The Growing Family of Federal Standards for HF Radio Automatic Link Establishment (ALE)

Part IV: Network Simulation for the Radio Amateur

Complex network designs need to be thoroughly tested. The first step of testing is simulation.

David Sutherland and Dennis Bodson, W4PWF

Introduction

An important question stems from the recent series of *QEX* articles on the Status and Future of HF Digital Communication, and that is "How can radio amateurs use computer simulation modeling to investigate questions in the field of Amateur Radio?"[1]

The purpose of this article is to interest the radio amateur in computer simulation modeling as an experimental method. It is not our intent that the radio amateur could read this article and immediately program a simulation. However, it is our contention that simulation modeling is accessible to the radio amateur. The most important requirement is a thorough knowledge of what the investigator is trying to simulate. So, why would you even consider simulation modeling?

Suppose you have an idea for a structure of networking protocols to speed up transmission of digital messages, to identify the propagating frequencies, and to provide error correction. Suppose the idea is intended to work with hundreds of radio stations operating around the clock, over many frequencies. Imagine the frustration of trying to coordinate several hundred other amateurs in enough 24-hour tests to gain reliable performance measurements of this new radio networking idea! Now imagine trying to do it all over again after fixing all the problems and bugs which were revealed by the first try. This sounds as much fun as banging your head against a brick wall, and it is.

How can an idea like this ever get tested enough to get the kinks out? Computer simulation modeling is an answer. The authors develop and use computer simulation models of ALE HF radio digital networks to determine network performance under proposed networking protocols and networking features. The idea is to use simulation in lieu of over-the-air testing so that development of federal HF radio standards can proceed, although a feature will be included in the standard only after being proven in over-the-air testing. Over-the-air testing can be expensive, labor intensive, and time consuming for radio manufacturers, standards developers and users. Even so, the authors do not propose that simulation modeling replace over-the-air testing. The predictions given by a simulation are always approximations to expected network performance (see Note 1).

The costs of over-the-air testing can be reduced by simulation. Most of the major bugs should be revealed prior to expensive and time consuming over-the-air testing. Here is a simple example.

Suppose the stations in a very busy HF ALE digital network have long queues of digital messages waiting for transmission. Suppose every station handles each message immediately after transmitting the previous message. It's easy to imagine the situation where every station is attempting to transmit, but no station can receive since no station is listening for incom-

[1]Notes appear on page 18.

David Sutherland
9400 Elm Ct #510
Denver, CO 80221

Dennis Bodson, W4PWF
233 N Columbus St
Arlington, VA 22203

ing messages. A simple solution is for each station to pause for a short period of time after completing a transmission and listen for incoming traffic. A computer simulation revealed this problem before it got on the air which would have added "EM pollution" to the already crowded HF spectrum. That's one problem that did not make it to an over-the-air test, and the solution was also verified by simulation.

Prerequisites

What does an amateur need in order to design a simulation program? The prerequisites are divided into two areas: equipment (hardware and software) and knowledge (skill and experience).

An AT class personal computer (PC), 286 Central Processor Unit (CPU), is basic. You don't need a workstation or a mainframe. A 386 or 486 (CPU) based PC is preferred. A floating point coprocessor is a plus if there is a great deal of floating point operations in the simulation program. The more random access memory (RAM) the better, but only if the simulation program uses the extra RAM.

The authors run simulations on a 486-50 MHz PC, with an EISA bus (32-bit), a SCSI 450-Mb hard drive with a fast caching controller card to speed up input and output (I/O), and 16-Mb of RAM. This is an expensive setup. The same simulations run at home on a 386-33 MHz PC with an 80-Mb hard drive and 4 Mb of RAM or on a 286-25 MHz PC (16-bit) with a 40-Mb hard drive and 2 Mb of RAM. They run more slowly at home, of course. As PC prices continue to fall, simulations will be more feasible for the amateur.

The amateur simulator needs to know a high level basic programming language and must have the language software (compiler/interpreter). High level means closer to English than machine language. BASIC, Pascal, FORTRAN, and C all qualify. Knowledge of data and file structures and efficient program design techniques are desirable, but are not absolutely necessary.

Knowledge of basic statistics is needed to analyze results and will help in deciding on the structure of the input data.

The most important prerequisite is a thorough knowledge of radio systems and the network operating procedures that are being modeled.

Simulation Characteristics

The authors have developed a discrete event simulation model for ALE HF Radio networks.[2] This will be the basis for describing some of the procedures for designing a network simulation.

Network simulations are generally discrete event simulations, which usually are stochastic as opposed to deterministic. Stochastic refers to randomly generated input data. Since the input data is randomly generated, the output data is also random in a sense. Specifically, there will be variability associated with the answer due to the randomness of the input. The relative extent of the variability gives an indication of how accurate the answer is. A deterministic simulation has no random components.

Network simulations are generally dynamic as opposed to static. A dynamic simulation evolves over time and may measure some performance parameter as a composite over the entire simulated time of the computer run or the measurement may be made periodically over the course of the simulation run. A static simulation does not evolve over time but generally simulates a situation with time fixed. Monte Carlo simulations, strictly speaking, are static simulations; however, some authors call anything with a random component "Monte Carlo."

Network simulations are generally discrete in that the discrete states or conditions of channels and stations change instantaneously in the model whether or not these changes are actually instantaneous. Engineers who do simulation call some components of the simulation finite state machines. A simple light switch can be described as a finite state machine. It has two states, on and off, and is always in one of those states. When the switch is moved from one state to the other, we assume that the state changes instantaneously. A simulation would assign a particular time for the event indicating the change of state of our light switch, and would not consider the real time it takes our hand to move the switch from one position to the other.

This abstraction is at the heart of simulations. A simulation does not include every detail, but abstracts some of the detail. In our light switch example, the details of the electricity moving in the wires and the light bulb element heating up are left completely out. The state of the light switch could also be extended to include the state of the light.

If the detail is important it should be included in a simulation. For example, the effects of the propagation condition of our radio channel on an individual bit in a packet of data could be left out if we are not considering such effects in our simulation. So we might consider the frequency either open or closed (like our light switch) and assume that the message is received as sent if the channel is open. However, if we are simulating the performance of a forward error correction scheme, then the effect on each individual bit must be considered.

A simulation model gives performance estimates of a system which cannot be described by a single equation or a small set of simple equations. Usually the situation is so complicated that it may not be easily described analytically or there are just too many interactions to consider.

Assumptions and Limitations of the Simulation Model

Simulation results are meaningless without a listing of the assumptions and limitations of the model. Assumptions and limitations indicate how far the model deviates from the real system. An example of a simplifying assumption is that a digital message will be received by another station, as transmitted, if the radio channel between them has an SNR of 5 dB or better. An example of a limitation may be that a simulation covers daylight hours only. Another limitation could be that the simulation is run only five times with different sets of randomly generated input data and that statistics are based on these five runs.

Assumptions and limitations serve to simplify the simulation model and its development. If, for instance, propagation time is not important to your simulation, then you might assume that the signal propagation time is effectively zero. This means that every time a message is transmitted you do not have to add the propagation time to the simulated time that the channel is busy with this transmission. An alternative is to assume that the propagation time is included in the time it takes for a message to be transmitted. Either way, computing time is reduced by this minor simplification. The size of the program is reduced, and the model is not so complex.

Assumptions are not intended to hide or ignore features of the real system which are important to the model. The above statements about propagation are moot if multipath delay is important. Assumptions need to be reasonable and reflect the system being simulated in its normal state.

A Random Number Generator

Pseudo-random number generators are important parts of discrete event simulations. They are used to provide a stream of numbers whose statistical characteristics are similar to a true sequence of random numbers which are uniformly distributed between zero and one (the probability of a specific number between zero and one occurring is the same as for all other numbers in the same range). The pseudo-random number generator actually generates a long sequence of integers which are normalized to produce the pseudo-random numbers. Pseudo-random numbers will be called random numbers in the rest of this article. We don't go into detail on random number generators but will provide one which should get the interested experimenter started.

$$Z_i = a Z_{i-1} \bmod m \qquad \text{Eq 1}$$

$$U_i = \frac{Z_i}{m} \qquad \text{Eq 2}$$

The integer sequence generator is defined recursively by Eq 1, where a is 630,360,016, m is 2,147,483,647 (or 2^{31}-1) and i is an integer between *1* and *m-1*. The *mod* operator is the usual modulo operator (m mod n is the integer remainder of integer m divided by integer n). The uniformly distributed random numbers, U_i, on the interval (0,1) are obtained from Eq 2. This generator is widely used in applications requiring uniformly distributed, random numbers.[3,4] The last integer of the sequence seeds the next integer and therefore is the seed for the next random number. This generator has a full period. That is, the integer stream covers every integer between 1 and m-1 before it starts over again. Any integer in this range can be used as a starting point (initial seed).

Methods of doing large integer arithmetic are generally available in most modern programming languages. If not, the same generator can still be used by using synthetic division and multiplication techniques (see Note 3).

An alternative is to use the random number generators available as a library routine within programming languages or within other software. Exercise caution in using unknown random number generators, they may not be as random as you think and should be tested. However, this type of generator is a good place to start and may provide what you need.

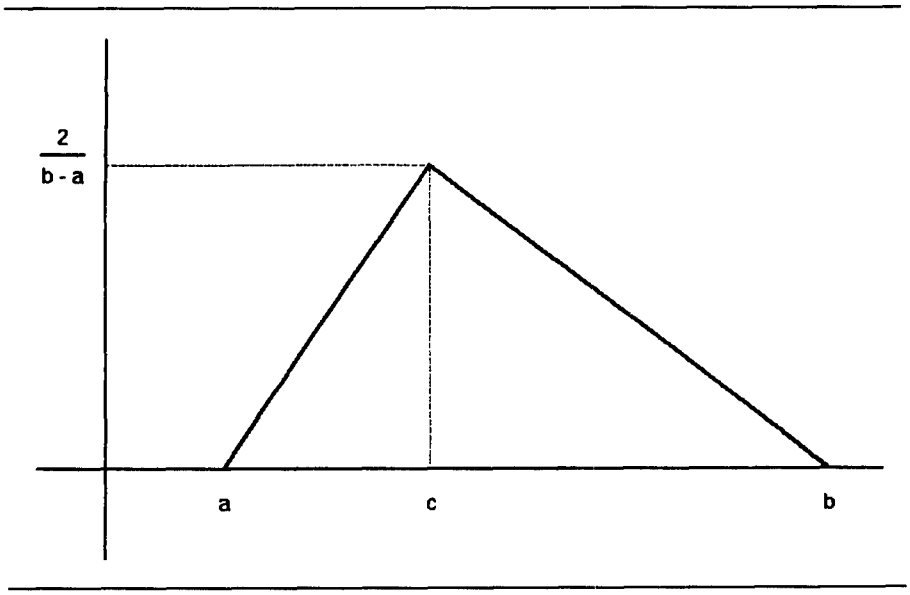

Fig 1—Triangular distribution.

Modeling the Input Data

To model input data it is necessary to statistically describe each of the input data parameters. A data parameter of the message traffic, for example, could be arrival time, length of message, originating station, destination station, etc. The statistical distribution, the mean (average), and the standard deviation of the input data parameters must be known. From this information the parameters which characterize the input can be generated using random number generators (see Note 3).

In the absence of knowledge about the input data parameter in question then the following procedure utilizing a triangular distribution can be used (see Note 3).

The example parameter is the random length, in minutes, of a voice conversation on a frequency. Let a be a reasonable estimate of the minimum length of a conversation, let b be a reasonable estimate of the maximum length of a conversation, and let c be a reasonable estimate of the most likely length of a conversation. Note that c is not the average but the mode. Now let the percentage of calls with length c be $2/(b-a)$. This value is used to normalize the distribution; that is, the probability of a conversation having length greater than or equal to a but less than or equal to b is 1. The triangular distribution of conversation lengths is shown in Fig 1. The average conversation length is $(a+b+c)/3$.

The length of the ith conversation is given by Eq 3, where U_i is the next uniformly generated random number in the random number string from Eq 2.

$$L_i = \begin{cases} \sqrt{cU_i} & 0 < U_i \le c \\ 1 - \sqrt{(1-c)(1-U_i)} & c < U_i < 1 \end{cases} \qquad \text{Eq 3}$$

If I can guess at maximums and minimums, then why can't I just guess at the mean (average) and then use that average to model each conversation? This means that every conversation in the simulation will have the same length, which could lead to simulation failure. For example, consider a simple factory welding process on an assembly line. The process takes 5 minutes, on average. Assume that the average interarrival times (time between arrivals) of the materials to be welded is 6 minutes. Having the average interarrival times greater than the average process time ensures that we will have no backups, doesn't it? A simulation of the process using the measured average time for the process to model each of the welding processes, and using the average interarrival times to model the arrival of each set of materials, will indicate that this is true.

The real system could incur backlogs hours long. What happens? By using only the averages to model processes or arrival times the statistical variation is ignored. There is possibly another mistake made in the simple example above. The simulator may not know enough about the input data and may not realize the variations possible in

the welding process or the delivery process of material.

Notice one more thing in this example. By determining and using the statistical mean, variance, and distribution to model the input data in the welding process above, the simulation designer has abstracted all the variables mentioned into one statistical model. It might seem complicated to deal with the statistics, but it is simpler since the model for the welding process could encompass much more detail.

Simulation Events

Managing the occurrence of events is the most important part in developing the simulation program. The simulation events are either randomly created input data, they are events which are created by the simulation program, or they are housekeeping events. In an HF radio network simulation, example events are: message arrival, by which messages arrive at a particular radio station for transmission; message departure, which handles the completion of the transmission of a message to another radio station; call attempt, which models an attempt by a radio station to link with another radio station; and hourly housekeeping events, by which the program compiles statistics and resets variables and event counters after each hour of simulation time.

The event records are kept in a queue or in a sorted list of records. Each event record contains all necessary information about the event: event type, event time, radio station, call signs, frequency, etc. The simulation will read the next event to occur and call the appropriate subroutine of the simulation program to accomplish the event. Any new events created by this action are assigned an event record and the record is placed in its sorted position in the event queue.

The queue or stack of event records could be a linked list (linked by pointers) or an array (vector) of records. In either case the event records are kept in sorted order, by time of occurrence with the next event at the top of the stack or front of the list.

Sorting routines are available in programming software. For additional sorting methods, see Notes 5 and 6.

State Variables and State Changes

As the events occur, the associated simulation program subroutines change the state variables. The state variables, together, completely describe the simulated system at any point during the simulation.

An important state variable is the simulation clock. In a discrete event simulation the events generally drive the clock. The first action of an event subroutine is usually to advance the clock to the time of the event. Another important example is the state of each frequency or channel. In this case the state variable could indicate, 0-idle or 1-busy like the finite state machines mentioned previously (the light switch).

State variables are used in decision making by the simulation program. They are also handy debugging tools.

Queues and Stacks

Messages (actually records characterizing the messages) waiting at a station for transmission are normally held in a queue which is sorted by arrival time. When a radio station is finished receiving a message from another station or finished transmitting a message, the first message in the queue is taken out of the queue for transmission. The rest of the messages gain one position in the queue. It's like waiting in the checkout line at the grocery store. This is a first-in first-out queue.

Event records are held in a stack. This is the same as the queue mentioned above except that the events are held in the order of occurrence in the simulation and not in the order of arrival to the stack. There are two common ways to set up each of these structures. One is a linked list and the other is an array (or vector).

A linked list uses pointers and can take advantage of computer dynamic memory allocation. That is, the records which characterize the messages in a stack only take up memory in the computer when they actually exist. When the message transmission has been completed by the simulation, then the record is deleted and the allocated memory is available for other use.

Definitions and descriptions of these structures are given in the documentation of programming software. Methods of using these structures can be found in a text on data structures, see Notes 5 and 6, for instance.

Propagation Model

All the previous details are, in a sense, just bookkeeping. Now comes the interesting (hard) part—the radio channel (frequency) propagation model. How do you model the effect of the radio channel on the messages you transmit? The answer depends on what you are trying to measure. For instance, if you are measuring the delay of messages caused by the overhead for a particular networking protocol, the propagation model may be an on/off model: another finite state machine. That is, the channel is either open or closed. No attempt is made to model the degree of channel reliability. The authors are using the HF propagation prediction program IONCAP for this type of simulation model.

On the other hand if you are testing an error detection code requiring ARQ and retransmission, the effects of the channel on individual bits is important. The propagation model must then indicate whether an individual received bit is good or is in error. A channel mode that uses bit error statistics may be in order here.

If one does not have a propagation prediction program in hand, such as IONCAP, then a good way to model a channel is by using your own experience or statistics that you have compiled. For example, you probably know which frequencies propagate at night or in daylight hours only. You are probably familiar with the effects of the sunspot cycle and what propagation performance you can reasonably expect with a certain sunspot number. And you probably have some experience of seasonal variation. You can include this information in your propagation prediction model.

A simulation program does not necessarily need a complicated mathematical model for simulation of radio channel propagation. A simulation designer recently assumed that five out of the ten available HF frequencies were open at any one time.[7] However, if you report your results to others, you should detail the propagation model. Limitations, assumptions, or restrictions that are placed on the propagation model are limitations to the simulation.

Examples

The following examples are generalizations of simulation events in the simulation the authors use to model HF ALE Radio networks (see Note 2). The program was developed using top-down design methods; however, general flow chart model diagrams can describe the logical flow.

A message arrival event is detailed in Fig 2. Most events begin by advancing the simulation clock. The first decision is a determination of the state of the transmitting radio station. If the station is busy (transmitting or receiving another message, for example), the newly arrived message is placed in that

station's message queue. If the station is idle then the list of available channels (frequencies) are sorted by the recorded (or predicted) channel quality. An event record is created for a call event on the first channel in the sorted list. The event is placed in the proper position in the event queue. Program control returns to the main program which pops the next event from the top of the event queue and calls the appropriate subroutine to accomplish the new event.

A call event is illustrated in Fig 3. A call is an attempt by a radio station to link with another station over one of the available frequencies (channel i). As before, the call event begins by advancing the simulation clock. The state of both channel i and the receiving station is determined. If both the receiving station and the channel are idle (not busy) and the two-way propagation path between the stations is open, then the two stations link (message transmission begins). The state of both stations and channel i is changed to busy. A departure event is created and placed in the event queue. The departure event occurs when the message transmission is finished and accomplishes the return to the idle state for channel i and for the two stations.

If the radio station is busy or if channel i is busy or closed (propagation not possible), the channel state is recorded (busy or closed). If channel i is the last channel in the sorted list of channels then the message is placed on the end of the message queue to await a retransmission attempt. Otherwise a new call event is created for the next channel, $i+1$, and the event is placed in the message queue. Control returns to the main program.

Many details have been left out of these descriptions for the sake of brevity and clarity. For instance, the propagation states of the channel in the call event procedure are determined by the implementation of the propagation model. Another omitted detail is how a newly created event is placed in sorted position in the event queue.

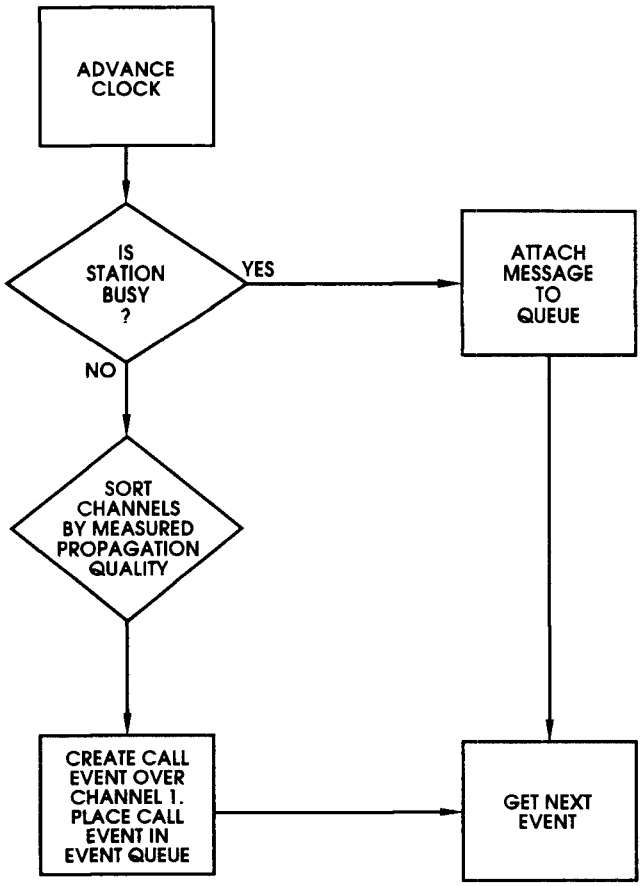

Fig 2—Arrival event flow chart.

Example Results

The authors have recently completed a simulation study of the effects of sounding on an ALE HF radio network (see Note 2). Sounding is the periodic broadcast of self identification information which may be monitored, recorded, and evaluated by other radio stations. The state of the one-way HF propagation channels may then be determined. The channel quality information may then be used by radio stations in the network to select a channel that is most likely to provide a link. However, the overhead of sounding may be detrimental to network performance: A sounding station cannot receive incoming traffic and a channel being sounded upon is unusable by other stations.

The conclusion of the simulation study is that sounding is generally detrimental to network performance. Any efficiency due to the additional channel information obtained by sounding is outweighed by the interference to other traffic caused by the periodic sounding.

An exception to our major conclusion is that in poor propagation conditions, with low traffic rates, sounding may enhance some aspects of network performance. This is illustrated in Fig 4, which shows hourly call success rates for a twenty-four-hour period. Call success rates are shown: (1) with each station sounding, on each available frequency, once every hour, and (2) with stations not sounding. Each data point represents the average call-success by hour for twenty-five runs of the simulation. The success rate is better without sounds during the first 13 hours of the period. In the second part of the twenty-four-hour period, during more adverse propagation conditions, sounding enhances the call success rate. This is due to the identification, by sounding, of the few, 2 or 3, propagating HF channels. Therefore, the radio stations are not wasting time trying to link over the channels that are not usable.

These results and conclusions from a simulation study show some of the possibilities of simulation of radio networks.

Conclusion

This article has presented simulation modeling as a tool for experimentation and investigation. This is intended to serve as a starting point for

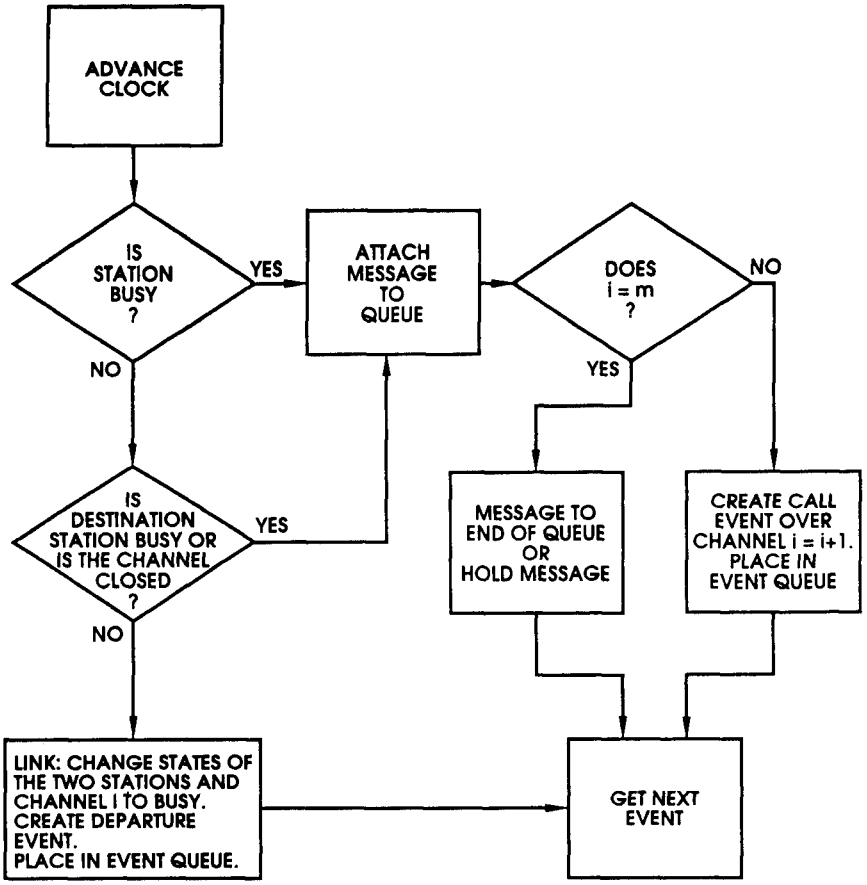

Fig 3—Call event flow chart.

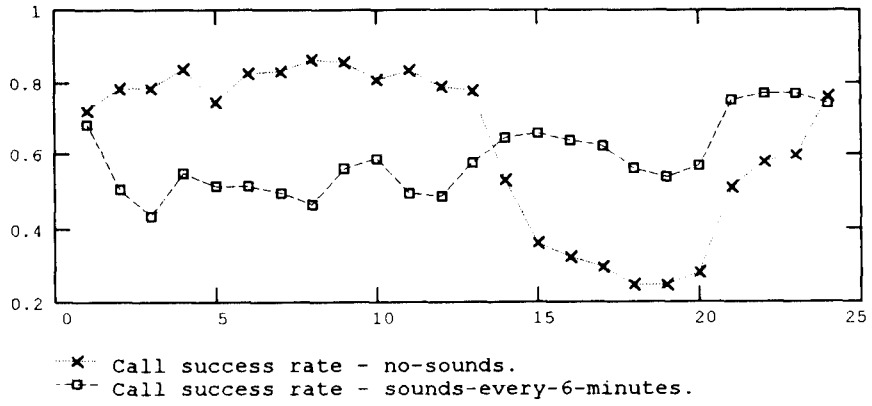

Fig 4—Simulated network performance by hour-data traces—5 messages per hour—sounds every 6 minutes.

the prospective simulation designer. Further information and concepts on simulation modeling are available by consulting the books in the Notes.

The book by Law and Kelton (Note 3) is a good reference and text for general simulation. It is self contained and well referenced. The book gives algorithms for many example procedures, including input data generation and random number generation. It requires some knowledge of statistics and probability but contains a comprehensive review chapter of these subjects. Dr. Law teaches a short course on simulation modeling every few months at George Washington University. The book by Bratley, Fox, and Schrage (Note 4) is a mathematics text that requires knowledge of statistics and probability. The Horowitz and Sahni books (Notes 5 and 6) contain detailed explanations, techniques and algorithms on linked lists, searching, sorting and manipulation of data with pointers to data structures.

Acknowledgments

This work was supported by the National Communications System (NCS) and the National Telecommunications and Information Administration/Institute for Telecommunication Sciences (NTIA/ITS).

Notes

[1]Wickwire, K., "The Status and Future of High Frequency Digital Communication, Part III: Simulating the Performance of HF Digital Networks," *QEX*, No. 126, August 1992, pp 12-17.

[2]Sutherland, D. (1992), NTIA Report 93-291, NCS Technical Information Bulletin 92-21, *Simulated effects of sounding on automatic link establishment HF radio network performance.*

[3]Law, A.M., and Kelton, W.D., (1991) *Simulation Modeling and Analysis*, 2nd Ed, (McGraw Hill, NY).

[4]Bratley, P., Fox, B.L., and Schrage, L.E. (1987), *A Guide to Simulation*, 2nd Ed (Springer-Verlag, New York).

[5]Horowitz, E., and Sahni, S. (1987), *Fundamentals of Data Structures in Pascal* (Computer Science Press, Rockville, Maryland).

[6]Horowitz, E., and Sahni, S. (1983), *Fundamentals of Data Structures* (Computer Science Press, Rockville, Maryland).

[7]Johnson, E. (1992), New Mexico State University, private correspondence. □□