Concise Papers

Hierarchical Modeling of Availability in Distributed Systems

Salim Hariri and Hasan Mutlu

Abstract— **Distributed computing systems are attractive due to the potential improvement in availability, fault-tolerance, performance, and resource sharing. Modeling and evaluation of such computing systems is an important step in the design process of distributed systems. In this paper, we present a two-level hierarchical model to analyze the availability of distributed systems. At the higher level (user level), the availability of the tasks (processes) is analyzed using a graph-based approach. At the lower level (component level), detailed Markov models are developed to analyze the component availabilities. These models take into account the hardware/software failures, congestion and collisions in communication links, allocation of resources, and the redundancy level. A systematic approach is developed to apply the two-level hierarchical model to evaluate the availability of the processes and the services provided by a distributed computing environment. This approach is then applied to analyze some of the distributed processes of a real distributed system, Unified Workstation Environment (UWE), that is currently being implemented at AT&T Bell Laboratories.**

Index Terms—**Availability, reliability, task availability, distributed system availability modeling, hierarchical availability modeling, task availability Optimization.**

I. Introduction

Advances in networking and computer technology and software have led to an increased interest in distributed computing systems. Distributed systems consist of several computers (could be a workstation, a mainframe, or a supercomputer) that interact and cooperate to solve one distributed application. These systems are perceived by the users as logical time-shared computing systems and provide an increase in throughput, reliability, fault-tolerance, resource sharing, extensibility, and a cost-effective computing environment. Designing and engineering such systems are difficult and more complex than the ones experienced in centralized computing systems because of the usage of heterogeneous dispersed components. Consequently, performance metrics such as reliability, availability, throughput, delay, and error rate are important parameters to evaluate and optimally engineer the design of cost-effective distributed systems.

This paper addresses the availability issues in a distributed computing environment. The availability of a computing system, $A(t)$, can be defined as the probability that the system will function satisfactorily at a given time (t). Steady state availability is the probability that the system will be available at any random point of time. Asymptotically, for large t, it can be shown that the system availability is numerically the same as steady-state availability. Although we consider steady-state availability in the analysis proposed in this paper, the same techniques can be used to analyze other dependability measures. Reliability/Availability analysis of computer systems has been an active area of research for some time and several techniques have been proposed. A survey of modeling techniques and tools can be found in [10], [13], [14]. These methods can be broadly categorized as follows: 1) combinatorics methods: a) probabilistic network graph or reliability block diagram [1], [8], [9], [10], [17], b) fault tree [5], [10]; 2) Mathematical Models: a) Markov chains [10], [15], b) Petri nets [3]; and 3) Simulation Methods [4], [7].

There are several hierarchical techniques that have been proposed to analyze availability using different modeling techniques [15], [16]. These methods use either fault tree, series parallel system or Markov chain to model each level of the hierarchy. In this paper, we present a hierarchical approach that uses a graph based approach at the system level and uses Markovian technique to analyze component availability. The use of Markovian technique to model component availabilities enables us to take into consideration dependent failures among components, software failures, and other performance constraints. At the component level, the number of states for most system components is relatively small and closed form solutions are obtainable. Furthermore, other dependability and performance measures can be modeled using Markovian chains. At the system level, we use a network graph to describe the interactions and the connectivities among the components of the system. The nodes of this probabilistic graph represent the components of the system and the edges correspond to the logical and/or physical connections among the components. Consequently, all the algorithms developed to determine the availability of communication networks can be used to evaluate the overall system availability.

The organization of this paper is as follows. In Section II, we present a hierarchical model to analyze the availability of distributed computing tasks. In Section III, we apply our approach to analyze the availability of the services provided by the Unified Workstation Environment (UWE) with respect to different user sites. Finally, we present a summary and concluding remarks on how to extend this work to analyze other performance metrics.

II. Hierarchical Availability Modeling

A. Modeling Availability at the System Level

At this level, we address the problem of modeling the availability of a task (process) in a distributed computing environment. Process availability can be defined as the probability that the process is functioning properly at a given time. Modeling process availability in a distributed computing environment is complicated because process execution involves the cooperation and interaction of several types of resources. For example, Fig. 1 shows an example of a distributed system in which a process, say P, would require accessing three processes running on three separate computers.

Such a process could be an application program to transfer a fund from one account to another and interact with three resources (software modules, computers, devices, etc.): one software module (**request_handler**) to manage and supervise the processing of user requests, second process (program such as **fund_transfer**) is needed to transfer fund from one account to another in an atomic manner, and the third resource is a database to store **user_accounts.** To increase the availability of executing processes in this environment, double redundancy is introduced as shown in Fig. 1. Process P

Manuscript received January 1992; revised February 1993, June 1993, December 1993, and September 1994.

S. Hariri is with Department of Electrical and Computer Engineering Syracuse University, Syracuse, NY 13244 USA (e-mail: hariri@cat.syr.edu).
H. B. Mutlu is with Mutek, Inc.

IEEE Log Number 9407723.

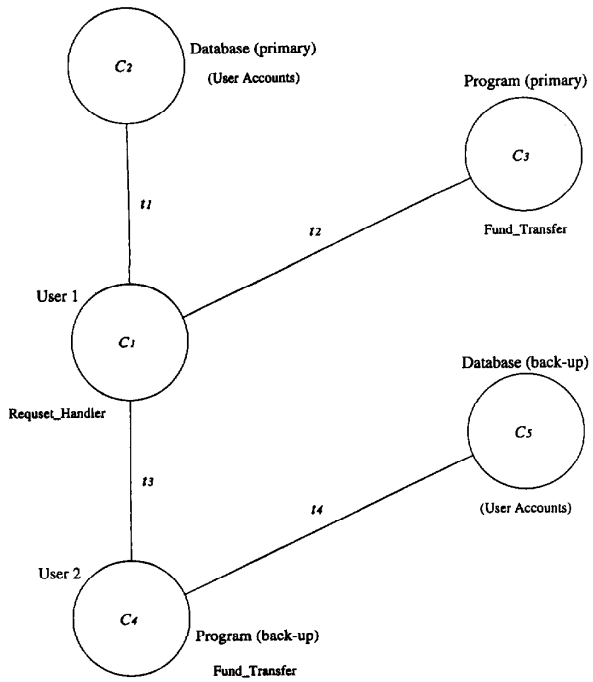

Fig. 1. Process Execution in a Distributed (Redundant) Environment.

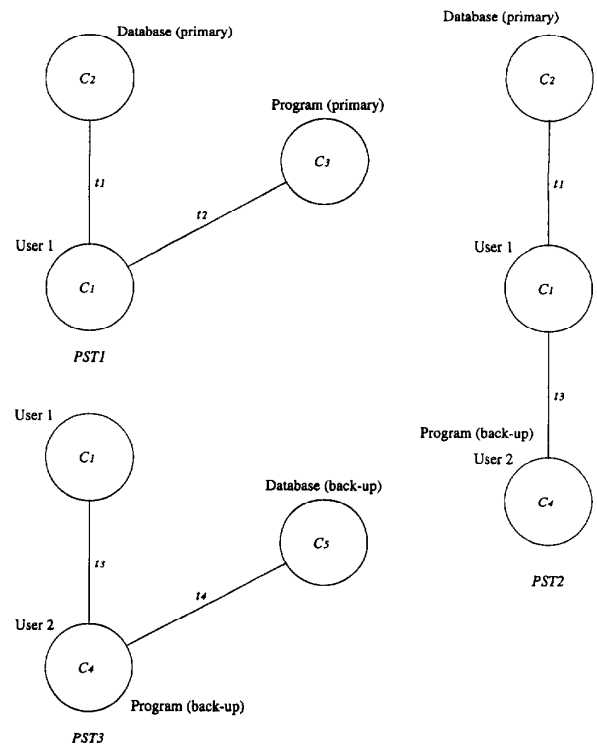

Fig. 2. Process spanning trees.

that is invoked from site c_1 can be processed successfully if the computers c_1, c_2, and c_3, the communication links t_1 and t_2, and software modules used by P are all running properly. Furthermore, if computer c_3 or link t_2 is down, it is still possible to run P using the copy of the **fund_transfer** program available at computer c_4.

Process availability depends on the 1) availabilities of computers, 2) availabilities of software modules, 3) availabilities of communications links, 4) collision (delays), 5) allocation of resources, and 6) redundancy level. In our proposed hierarchical model, the computer availability module takes into consideration the failure rates caused by both hardware and software. Furthermore, the link availability model considers failures caused by hardware as well as those caused by excessive delays (item 4 mentioned above). The main idea behind lumping items 1 and 2, and 3 and 4 is to capture the effects of the parameters mentioned above by a tree (referred to as process spanning tree PST that was introduced first in [11]). The allocation of resources and their redundancy level are modeled by the existence of several PSTs that can be used to run process P with respect to user 1 as shown in Fig. 2. Thus, process P is available if at least one PST is in operational state (all nodes and links of that tree are functioning properly). Hence, the availability of process P can be evaluated by determining the probability that at least one PST is operational. That is,

$$A(P, \text{user 1}) = P_r(\text{at least one PST is up}) = P_r\left(\cup_{i=1}^{i=3} PST_i\right)$$

where $A(P, \text{user 1})$ denotes the availability of process P with respect to user 1.

One approach to evaluate the probability described by the above equation is to modify the set of trees (PST's) into another equivalent set of mutually exclusive PST's; this approach is similar to the sum-of-products (SOP) approach used in fault-tree and reliability block diagram. For example, the events[1] corresponding to the union of the three PST's associated with process P can be decomposed into the following three events: first event occurs when PST_1 is up, second event occurs when PST_2 is up and PST_1 is down, and third event occurs when PST_3 is up and both PST_1 and PST_2 are down. Hence, the availability of process P can be written as

$$A(P, user1) = P_r(PST_1) + P_r\left(PST_2 \cap \overline{PST_1}\right) + P_r\left(PST_3 \cap \overline{PST_1} \cap \overline{PST_2}\right)$$

where $\overline{PST_i}$ denotes that PST_i is in the failure state.

The term corresponding to the first event is $A_{c_1}A_{c_2}A_{c_3}A_{t_1}A_{t_2}$, where A_i denotes the availability of component i. In the second event, PST_2 is up which results in having all the components of PST_2 being operational, and PST_1 is down which occurs when either t_2 or c_3 has failed. Hence, the term corresponding to the second event is $A_{c_1}A_{c_2}A_{c_4}A_{t_1}A_{t_3}(1 - A_{t_2}A_{c_3})$. In a similar manner, the term corresponding to the third event is $A_{c_1}A_{c_4}A_{c_5}A_{t_3}A_{t_4}(1 - A_{c_2}A_{t_1})$.

The process availability with respect to a user at site c_1 is the sum of the terms associated with these three events. That is,

$$A(P, user1) = A_{c_1}A_{c_2}A_{c_3}A_{t_1}A_{t_2} + A_{c_1}A_{c_2}A_{c_4}A_{t_1}A_{t_3} \cdot (1 - A_{t_2}A_{c_3}) + A_{c_1}A_{c_4}A_{c_5}A_{t_3}A_{t_4}(1 - A_{c_2}A_{t_1}).$$

The process availability evaluated above was with respect to users at site c_1. Similar analysis can be applied to evaluate process availability with respect to users at site c_4. A detailed analysis of evaluation algorithms for similar reliability/availability measures can be found in [11, 12].

[1] PST_i is used to denote process spanning tree i as well as the event that PST_i is in the operational state.

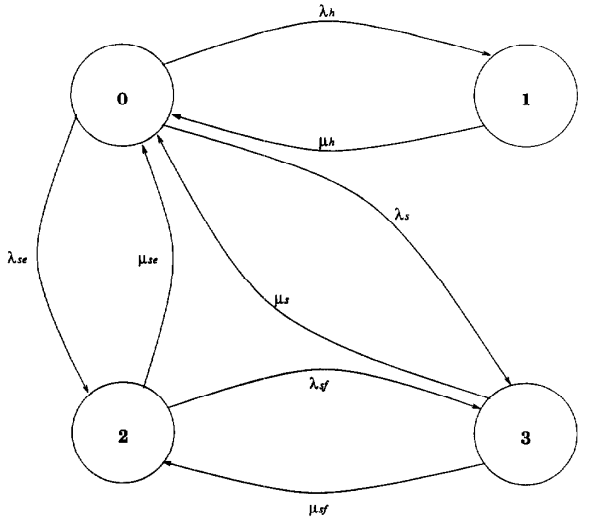

Fig. 3. Transition diagram for the four-state Markovian model.

B. Modeling Availability at the Component Level

The availability metrics, developed in the previous section, model the availability perceived by user's processes and are described in terms of node and link availabilities. In this section, we use Markovian models to evaluate the component availabilities. Other methods such as SHARPE [15], Extended Petri Nets [5], Simulation [4] can also be used to evaluate the component availability. For example, Fig. 3 shows a four-state Markovian model to evaluate the availability of a host. This model takes into consideration hardware and software failures as well as transient failures. The states of the component are as follows:

state 0—correct operation state. The state in which the host operates correctly and has no software failure.

state 1—hardware failure state. The state in which the host failure occurs due to a hardware or environmental problem.[2]

state 2—transient software failure state. The state in which the host failure occurs due to a transient software failure.[3]

state 3—permanent software failure state. The state in which the host failure occurs due to a permanent software failure.

Notation and Definitions

λ_h hardware failure rate from the healthy state.

λ_s (λ_{se}) permanent (transient) software failure rate from the healthy state.

λ_{sf} permanent software failure rate from the transient software failure state.

μ_h hardware failure repair rate from the hardware failure state.

μ_s software failure repair rate from the permanent software failure state.

μ_{se} fast software recovery rate from the transient software failure state.

[2]Typical environmental problems are power outages or errors caused by improper operation of the component.

[3]This failure can be caused by bugs in software or intermittent hardware faults.

μ_{sf} slow software recovery rate from the permanent software failure state.

The transition from state 0 to state 1 is caused by a hardware or environmental problem, while the transition from state 1 to state 0 is caused by removing the failure. Likewise the transition from state 0 to state 2 is caused by a transient software failure, while the transition from state 2 to state 0 occurs after recovering from the transient software failure. The occurrence of permanent software failure in state 2 will cause the system to go to state 3. Partial restoration of system software can lead to a transition to state 2 from which a fast recovery can move the system back to its operational state. Finally, the transition from state 0 to state 3 is caused by a permanent software failure, whereas the transition from state 3 to state 0 takes place by removing the permanent software failure which requires complete reset and reload of the host's system software. The transition diagram covers all possible states and transitions that a component is likely to experience.

By analyzing the system for the steady-state availability, we can write the host availability as shown at the bottom of the page.

III. Evaluation of Tasks Availability in Distributed Systems

In this section we analyze the availability of the services and tasks provided by a distributed computing environment with respect to a user site. To reduce the complexity of this analysis, we develop a two-step procedure. In the first step, we group the resources of a given distributed system according to their access requirements. In the second step, we apply our modeling technique presented in Section II to evaluate the availability of UWE services.

A. Classification of Distributed System Resources

The type of activities and processes in distributed systems will be outlined by analyzing the activities of a general purpose distributed system currently being developed at AT&T Bell Laboratories. The main objective of the UWE project is to investigate seamless distributed computing environments. In this environment, a user would have a single account with a single physical login directory across all machines, called community. Also, the user view the system as one logical machine in which all its resources are accessible to the user regardless of his/her physical location. Users are identified within the community with **user_id** and **community_name!user_id** to those outside the community. The Remote File Sharing (RFS) and Remote Execution (REXEC) facilities have provided the capability of sharing file resources over the network and executing arbitrary commands on any machine, respectively. Fig. 4 shows one implementation of the UWE paradigm which consists of 6 3B2s: 2 310s with 4 Mbytes of memory each, 2 400s with 4 Mbytes each, and 2 600 with 8 Mbytes of memory. The communication network implements both Datakit and Ethernet. The UWE services can be characterized as follows: 1) **Login Process**; 2) **Command Execution**; The commands that are frequently used can be divided into four groups: 1) Simple: ls, date, mv, rm, cd, pr, cat; 2) Complex: find, cpio, nroff, spell, cp, cc; 3) Editor commands; and 4) Miscellaneous: rmail, smail, post. The complexity of these commands depends also on the number and type of parameters used in these commands. Also, their execution times vary substantially depending on whether or not the commands involve local or remote access to their needed resources.

The execution of UWE processes depends on the services provided by a set of main resources. Analyzing the availability of the services

$$A = \frac{\mu_h(\lambda_{sf}\mu_s + \mu_s\mu_{se} + \mu_{se}\mu_{sf})}{(\lambda_h + \mu_h)(\lambda_{sf}\mu_s + \mu_s\mu_{se} + \mu_{se}\mu_{sf}) + \mu_h(\lambda_s\mu_{sf} + \lambda_{se}\mu_s + \lambda_{se}\mu_{sf} + \lambda_s\lambda_{sf} + \lambda_{se}\lambda_{sf} + \lambda_s\mu_{se})}.$$

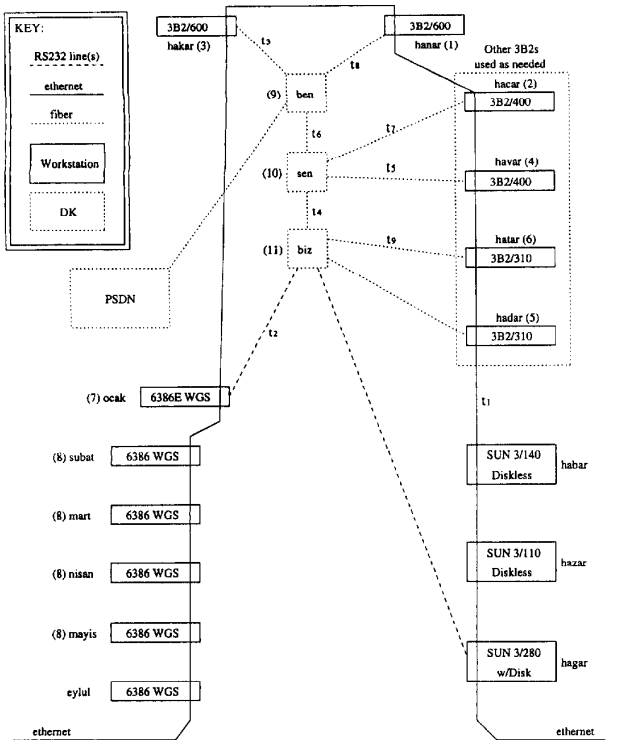

Fig. 4. Topology of current UWE.

TABLE I
DISTRIBUTION OF UWE MAIN RESOURCES

RESOURCE NAME	COMMANDS (PROCESSES)	LOCATION(S)
rfs_name_server	rfadmin, nsquery, mount	hanar, hakar
usr_files	cp, mv, cat, ls, pwd, troff	hanar, hakar
login_server	login process	hanar, hakar
/usr/mail	rmail, mail, mailx, post	hanar
/usr/spool/uucppublic	uucp into community	hanar
/usr/spool/uucp	uucp outside community	hanar
wwb	spell, wwb	hakar
manpages	man	hakar
news	news	hakar
rdssrc	source directories	hakar
print_server	prt, prtfmt	hanar
exptools	lf, sam, xcip, xproof, gmacs,	hanar
dmd software	layers	hanar
630 software	layers	hakar
post database	pq, post -w	tr/sari/post4
linus	linus	tr/mor/yesil
rnd tools	lprint, slides, rdsae, face, ...	sy/yaz/beyaz

TABLE II
RESOURCE SETS

RESOURCE SET (RS)	HOST(S)	RESOURCE NAMES
RS_1	hanar	mail, uucppub, dmd, exptools, printer_server
RS_2	hakar	man, wwb, news, rdssrc, 630
RS_3	hanar, hakar	rfs_name_server, login_server,
RS_4	hanar, hakar	user_files
RS_5	remote hosts	post database, linus, rnd tools

provided by a distributed system can be described in terms of the availability of accessing these resources. For example, a print command executed in UWE involves the use of two resources: the user_file system and the print_server. First, the user_file to be printed is copied so it can be processed locally at the machine which has originated the command by a distributed printer spooler (UNISON). Once the file has been processed by UNISON and ready to be printed, the print_server is called to format and then route that file to the required printer. Also, the print_server sends the user information about the status of the file to be printed. Hence, the print command can be executed successfully if these two resources are accessible and their associated processes can run without malfunctioning until completion. In other words, the availability of the print command can be described in terms of the availabilities of accessing these two resources simultaneously, i.e.,

$$A(\text{print}) = A(\text{user\_files and print\_server}).$$

Table I shows the UWE main resources and the hosts that store and/or provide their services. Since the software failures are incorporated in the computer availability, then the availability of a resource running on a host, say x_i, with respect to a user at site, say x_j, can be measured by the probability of accessing host x_i from host x_j successfully. This means that the availability of the resources of a distributed system with respect to site x_j can be described in terms of the probability of accessing the hosts running these resources from site x_j without any failures. Consequently, this can be used to reduce the analysis to a few Resource Sets (RS) such that the resources in each set have the same access requirement. For example, the resources (**uucppub, uucp, print_server, and mail**) are stored at **hanar** and therefore belong to one set, say RS_1. If all the resources shown in Table I are grouped into sets based on their access requirements, the analysis can be reduced to studying the availability of accessing the hosts of the five resource sets shown in Table II. Although RS_3 and RS_4 (shown in Table II) have the same host set (**hanar** and **hakar**), we did not combine their resources in one set and that can be explained in the following. In RS_3, the primary login-server runs on **hanar** while its back-up copy runs on **hakar** when **hanar** is not available. This access requirement is different from that of the resources of RS_4. In RS_4, the user files are partitioned evenly over two computers (users 2, 3, and 5 are allocated to **hanar** while users 6, 7, and 8 are allocated to **hakar**) as shown in Table I. If **hanar** is not available, users 2, 3, and 5 can not access their files, whereas users 6, 7, and can access their files stored at **hakar**.

B. Evaluating UWE's Availability Metrics

To evaluate the availability of a resource with respect to one site, say **ocak** workstation, the set of process spanning trees (PSTs) should be determined. Then, the availability of having at least one PST operational can be evaluated using the approach presented in Sections II and III. In what follows, we will determine the availabilities of three resources such as print_server, user_files, and a print process with and without redundant resources with respect to a user located at **ocak** workstation (see Fig. 4).

Case 1) Without Redundant Resources: **A) Access Availability of the print_server (A (print_server, ocak))**

The availability of the print_server service can be evaluated in a similar method to that described in Section II and is shown below as

$$A(\text{print\_server}, ocak) = A_{ocak}A_{l1}A_{hanar} + \overline{A}_{l1}A_{ocak}A_{l2}A_{biz}A_{l4}A_{sen}A_{l6}A_{ben}A_{l8}A_{hanar}$$

where A_i ($\overline{A}_i$) denotes the availability (unavailability) of component i. The unavailability of a component is equal to ($1-A_i$).

B) Access Availability of the user files (A (user_files, ocak))

In the current implementation of UWE, all user files are evenly distributed across two hosts: **hanar** and **hakar** and thus the probability of storing the files of a user at one of these two computers is 0.5. Consequently, the PSTs must be determined for both scenarios (user files are stored at either **hanar** or at **hakar**). The availability of accessing a file is shown below:

$$A(\text{user\_files}, ocak) = 0.5(A_{ocak}A_{t1}A_{hanar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t8}A_{hanar}) + 0.5(A_{ocak}A_{t1}A_{hanar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t3}A_{hakar}).$$

C) Availability of the print process (A (print_process, ocak))

The print_process can be defined as a sequence of two processes: accessing the file_system and accessing the print_server. The availability of the print command is therefore the probability of achieving these two processes successfully at a given time. That is,

$$A(\text{print}, ocak) = A(\text{ user\_files and print\_}server, ocak).$$

The availability of the print process is shown below:

$$A(\text{print}, ocak) = 0.5\big(A_{ocak}A_{t1}A_{hanar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t8}A_{hanar}) + 0.5(A_{ocak}A_{t1}A_{hakar}A_{hanar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t3}A_{hakar}A_{t8}A_{hanar}\big).$$

Case 2) With Redundant Resources: The inherent redundant resources in UWE can be used to increase availability, fault-tolerance, and minimize the average response time. For example, if we use **hacar** workstation as a replica of **hanar**, then **hacar** will execute all processes routed to **hanar** when it is down or heavily loaded. Similarly, **havar** can execute **hakar**'s tasks when it is down. In what follows, we will reevaluate the availabilities of the resources studied in Case 1 after introducing double redundancy to the resources available at **hanar** and **hakar** computers.

A) Access Availability of the print_server (A (print_server, ocak))

The availability of accessing this resource is shown below:

$$A(\text{print\_server}, ocak) = A_{ocak}A_{t1}A_{hanar} + \overline{A}_{hanar}A_{ocak}A_{t1}A_{hacar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t7}A_{hacar} + \overline{A}_{t1} \cdot (1 - A_{t7}A_{hacar})A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t8}A_{hanar}.$$

B) Access Availability of user files (A (user_files, ocak))

The availability of accessing the user files from **ocak** is shown below:

$$A(\text{user\_files}, ocak) = 0.5\big[A_{ocak}A_{t1}A_{hanar} + \overline{A}_{hanar}A_{ocak}A_{t1}A_{hacar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t7}A_{hacar} + \overline{A}_{t1}(1 - A_{t7}A_{hacar})A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t8}A_{hanar}\big] + 0.5\big[A_{ocak}A_{t1}A_{hakar} + \overline{A}_{hakar}A_{ocak}A_{t1}A_{havar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t5}A_{havar} + \overline{A}_{t1}(1 - A_{t5}A_{havar})A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t3}A_{hakar}\big].$$

C) Availability of the print process (A (print, ocak))

$$A(\text{ print}, ocak) = 0.5\big[A_{ocak}A_{t1}A_{hanar} + \overline{A}_{hanar}A_{ocak}A_{t1}A_{hacar} + \overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t7}A_{hacar} + \overline{A}_{t1}(1 - A_{t7}A_{hacar})A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t8}A_{hanar}\big] + 0.5\big(A_{ocak}A_{t1}A_{hakar} + \overline{A}_{hakar}A_{ocak}A_{t1}A_{havar}\big)(A_{hanar} + \overline{A}_{hanar}A_{hacar}) + 0.5A_{t7}A_{hacar}\big[\overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t5}A_{havar}\big] + 0.5A_{t8}A_{hanar}(1 - A_{t7}A_{hacar})\big[\overline{A}_{t1}A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t5}A_{havar}\big] + 0.5A_{t8}A_{hanar}(1 - A_{t7}A_{hacar})\big[\overline{A}_{t1}(1 - A_{t5}A_{havar})A_{ocak}A_{t2}A_{biz}A_{t4}A_{sen}A_{t6}A_{ben}A_{t3}A_{hakar}\big].$$

TABLE III
AVAILABILITY IMPROVEMENT FOR A(user_files, site) IN CURRENT UWE

SITE	AVAILABILITY	AVAILABILITY WITH REDUNDANCY	AVAILABILITY IMPROVEMENT(%)
hanar	0.99448	0.99637	34.17
hacar	0.98898	0.99637	67.05
hakar	0.99197	0.99574	46.93
havar	0.98898	0.99574	61.32
hadar	0.98794	0.99460	55.23
hatar	0.98794	0.99460	55.23
ocak	0.98794	0.99460	55.23
uow*	0.96274	0.96708	11.65
T@ben**	0.91341	0.95805	51.55
T@sen	0.87715	0.95805	65.85
T@biz	0.84233	0.92001	49.27
E(A)	0.95671		
E(A)		0.97920	

* (user owned workstations: subat, mart, nisan, and mayis).

** (terminal on Datakit node ben).

The improvement in availability because of introducing redundancy to the resources of a distributed system can be used as a cost measure for the availability improvement. This measure can be defined as the ratio between the availability increase to the maximal increase (which occurs when availability becomes one), i.e.,

$$\text{AIF} = \frac{A(\text{with redundancy}) - A(\text{without redundancy})}{1 - A(\text{without redundancy})}.$$

The availabilities of the resources discussed previously (**print_server, user_files,** and **print_process**) have been evaluated along with their associated AIF's. Tables III–V show the availabilities of these resources with respect to each user site, and the availability improvement factors (AIF's) obtained because of introducing double redundancy to the UWE's resources. In this analysis, we calculate the component availabilities based on data collected by the network management system during a period of six months. This system collects the mean-time-to-failure (MTTF) and mean-time-to-repair (MTTR) for each component in UWE. Based on this data, the availabilities of the UWE components are as follows: Host Availability = 0.998; Node Availability = 0.99; Link Availability = 0.97 and; Ethernet Availability = 0.97. The complexity of algorithms to generate the PST's that satisfy certain properties can be mapped into a recognition problem and have been shown to be NP-hard problem [2]. However, for most practical distributed systems, e.g., the UWE system, the PSTs can be generated easily from the network topology and the allocation of the system resources.

Our analysis can be used to tune the availability of the system by experimenting with different configurations. For example, Fig. 5 shows another topology for the UWE in which **hanar** is connected to Datakit node **sen** instead of **ben**, and **hacar** to node **ben** instead of **sen.** Tables VI–VIII show the result of the availability analysis of the same resources in this new topology. If we assume the probability of accessing the resources are uniformly distributed to all sites and equal to $\frac{1}{\text{number of sites}}$, the expected availabilities of accessing user_files and print_server, and the print process with respect to all sites are shown in Table IV. It is clear from Table IV that the topology shown in Fig. 5 is more reliable than the one shown in Fig. 4. The low availability of the topology shown in Fig. 4 caused by the fact that the failure of node **ben** will isolate both **hanar** and **hakar** computers

TABLE IV
AVAILABILITY IMPROVEMENT FOR A(print_server, site) IN CURRENT UWE

SITE	AVAILABILITY	AVAILABILITY WITH REDUNDANCY	AVAILABILITY IMPROVEMENT(%)
hanar	0.99448	0.99637	34.17
hacar	0.99086	0.99637	60.25
hakar	0.99448	0.99637	34.17
havar	0.99086	0.99637	60.25
hadar	0.98980	0.99460	47.03
hatar	0.98980	0.99460	47.03
ocak	0.98980	0.99460	47.03
uow	0.96419	0.96708	08.08
T@ben	0.92870	0.95805	41.16
T@sen	0.89183	0.95805	61.22
T@biz	0.85642	0.92001	44.29
E(A)	0.96193		
E(A)		0.97931	

TABLE V
AVAILABILITY IMPROVEMENT FOR A(print_process, site) IN CURRENT UWE

SITE	AVAILABILITY	AVAILABILITY WITH REDUNDANCY	AVAILABILITY IMPROVEMENT(%)
hanar	0.99700	0.99700	00.00
hacar	0.99086	0.99700	67.17
hakar	0.99197	0.99574	46.93
havar	0.99086	0.99574	53.34
hadar	0.98980	0.99460	47.03
hatar	0.99052	0.99460	43.04
ocak	0.98980	0.99460	47.02
uow	0.96419	0.96708	08.08
T@ben	0.92870	0.95805	41.16
T@sen	0.89183	0.95805	61.22
T@biz	0.85642	0.92001	44.29
E(A)	0.96200		
E(A)		0.97931	

TABLE VI
AVAILABILITY IMPROVEMENT FOR A(user_files, site) IN REVISED UWE

SITE	AVAILABILITY	AVAILABILITY WITH REDUNDANCY	AVAILABILITY IMPROVEMENT (%)
hanar	0.99095	0.99338	26.84
hacar	0.98236	0.99338	62.47
hakar	0.98485	0.99212	47.98
havar	0.98345	0.99212	52.39
hadar	0.98129	0.98923	42.44
hatar	0.98129	0.98923	42.44
ocak	0.98129	0.98923	42.44
uow	0.92966	0.93525	07.94
T@ben	0.81180	0.91785	56.35
T@sen	0.84536	0.91785	46.88
T@biz	0.77957	0.84642	30.33
E(A)	0.93199		
E(A)		0.95964	

TABLE VII
AVAILABILITY IMPROVEMENT FOR A(print_server, site) IN REVISED UWE

SITE	AVAILABILITY	AVAILABILITY WITH REDUNDANCY	AVAILABILITY IMPROVEMENT(%)
hanar	0.99393	0.99637	40.15
hacar	0.99141	0.99637	57.70
hakar	0.99393	0.99637	40.15
havar	0.99141	0.99637	57.70
hadar	0.99033	0.99460	44.12
hatar	0.99033	0.99460	44.12
ocak	0.99033	0.99460	44.12
uow	0.96419	0.96708	08.08
T@ben	0.91026	0.95805	53.25
T@sen	0.91026	0.95805	53.25
T@biz	0.87412	0.92001	36.46
E(A)	0.96368		
E(A)		0.97931	

where most of UWE's resources are stored. However, because it was assumed in our analysis that all communication links have identical availabilities (i.e., 0.97), the two configurations have the same availability when double redundancy is used.

TABLE VIII
AVAILABILITY IMPROVEMENT FOR A(print_process, site) IN REVISED UWE

SITE	AVAILABILITY	AVAILABILITY WITH REDUNDANCY	AVAILABILITY IMPROVEMENT(%)
hanar	0.99700	0.99700	00.00
hacar	0.99086	0.99700	67.17
hakar	0.99086	0.99574	53.34
havar	0.99197	0.99574	46.93
hadar	0.99086	0.99460	40.89
hatar	0.99086	0.99460	40.89
ocak	0.99086	0.99460	40.89
uow	0.96419	0.96708	08.08
T@ben	0.89183	0.95805	61.22
T@sen	0.92870	0.95805	41.16
T@biz	0.89183	0.92001	26.06
E(A)	0.96544		
E(A)		0.97931	

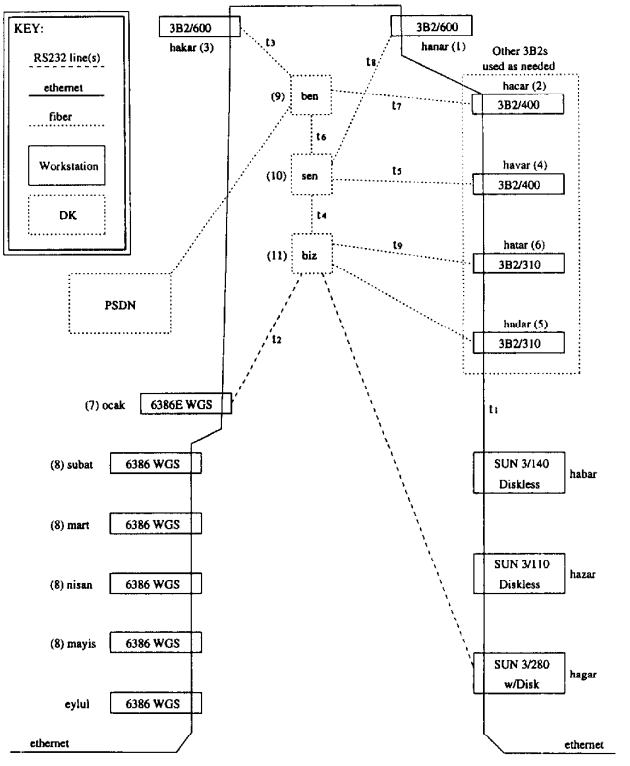

Fig. 5. Revised topology for the UWE.

IV. CONCLUSIONS

In this paper, we present a two-level hierarchical approach to model availability in distributed systems. At the system level, we use graph-based approach, which is suitable to analyze availability of distributed programs, and use Markovian technique to analyze component availability. We show how to apply this approach to analyze availability of distributed applications, programs in a real distributed computing environment. The presented approach provides not only the one-index representation of critical component availability measures, but also assists the process of designing, and engineering the overall distributed computing environment. The availability approach presented in this paper can also be used to identify bottlenecks and critical components; determine the sensitivity of system availability to hardware and software failures; and optimize the process of improving the availability of a distributed computing environment; In addition, since the failure and repair rates can be calculated for each component in the distributed computing environment,

the reliability, maintainability, and other time-dependent measures can also be computed using the approach presented in this paper. Furthermore, the links and the nodes of process spanning trees can be labeled in a manner such that the effect of some other performance metrics (e.g., delay, throughput) can be modeled. Consequently, the same techniques can be adopted to evaluate these combined measures.

ACKNOWLEDGMENT

The authors would like to thank D. T. Klein and W.-M. Chen for their valuable contributions on modeling and analysis of UWE. The prototype UWE configuration and resource allocation in the environment were also provided by D. T. Klein.

REFERENCES

[1] K. K. Aggarwal and S. Rai "Reliability evaluation in computer communication networks," *IEEE Trans. Reliability*, vol. R-30, pp. 32–35, Apr. 1981.

[2] M. O. Ball, "Computational complexity of network reliability analysis: An overview," *IEEE Trans. Reliability*, vol. R-35, pp. 230–239, Aug. 1986.

[3] K. Barkaoui, G. Florin, C. Fraize, B. Lemaire, and S. Natkin, "Reliability analysis of non repairable systems using stochastic Petri nets," in *Proc. 18th Int. Symp. Fault-Tolerant Comput.*, June 1988, pp. 90–95.

[4] E. Conway and A. Goyal, "Monte Carlo simulation of computer system availability/reliability models," in *Proc. 17th Int. Symp. Fault-Tolerant Comput.*, June 1987.

[5] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola, "Extended stochastic Petri nets: Applications and analysis," in *Performance '84*, E. Gelenbe, Ed. Amsterdam, The Netherlands: North-Holland, 1984, pp. 507–519.

[6] J. B. Dugan, S. Babuso, and M. Boyd, "Dynamic fault-tree models for fault-tolerant computer systems," *IEEE Trans. Reliability*, vol. R-41, pp. 363 377, Sept. 1992.

[7] G. Fishman, "System reliability: Estimation, sensitivity and parameter errors," in *Computer Performance and Reliability*, Iazeolla, P. Coutois, and O. Boxma, Eds. New York: North-Holland, 1988.

[8] S. Hariri, C. S. Raghavendra, "SYREL: A symbolic reliability algorithm based on path and cutset methods," *IEEE Trans. Comput.*, vol. C-36, pp. 1224–1232, Oct. 1987.

[9] C. Hwang, F. Tillman, and M. Lee, "System reliability evaluation techniques for complex large systems—A review," *IEEE Trans. Reliability*, vol. R-30, pp. 411–423, Dec. 1981.

[10] A. Johnson and M. Malek, "Survey of software tools for evaluating reliability, availability and serviceability," in *ACM Comput. Surv.*, vol. 20, Dec. 1988.

[11] V. K. P. Kumar, S. Hariri, and C. S. Raghavendra, "Distributed program reliability analysis," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 42–50, Jan. 1986.

[12] C. S. Raghavendra, V. K. P. Kumar, and S. Hariri, "Reliability Analysis in Distributed Systems," *IEEE Trans. Comput.*, vol. 37, pp. 352–358, Mar. 1988.

[13] S. Rai and D. P. Agrawal, "Advances in distributed system reliability," Tutorial Text, IEEE Computer Society Press, 1990.

[14] A. Reibman and M. Veeraraghavan, "Reliability modeling: An overview for system designers," *IEEE Comput.*, pp. 49–57, Apr. 1991.

[15] S. Sahner and K. Trivedi, "A hierarchical, combinatorial-markov method of solving complex reliability models," in *Proc. 1986 Fall Joint Comput. Conf.*, pp. 817–825, Nov. 1986.

[16] ——, "Reliability modeling using SHARPE," *IEEE Trans. Reliability*, vol. R-36, pp. 186–193, June 1987.

[17] A. Satyanarayana and M. Chang, "Network reliability and factoring theorem," *Networks*, vol. 13, pp. 107–120, 1983.

[18] J. Toerrey, "A pruned tree approach to reliability computation," *IEEE Trans. Reliability*, vol. R-32, pp. 170–174, June 1983.

IEEE COPYRIGHT FORM

The IEEE has developed this form with great care and with the best interests of its members and contributing authors in mind. Therefore, in order to maintain uniform treatment among all contributors, other forms may not be substituted for this form, nor may any wording of this form be changed. This form is intended for original, previously unpublished material submitted to IEEE publications. This form, when completed, must accompany any such material in order to be published by IEEE. Please read it carefully and keep a copy of it for your files.

TITLE OF PAPER (hereinafter, "the work"):

AUTHOR(S):

PUBLICATION TITLE/DATE:

PART A — COPYRIGHT TRANSFER FORM

(U.S. Government employees whose work is not subject to U.S. copyright should so certify by signing Part B overleaf. Authors of works subject to Crown Copyright should sign Part C overleaf.)

The undersigned hereby assigns all copyright rights in and to the above work to The Institute of Electrical and Electronics Engineers, Inc. (the "IEEE"). The undersigned hereby represents and warrants that the work is original and that he/she is the author of the work, except possibly for material such as text passages, figures, and data that clearly identify the original source, with permission notices from the copyright owners where required. The undersigned represents that he/she has the power and authority to make and execute this assignment.

In return for these rights, the IEEE recognizes the retained rights noted in Items 1 and 4 below, and grants to the above authors and employers for whom the work may have been performed a royalty-free license to use the material as noted in Items 2 and 3. Item 5 stipulates that authors and employers must seek permission to republish in cases not covered by Items 2, 3, and 4.

1. Employers (or authors) retain all proprietary rights in any process, procedure, or article of manufacture described in the work.

2. Authors/employers may reproduce or authorize others to reproduce the above work, material extracted verbatim from the above work, or derivative works for the author's personal use or for company use provided that the source and the IEEE copyright notice are indicated, that the copies are not used in any way that implies IEEE endorsement of a product or service of an employer, and that the copies themselves are not offered for sale. (See "Author/Employer Rights".)

3. Authors/employers may make limited distribution of all or portions of the above work prior to publication if they inform the IEEE of the nature and extent of such limited distribution prior thereto.

4. In the case of work performed under a U.S. Government contract or grant, IEEE recognizes that the U.S. Government has royalty-free permission to reproduce all or portions of the above work, and to authorize others to do so, for official U.S. Government purposes only, if the contract/grant so requires. (Appropriate documentation may be attached, but IEEE's Copyright Form MUST BE SIGNED. See "U.S. Government Employees/U.S. Government Contract Work".)

5. For all circumstances not covered by Items 2, 3, and 4, authors/employers must request permission from the IEEE Copyrights Office to reproduce or authorize the reproduction of the work or material extracted verbatim from the work, including figures and tables.

Please see notes on "IEEE Obligations" as copyright holder.

In the event the above work is not accepted and published by the IEEE or is withdrawn by the author(s) before acceptance by the IEEE, this agreement becomes null and void.

AUTHORIZED SIGNATURE (or if joint work, as AGENT for all authors)	TITLE
EMPLOYER FOR WHOM WORK WAS PERFORMED	DATE

JOINT AUTHORSHIP

For jointly authored works, all the joint authors should sign, or one of the authors should sign as an authorized agent for the others. In the case of multiple authorship where one or more authors are Government employees but at least one author is not, the non-Government author should sign Part A of this copyright transfer form.

PLEASE DIRECT ALL QUESTIONS ABOUT IEEE COPYRIGHT POLICY OR THIS FORM TO: Manager, Rights and Permissions, IEEE Service Center, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331. Telephone: (908) 562-3966.

149-00-0001
Rev. 7/93

PART B — U.S. GOVERNMENT EMPLOYEE CERTIFICATION

Authors who are U.S. Government employees are not required to sign Part A of the IEEE Copyright Form, but any coauthors outside the U.S. Government are required to sign Part A (see JOINT AUTHORSHIP overleaf). Authors whose work was performed under a U.S. Government contract or grant, but who are not Government employees, are required to sign Part A of this form. (Note: If your work was performed under Government contract but you are not a Government employee, sign transfer form above (Part A) and see Item 4.)

This will certify that all authors of the work are employees of the U.S. Government and performed this work as part of their official duties and that the work is therefore not subject to U.S. copyright protection.

AUTHORIZED SIGNATURE

TITLE

NAME OF GOVERNMENT ORGANIZATION

DATE

PART C — CROWN COPYRIGHT

Authors who are employees of the British Government (or a British Commonwealth Government) and whose works are subject to Crown Copyright may sign Part C. IEEE recognizes and will honor Crown Copyright as it does U.S. Copyright. It is understood that, in asserting Crown Copyright, IEEE in no way diminishes its rights as publisher. Sign only if ALL authors are subject to Crown Copyright.

This will certify that all authors of the work are subject to Crown Copyright. (Appropriate documentation and instructions regarding wording of Crown Copyright notice may be attached.)

AUTHORIZED SIGNATURE

TITLE

NAME OF GOVERNMENT ORGANIZATION

DATE

Notes and Information for Authors and Their Employers

IEEE POLICY

In connection with its publishing activities, it is the formal policy of the IEEE to own the copyrights to all copyrightable material in its technical publications and to the individual contributions contained therein in order to protect the interests of the IEEE, its authors and their employers, and, at the same time, to facilitate the appropriate re-use of this material by others. The IEEE distributes its technical publications throughout the world and does so by various means such as hard copy, microfiche, microfilm, and electronic media. It also abstracts and may translate its publications, and articles contained therein, for inclusion in various compendiums and similar publications, etc. When an article is submitted to the IEEE for publication, the IEEE understands that its acceptance of the article implies that IEEE has the rights to do all of the things it normally does with such an article.

IEEE Policy 6.17 — CLEARANCE OF PAPERS — applies to all material submitted to IEEE: "The IEEE must of necessity assume that material presented at its meetings or submitted to its publications is properly available for general dissemination to the audiences these activities are organized to serve. It is the responsibility of the authors, not the IEEE, to determine whether disclosure of their material requires the prior consent of other parties and, if so, to obtain it."

Furthermore, if an author uses within his/her article material that has been previously published and/or copyrighted, the IEEE must assume that the appropriate permission has been obtained for such use and that any required credit lines, copyright notices, etc. are duly noted.

IEEE OBLIGATIONS

In exercising its rights under copyright, the IEEE will make all reasonable efforts to act in the interests of the authors and employers as well as in its own interest. In handling third-party reprint/republication requests for an IEEE work, the IEEE requires that: 1) The consent of the first-named author be sought as a condition in granting republication (of a full paper) to others. 2) The consent of the employer be obtained as a condition in granting permission to others to re-use all or portions of a paper for promotion or marketing purposes.

AUTHOR/COMPANY RIGHTS

If you are employed and you prepared your paper as a part of your job, the rights to your paper initially rest with your employer. In that case, when you sign the copyright transfer form, we assume you are authorized to do so by your employer and that your employer has consented to all the terms and conditions of this form. If not, it should be signed by someone so authorized. (See also Policy 6.17 above.)

SPECIAL NOTE TO EMPLOYERS: Just as the IEEE requires a signed copyright transfer form (for copyrightable material) in order to do "business as usual," it is the intent of the transfer portion of the form to return rights to the author and employer so that they, too, may do "business as usual."

Please note that, although authors are permitted to reuse all or portions of their IEEE-copyrighted material in other works, this does not include granting third-party requests for reprinting, republishing, or other types of re-use. All third-party requests must be handled by the IEEE Rights and Permissions Office.