table 4 provides a summary description of the queues. The discussion of the operations of WFA follows.

Queue Name	*Module Associated*	*Description of Jobs in the Queue*
QW0	WFA	Jobs (for outside RTs) carried over from the previous day
QW1	WFA	Jobs sent back from DM or RTs
QW2	WFA	Jobs sent from TTC
QD	DM	Jobs sent from TTC, TST, or WFA
QTO	TTC	Completed jobs including test-OK and repair jobs

Table 4. Description of Jobs in Queues.

When TTC sends jobs to WFA, they are put in QW2. WFA retrieves jobs in QW2, sends them to QD of DM. After DM processes these jobs in QD, it returns them to WFA. They are put in QW1 and are thereby ready to be assigned by WFA to RTs. The completed repair jobs sent from RTs are also put in QW1. WFA starts to assign jobs to RTs at 8:00 am when RTs start to work. From the assumptions described in Section 4.1, WFA stops assigning jobs to RTs at 3:30 pm. Thus, between 8:00 am and 3:30 pm, WFA processes jobs in QW1 and QW2 with QW1 having a higher priority. That is, WFA first processes jobs in QW1 until there is no job in QW1 and then processes jobs in QW2. After 3:30 pm and before 8:00 am in the morning, WFA only processes jobs from QW2. QW0 is the queue for storing the outside jobs retrieved from QW1 between 2:30 pm and 3:30 pm. These jobs will be assigned to outside RTs at 8:00 am first on the next day. That is, QW0 has the highest priority among these three queues.

Figure 7 shows the queue length changes of QW0, QW1, QW2, QD, and QTO over time of day for the simulation run with 8 RTs and 10 second data access time. From this figure, jobs in queue QW0 were processed until QW0 was emptied at around 12:30 pm. Since there were 6 outside RTs, these jobs were processed in groups of 6, and each job took about one and a half hours to complete. QW0 started to grow at 2:30 pm, since this is the stop time for assigning jobs to outside RTs.

QW1 grew at the same rate as QW0 shrank from 10:00 am to 12:30 pm since the completed repair jobs sent back from RTs were put at QW1. QW1 was then emptied at 3:00 pm. This is due to the fact that WFA sent the repaired jobs to TTC. The same effect can be seen from the queue length increase of QTO. QW1 started to grow at 3:00 pm because at this time WFA started to process QW2. Jobs in QW2 were sent to QD of DM and then back to QW1. This situation can also be observed from the queue length changes of QW2 and QD.

From Figure 7, it can be seen that WFA mainly assigns jobs in QW0 , which are carried over from the last day to RTs. This is due to the insufficient number (8) of available RTs for the workload. The results of the simulation runs with 20 RTs are shown in Figure 8. The effect of increasing RTs is now described.

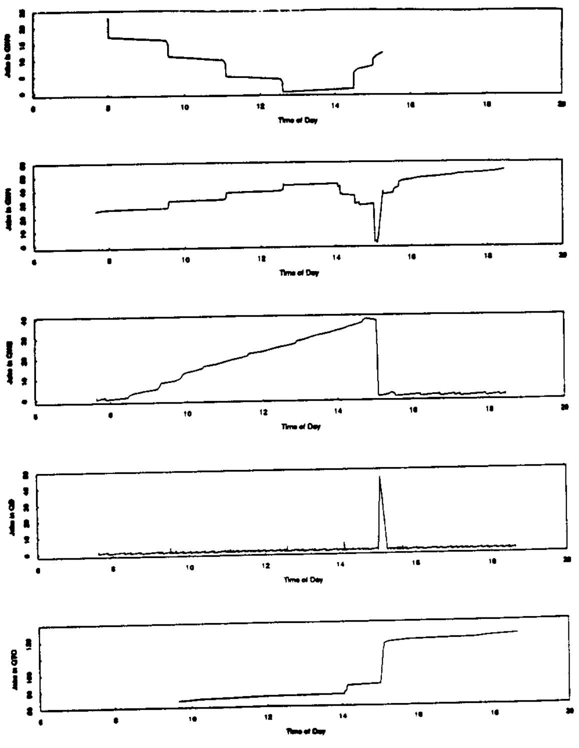

Figure 7. Queue Length Changes over Time of Day (with 8 RTs).

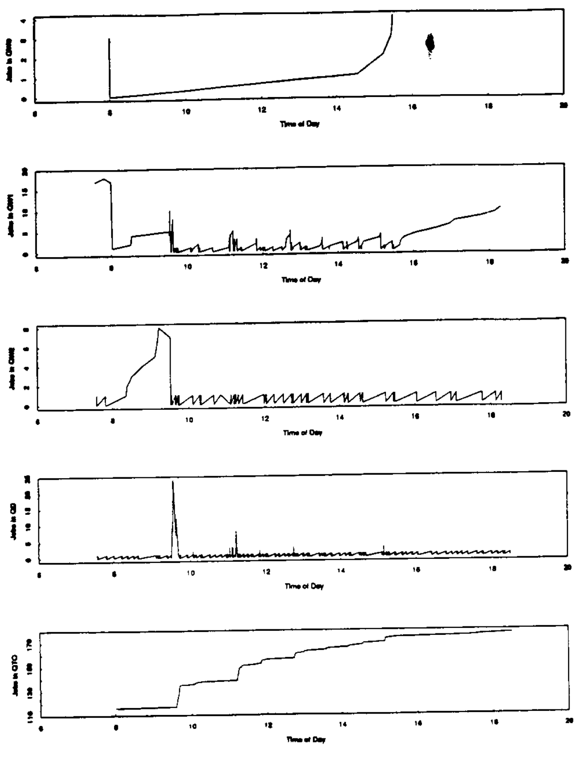

Figure 8. Queue Length Changes over Time of Day (with 20 RTs).

There were only three jobs in QW0 (instead of 23 in the previous case). Therefore these three jobs were assigned to RTs at 8:00 am. WFA then worked on QW1. At 9:30 am QW1 was emptied (jobs in QW1 were either assigned to RTs for repairing or sent to TTC for closing out). Then it grew again since WFA processed QW2, and jobs were sent to QD of DM then back to QW1. This situation reappeared again at 11:00 am, 12:45 pm, etc. due to the completion of dispatched jobs. Notice that only 5 jobs were assigned to RTs during these time periods. That is, some RTs were idle and 20 RTs were more than enough for the current workload. This fact can also be observed from the queue length changes of QTO. The completed repair jobs arrived at QTO with a lesser rate during later hours.

From the results of the simulation, there are two places where the simulation model could be improved:

1. For smaller numbers of RTs, many jobs queue at QW0. These will be processed first the next day. As a result, inside RTs could be idle for some time until the jobs in QW0 are processed.
2. Since the same queue, namely QW1, is used to accommodate jobs to be assigned to RTs and jobs repaired and sent back from RTs, for smaller numbers of RTs the long queue at QW1 could possibly prevent a repaired job from being sent to TTC and closed right after it is repaired.

The current model will not have these two problems when there are enough RTs.

5. CONCLUSIONS AND FUTURE WORK

In this paper, a simulation model for customer trouble report processing in the operations environment, its implementation using the IC* system, and the performance analysis have been presented. From this work, we have:

- demonstrated the capabilities of providing visual validation of the simulation model by displaying transaction flows and queuing effects for the operations process modeled.
- validated that the simulation model did approximate the actual operations for the process flow chosen.
- studied the receipt-to-close time as a function of the number of repair technicians and data access time for different types of troubles including CPE jobs, test-OK jobs and repaired jobs.
- analyzed the workload distribution and identified potential bottlenecks in the operations system modeled.
- performed sensitivity analysis for OS components to offer suggestions for achieving better performance. These include: (1) determining the smallest number of repair technicians required to meet the repair commitment time. This is done by studying the impact on the average receipt-to-close time of the number of repair technicians, and (2) deciding the need to improve the data access speed by studying the impact on the average receipt-to-close time of varying data access times.
- shown the capability of the IC* system to simulate an operations process and its specified transactions.

- provided modeling improvement suggestions for the future simulation modeling effort.

The IC* system provides the 'richness' of environment required to perform the simulation modeling described in this paper. This richness provides the required flexibility to study and validate initially coarse models and then to expand them to more complex and refined models.

Future plans include modeling more sophisticated customer trouble report processes, modeling different network operations such as network detected trouble processing, and modeling data management alternatives. For the first two plans there is a need to model a more complicated operations environment. For the last plan (see [6] for a detailed description of this plan) the performance for different data management alternatives can be compared in order to determine the best data management strategy to be incorporated into future operations systems.

REFERENCES

1. "System Engineering Requirements for Network Operations Processes," Bellcore Special Report, SR-TAP-000699, Issue 1, April 1987.
2. "Operations Systems Strategic Plan," Bellcore Special Report, SR-NPL-000022, Issue 3, December 1986.
3. E. J. Cameron, D. M. Cohen, B. Gopinath, W. M. Keese, L. Ness, U. Premkumar, J. R. Vollaro, "The IC* Model of Parallel Computation and Programming Environment", IEEE Transactions on Software Engineering, March 1988.
4. E. J. Cameron, D. M. Cohen, B. Gopinath, W. M. Keese, L. Ness, U. Premkumar, J. R. Vollaro, "IC* Super Computing Environment", Proceedings of the Workshop on Parallel Computation, Princeton, Sept. 1987.
5. R. A. Becker and J. M. Chambers, "S An Interactive Environment for Data Analysis and Graphics," Wadsworth Advanced Book Program, Belmont, California, 1984.
6. A. L. P. Chen, E. C. Anderson and M. Y. Lai, "Simulation Modeling of Data Management for Network Operations," Proc. Bellcore Database Symposium, SR-STS-000783, Sept. 1987.