

Distributed Processing for Contingency Screening Applications

Shangyou Hao
Pacific Gas and Electric Company
San Francisco, CA

Alex Papalexopoulos

Tie-Mao Peng
Consultant
San Francisco, CA

Abstract --- The emergence and widespread availability of network computing has made it possible for many computationally intensive applications to take advantage of distributing processing techniques at almost no additional cost. In this paper, we present a successful implementation of distributed processing for screening all possible single contingencies in an electric network model. The implementation is based on the remote execution model that is available in most UNIX based workstations. Our testing results show that near ideal speedups can be achieved for the contingency screening applications.

Keywords: Distributed Processing, Contingency Screening, Security Analysis, Energy Management Systems

I. INTRODUCTION

In a power system context, security is defined as the ability of the system to withstand disturbances. At a certain time, the absence of risk of disruption of continued system operation is evaluated using security analysis tools with respect to a preselected list of "credible" disturbances or contingencies. Currently, security analysis functions used for security assessment have been implemented in actual system operations through a number of software packages in modern Energy Management Systems (EMS). In general, security analysis results should be received within a few minutes up to five to ten minutes in order to be considered reliable. Slower security analysis response would probably render the results unreliable due to changes that may have occurred on the system model that simulates the real-time conditions. Assuming that the level of modeling details of contingency cases is the same as that of the study power flow cases, achieving execution times within the acceptable time frame is an extremely difficult task. For large scale, real life systems the list of single contingencies could contain thousands of contingency events to be tested. If double

contingencies are included, the number of possible events to be screened is in order of millions.

One way to speed up the security analysis evaluation process is to produce a reduced list of contingencies to be fully analyzed. Usually the creation of the reduced list is obtained by the contingency screening processor which scans a predefined list of contingencies and ranks the most important ones for further analysis by the Contingency Analysis Processors[1]. The ranking can be obtained using scalar, network performance-based indices for branch MW/MVA/Amp limit violations and/or branch MVAR and voltage limit violations. The indices are designed to reflect the loading of the entire system.

Contingency screening technologies have been the focus of intense research for more than a decade. As a result their efficiency and reliability have substantially improved. Fast contingency screening methods have been developed to screen both branch MW overload and reactive power flow problems. The methods are based upon bounding the outage effects and attempting to perform a solution only in the stressed area of the system [2-3]. Combined with sparse vector methods, the bounding methods have improved in terms of performance and computational efficiency [4-7]. Other methods have concentrated on improving the overall large scale solutions by means of partial incomplete solutions, reduced network representations and localization of the contingency effects [8-11]. These methods have served the industry well. However, recent changes in the utility business environment put an even higher premium in ensuring that the current level of reliability is maintained. These changes include the emergence of competitive energy marketplace, regulatory reform, increased pressure for essentially uncontrolled access to the bulk power transmission networks, rising cost of money and stricter environment regulation for generation and transmission construction. These changes threaten to erode the traditionally high level of security by exposing the system to smaller security margins and resulting in transmission of electric energy over long distances in patterns other than those for which the transmission networks have been originally designed. The current situation means that there is a need more than ever to screen and process an even larger number of contingencies at a faster pace.

In this paper, we explore the use of the distributed computing capability, now available in most utilities' operating environment, coupled with efficient bounding and sparse vector methods to further improve the speed of contingency screening applications. The availability of networked workstations, that are used to host ported EMS applications from dedicated mainframe EMS computers, presents an opportunity that needs to be explored. Distributed processing, as opposed to parallel processing, does not require dedicated computer hardware and software and general purpose workstations that are networked can be used as distributed processors. In this computing environment, computationally intensive tasks that possess certain properties can be solved extremely fast at almost no additional cost. Contingency screening is one of the applications that can make use of the network computing environment very effectively. In contingency screening, all the outage events are simulated on the same base case and the evaluation of each contingency case is performed independently of the others. The independence of the contingency cases to be evaluated implies that no communications between the processors are required during the main computational task, thus making the contingency screening an ideal application for distributed processing.

In this paper we present a successful implementation of a distributed processing method for contingency screening applications. The implementation of the basic contingency screening program for sequential execution is presented in Section II. The implementation of distributed processing scheme and the required modifications to the sequential code is presented in section III. Test results using distributed methods for a large scale system are given in section IV. The major conclusions of this work and the paper's contributions are discussed in Section V.

II. CONTINGENCY SCREENING IMPLEMENTATION

The steady state of the power system is modeled by a set of nonlinear algebraic equations:

$$g(x, \mathbf{e}_i) = 0 \quad (1)$$

where g contains the power flow balance equations and other equality-type constraints such as interchange requirements and voltage regulations; x is the system state vector that includes the complex bus voltage and the participating generator output; and \mathbf{e}_i can be regarded as the independent outage parameter under consideration (for example, a generating unit or branch outage). The problem of contingency screening is to obtain N sets of the approximate solution to (1) for a given set of contingency parameters \mathbf{e}_i , $i=1 - N$.

In PG&E, we have implemented such a program, named

WORSTCON, which sequentially computes the approximate power flow solutions for all possible single contingencies of an electric network. We make use of the first iteration of the fast decoupled method [13] to compute the network solution and rank the contingencies according to voltage and thermal violations for easy interpretation of the results. For each outage, the voltage angle vector is first solved. Then the reactive mismatch vector is updated before it is used for solving for the voltage magnitude vector. This way, the impact of the voltage angle changes is accounted for in computing the voltage magnitude vector. Many recent advances in contingency screening and several novel methods have been also incorporated in order to improve the performance of the program. The most important ones are very briefly described next:

Sparse vector method: The sparse nature of the right hand side in (2) is exploited in computing the matrix solution. The MA27A subroutine from Harwell [12] is used to compute the matrix factorization of B' and B'' , which is then used by the sparse vector method. Our sparse vector program can solve any four combinations of sparse matrices depending on whether the right-hand side of the solution vector is sparse or full. These four combinations include 1) fastforward and fastbackward substitution; 2) fastforward and full backward substitution; 3) full forward and fast backward substitution; and 4) full forward and full backward substitution.

Local solution procedure: In the basic implementation, only the buses in the nearby local areas are monitored. We implemented a breadth-first search algorithm to determine the local area around the outage [3, 10]. Heuristics are used to expand the local area until the outage impact to the rest of the network is relatively small.

Violation ranking: Ranking of the contingencies was implemented not based on the conventional performance index to measure the stress of transmission system, but rather on actual violations. We found that the program users do have a strong preference in quantifying the severity of the contingencies based on physical interpretations rather than abstract indices. In this case, the ranking can help users quickly weight the importance of the vast amount of information that a screening program can provide.

Contingency equivalence preprocessing: Inclusion of all single contingencies creates many redundant contingencies in the network model that don't need to be screened. With minimum amount of computing effort, these redundant contingencies can be eliminated. In our implementation, this is achieved by using several heuristics to define the

equivalent or the redundant set of the contingencies to be eliminated. For example, one of the identical parallel branch outages is classified as equivalent to the other branch. Another example of a redundant contingency is the outage of the smaller unit at a bus that is supported by multiple generating units since inclusion of the largest unit will suffice.

Island identification: The islanding conditions for all possible outages of transmission branches are identified with a very efficient algorithm before the execution of main computation task. The island identification is implemented with the aid of a depth-first search algorithm. Instead of searching for the islanding condition for the removal of each branch, all branches whose removal causes network islanding will be identified with one depth-first search traverse. In the depth-first search algorithm, the network model is treated as an un-directional network graph and all of the network branches are classified as either a tree edge that is in a forward path or a back edge that is in the backward path [14]. The island identification is achieved by keeping track of all loops that are formed by one tree edge and one or more forward edges. Then edges which are not covered by any loops can be easily identified. The associated branches are the ones whose remove will result in network islanding.

The program operation is divided into three phases: a) preprocessing; b) screening and c) postprocessing. In preprocessing, the program performs the following major tasks:

1. Read input data and user options
2. Identify equivalent and redundant outages
3. Perform island analysis
4. Compute base case branch flow
5. Build up the B' and B'' matrices
6. Order and factorize above matrices.

In the screening phase, the program loops through each contingency case and performs the following tasks:

1. Identify local area of the outage
2. Perform generation redispatch if needed
3. Compute the compensation and solution vector for voltage phase angle
4. Compute the updated reactive mismatch
5. Compute the compensation and solution vector for voltage magnitude
6. Compute branch flow and voltage violations
7. Rank voltage and flow violations.

In the post processing phase, the program calculates a list of the worst case contingencies and repeats the contingency screening analysis for the identified worst cases so that useful information about these worst cases, such as branch MW and

voltage violations, are available.

The program is executed sequentially through each phase. While the first and last phases are difficult to be paralleled, the screening loop is highly parallel in nature. As will be seen, the non-parallel section of the program execution takes less than 3.5 seconds --- an insignificant portion of the total execution time of 217 seconds for a large scale network. Thus contingency screening algorithms are very suitable for a distributed processing environment.

III. DISTRIBUTED PROCESSING CONTINGENCY SCREENING IMPLEMENTATION

In this section, we'll describe the implementation of the proposed distributed processing method. First UNIX-based remote execution in a networked environment and socket communication concepts are introduced. Then we'll discuss the required modifications to the sequential contingency screening program for distributed processing as well as the load distribution methods used. We'll then summarize the steps that the master control program and the screening program need to take to communicate with each other.

A. Distributed Processing Architecture

Unlike parallel processing, which usually requires dedicated hardware and software, distributed processing can make use of existing and general purpose software and network equipment. In most utilities' computing environment this general purpose software and hardware needed for the distributed computing for the most part is currently in place. Our computer network structure, shown in Figure 1, is typical among many other utilities. The network consists of Apollo workstations that are connected by a token ring. These workstations can communicate with other computers that are on an Ethernet through a gateway. Within the token-ring, a number of Apollo DN400 series computers are connected that will be used as the remote processors.

For Unix-based workstations that are networked, socket communication and remote execution services, that are usually supplied by the operating system kernel [15], form the basic paradigm of the distributed processing environment. The sockets are the communication interfaces for the application programs. A socket can be regarded as a high level communication channel that is managing the lower level of communication protocol and device interfaces. In our implementation of distributed contingency screening, the sockets are managed by an even higher level of service --

- the UNIX remote command execution procedures. These procedures, aided by Network File System (NFS) servers,

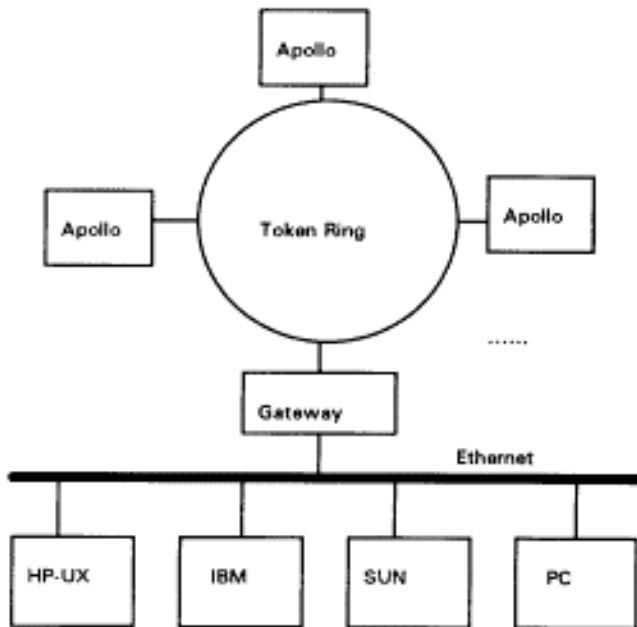


Figure 1: Computer Network Structure

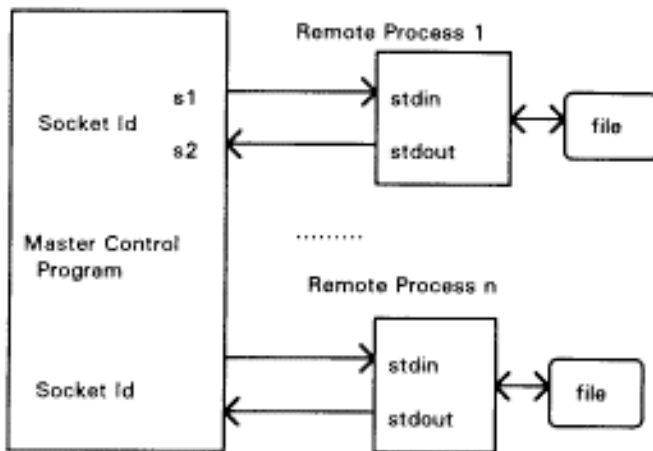


Figure 2: Communication with remote executions

function in a client-server mode. The remote execution paradigm consists of the *rexec* client and *rexecd* server process. In each of the remote computers, the *rexecd* process is executed in a server mode and is serving remote execution (*rexec*) requests from other computers. Using the NFS service, the remote process can obtain other data from the network file, besides the sockets, that may be needed by the invoked applications program. To use this paradigm, the master control program in one of computer nodes will send a request to other remote nodes, then the process *rexecd* in the remote computers will acknowledge the request, create the remote process for services requested and start the

applications program. The UNIX kernel maintains two communication sockets for each remote processing that are used by the master control and applications program to communicate with each other.

The communication of UNIX sockets is further illustrated in Figure 2. After the establishment of the two sockets, the master control program can send a message to the remote computer (slave node) by using a simple write statement to put the message in the *s1* socket. When the control process needs to get information from the remote computer, a simple poll and read from the *s2* socket will be performed. On the remote side, the *s1* and *s2* sockets connect to the standard input and output devices of the remote processors. In addition, the input and output to the remote process is buffered, and a polling function is available to check the availability of the data in the streams. When coding the control program or the remotely executed program, a programmer can simply treat the sockets as file streams.

Distributed processing may not be suitable for applications that need a large amount of inter-processor communications. The reason is that in distributed processing computing, communications with remote processors are slow. In addition, if synchronization of the remote processors is required, additional communications will be needed. Similar services, besides the UNIX remote execution service model described above, can be provided by other distributed computing environments [16]. The standardization of the computing environment should further simplify and facilitate the development of distributed applications.

B. Contingency Screening Implementation

To implement the distributed computing scheme for the contingency screening program, we first coded a master control program and then made the required modifications to the sequential version of the contingency screening program.

The control program uses the *rexec* command to start the execution of the contingency screening program on several predetermined remote processors. The control program supplies the address information that the *rexec* command needs to access the remote processors, passes the load distribution parameters (see below) to the remote processors, and monitors the status of each processor by using a predetermined termination message. It also collects and displays the output from the remote process. Note that the control program sequentially invokes the *WORSTCON* program on the remote nodes and no synchronization between the remote processors is needed. In the distributed environment, the control program signals successful completion when all remote processors complete

successfully. If there are execution errors on the remote processors, the control program will abort after waiting a predetermined timeout period.

To adapt the sequential contingency screening program for execution in the distributed environment, we need to make some minor changes to the sequential code. First we keep intact the sequential portion, namely the preprocessing and postprocessing routines, since they only take an insignificant amount of CPU time. Secondly we change the main loop of the program to evenly divide the computational burden into several portions to be assumed by each remote processor. We need to balance the loading for each of the remote processors to maximize system utilization so that they complete the execution at approximately the same time. The total number of contingencies to be analyzed is calculated in the preprocessing phase that is executed in each of the remote processors. Each remote process then needs to decide which portion of contingencies will assume. If the evaluation of each contingency case takes the same amount of time, we can equally allocate the number of cases in each processor. If the remote processors have different processing speeds, we can achieve load balance by taking their relative computing speed into account. In our implementation, the sequence number for the remote processors and their weights (relative computing speed) are passed to the distributed processor, and each processor determines which portion of the total number of contingency cases will assume. The number of contingencies for processor i is calculated as:

$$NC_i = \frac{W_i}{\sum W_j} NC \quad (2)$$

where NC_i is the number of cases to be analyzed by processor i , W_i is the relative speed of processor i and NC is the total number of cases to be analyzed for the entire system. The starting case also needs to be determined for each process by using the processor sequence number. For processor j , the starting case is computed by:

$$\sum_{i < j} NC_i + 1 \quad (3)$$

Note that the load distribution is not determined in the master control program, but rather it is decided by each processor remotely.

C. Algorithm Summary

We now summarize the execution steps for the master control program:

1. Identify the address of remote machines to be used and

2. setup user login information
2. Invoke the remote process sequentially and pass following information to the remote process: 1) the total number of processors used, 2) the sequence of the each process, and 3) the weighting factors for all remote processors
3. Establish the $s2$ socket id for each processor in the control program
4. Poll each machine for information on $s2$ sockets
5. Read and check if a terminator message is present in the $s2$ socket
6. Call a ranking procedure to produce the final ranking results from the output of each process.

For each of the remote machines, the execution steps are as follows:

1. Load and execute the program when called by the master control program
2. Receive the three load distribution parameters that are sent from the master control program in Step 2 above
3. Start preprocessing of the network data and record the total number of contingencies
4. Determine locally the loading percentage and translate it to the contingency cases to be analyzed as define by (2) and (3)
5. Execute the screening analysis for the identified cases and generate necessary local output file
6. Print completion message onto the $s2$ socket for the master control program to read.

IV. TESTING RESULTS

In this section, two testing results using the contingency screening program, WORSTCON, in a distributed processing environment are reported. In the first test, we used up to four remote processors of identical type (DN425 model). In the second test, we included an additional remote processor of different type, the DN433 model which is about 25 percent faster than the DN425 model. The characteristics of the test system are shown in Table 1. Exception handling due to network failure and redistribution of the loads have not been implemented. In the test, we have also disabled the post processing of the program for easy interpretation of the testing results.

Table 2 shows the testing results of using up to 4 identical remote processors. The elapsed time is measured on the HP-UX machine and the CPU time is measured on the remote processors. The actual speedup is calculated as the ratio of the CPU time of sequential processing and maximum CPU time (from the slowest processor) in the distributed processing mode. The ideal speedup for identical processors is computed by:

$$\text{Ideal Speedup} = \frac{T_s + T_p}{T_s + T_p / N} \quad (4)$$

where T_s is the CPU time for preprocessing and postprocessing portion of the sequential execution, T_p is the CPU time of the main screening program loop, and N is the total number of processors used. $T_s + T_p$ in (4) is the total CPU time for the sequential execution of the program that is measured using one of the DN425 models.

The difference of the maximum and minimum CPU times indicates the loading imbalance in the remote processors. There are many external factors which can impact the imbalance such as, communication delays, concurrent background processes in the remote processors and other random effects. In addition, internal factors regarding the computational requirements for each contingency may affect the load balance as well. For example, some of the parallel generator outages don't require the computation of the compensation vectors, which can reduce the computation time by as much as 50 percent. Also the size of the localized solution for each outage is different, which, in turn, can impact the execution speed. Nevertheless, the actual speedup improvements strongly indicate a very good performance. For instance, with 4 processors, the program achieved a speedup of 2.98.

In Table 3, the results using a faster model as a remote processor are listed. As mentioned before, the DN433 is about 25 percent faster than the DN425 model. We can increase the efficiency and load balance by assigning this faster processor to take the position of the slowest processor in Table 2. The ideal speed up is not computed for this result because it is not meaningful to measure the speedup for processors with varying capabilities. The results in Table 3 indicate a much improved speedup in general. Where two processors are used, 2.25 speedup is achieved. The reason that the speedup is larger than 2 is that one processor is faster and the sequential process CPU time is measured using the slower processor. When four processors are used, a speedup of 3.80 is achieved compared to a speedup of 2.98 with identical DN425 model.

Another observation is that the elapsed time is dominated by the processor with the longest CPU time. The communication delay is not negligible, but does not affect the overall speed substantially. In our test, we also found that the screening analysis for generator outages is usually faster than that of branch outages. As can be seen from these results, the use of distributed processing computing can deduce the computation time for contingency screening applications at a rate almost proportional to the number of processors being used.

Table 1: Test System Summary

Number of lines	2518
Number of generators	665
Number of buses	1983
Number of contingencies	1651

Table 2: CPU Timing with All DN425 Processors

Number of Processors	Elapsed Time	Max CPU time	Min CPU time	Actual Speedup	Ideal Speedup
1	220	217.5	217.5	1.00	1.00
2	133	126.1	94.0	1.72	1.96
3	89	87.8	57.8	2.48	2.90
4	75	73	41.2	2.98	3.81

Table 3: CPU Timing with One DN433 Processor Included

Number of Processors	Elapsed Time	Maximum CPU time	Minimum CPU time	Actual Speedup
2	99	96.8	94.8	2.25
3	79	76.9	57.8	2.82
4	59	57.2	41.5	3.80
5	51	46.8	29.8	4.65

V. CONCLUSIONS

Contingency screening technologies have been the focus of intense research for more than a decade. As a result their efficiency and reliability have substantially improved. Fast contingency screening methods have been developed to screen both branch MW overload and reactive power flow problems. These technologies have served the power industry well, but recent changes threaten to erode the traditionally high level of system security.

In this paper, distributed processing computing is proposed and implemented on UNIX based workstations as a way to screen and process a very large number of contingencies at a very fast pace. We demonstrated how contingency screening computations can be parallelized, and how independent tasks can be distributed into a number of networked processors. The testing results, and numerous other test results not presented here, indicate that distributed processing can reduce, at almost no additional cost, the computation time for contingency screening applications at a rate almost proportional the number of processors being used. We believe that near ideal speedups can be achieved for contingency screening applications with some additional fine-tuning. This conclusion is of significant importance because it makes distributing computing an essential

technology that can assist utility operators maintain a high level of system security. It is anticipated that the standardization of distributed computing will further simplify and facilitate the development of other power system applications.

VI. REFERENCES

1. N. Balu, T. Bertram, A. Bose, V. Brandwajn, G. Cauley, D. Curtice, A. Fouad, L. Fink, M.G. Lauby, B.F. Wollenberg and J. N. Wrubel, 'On-Line Power System Security Analysis,' *Proc. IEEE*, Vol. 80, pp. 262-280, Feb. 1992.
2. V. Brandwajn, and M. G. Lauby, 'Complete Bounding Method for AC Contingency Screening,' *IEEE Trans. Power Systems*, Vol. 4, pp. 724-729, May 1989.
3. Yilang Chen, and Anjan Bose, 'Choosing the Appropriate Boundary for Adaptive Reduction,' *IEEE Trans. on Power Systems*, pp. 745-752, May 1988
4. F. Tinney, V. Brandwajn and S. M. Chan, 'Sparse Vector Methods,' *IEEE Trans. on Power Apparatus Systems*, Vol. PAS-104, 295-301, Feb. 1985.
5. Bacher, G.C. Ejebe, and W.F. Tinney, "Approximate Sparse Vector Techniques for Power Network Solutions," Proceeding of Power Industry Computer Application Conference, Seattle, WA, May 1989.
6. O. Alsac, B. Stott and W.F. Tinney, 'Sparsity-Oriented Compensation Methods for Modified Network Solutions,' *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-102, No. 2, pp. 1050 -1060, May 1983.
7. G. Lauby, T.A. Mikolinnas, and N.D. Reppen, 'Contingency Selection of Branch Outages Causing Voltage Problems,' *IEEE Trans. Power Apparatus Systems*, Vol. PAS-102, pp. 3899-3904, Dec. 1983.
8. M. Peterson, W.F. Tinney, and D.W. Bree, "Iterative Linear AC Power Flow Solution for Fast Approximate Outage Studies," *IEEE Trans. Power Apparatus Systems*, Vol. PAS-91, pp. 2048-2053, Sept/Oct. 1972.
9. Albuyeh, A. Bose, and B. Heath, "Reactive Power Considerations in Automatic Contingency Selection, *IEEE Trans. Power Apparatus Systems*, Vol. PAS101, pp. 107-112, Jan. 1982.
10. F. Tinny and J.M. Bright "Adaptive Reductions for Power Flow Equivalents," *IEEE Trans. on Power Systems*, Vol. 2, pp. 351 -360, May 1987.
11. Bacher and W.F. Tinney, 'Fast local Power Flow Solutions: The Zero Mismatch Approach,' *IEEE Trans. on Power Systems*, Vol. 4, pp. 1345-1354, November 1989.

12. I.S. Duff and J.K. Reid, 'MA27 - A set of Fortran Subroutines for Solving Sparse Symmetric Sets of Linear Equations,' Report AERE R-10533, Harwell Laboratory, Oxfordshire, England, 1979.
13. B. Stott and O. Alsac, 'Fast Decoupled Load Flow,' *IEEE Trans. on Power Apparatus and Systems*, Vol. PAS-93, pp. 859 - 869, May 1974.
14. S. Baase, *Computer Algorithms: Introduction to Design and Analysis*, Addison-Wesley, Massachusetts, 1979.
15. W.R. Stevens, *Unix Network Programming*, Prentice Hall, Englewood Cliffs, New Jersey, 1990.
16. Open Software Foundation, *Distributed Computing Environment — An Overview*, Cambridge, MA, March 1993

VII. BIOGRAPHIES

Shangyou Hao received the B.S. degree from Wuhan Institute of Hydraulic and Electrical Engineering, PRC, in 1982, and MS. and Ph.D. degrees in Electrical Engineering from the Ohio State University in 1984 and 1988, respectively. He worked for PG&E as an independent consultant from 1988 to 1990. He has been with PG&E since 1990, working on the development of various analytical functions.

Alex D. Papalexopoulos received the Electrical and Mechanical Engineering Diploma from the National Technical University of Athens, Greece in 1980, the M.S. and Ph.D degrees in Electrical Engineering from the Georgia Institute of Technology in 1982 and 1985, respectively. Since October 1985, he has been with PG&E working on the development of advanced applications for PG&E's new Energy Management System. Alex is currently responsible for the development and implementation of advanced methodologies and models for operations, operation planning and transmission planning. Alex is also responsible for the development of models to support PG&E's efforts in the regulatory arena and in electric industry restructuring. His primary research interests include applications of large-scale theory to the real-time control of power system, dynamic simulation of power systems and electromagnetic transient analysis. Alex is a senior member of IEEE and a member of Sigma Xi and the Technical Chamber of Greece.

Tie-Mao Peng was born in Hebei, China. He received his B.S. in Electrical Engineering from the North China Institute of Electric Power (China) in 1982, and the M.S. and Ph.D degree in Electric Engineering from Arizona State University in 1988 and 1991 respectively. Since 1991, he has been working as a consultant for Pacific Gas & Electric Company.