JKAU: Eng. Sci., **vol. 13 no. 1,** pp. 95-109 (1421 A.H. / 2001 A.D.)

Reliability Analysis in Distributed Systems Using Fuzzy Set Theory

ZULFIQR AHMED SATTI
Department of Electrical-Electronics Engineering Technology, Yanbu Industrial College, Yanbu Al-sanaiah, Saudi Arabia

ABSTRACT. Distributed systems can be modelled as a collection of different objects (resources) that are interconnected via a communication network and controlled by a distributed operating system. Amoeba, Athena, Mach and V System are some examples of real world distributed systems. They provide cost-effective means for resource sharing and extensibility.

This study focuses on the reliability issue in distributed systems and develops an efficient algorithm for evaluating reliability using fuzzy set theory. With processing elements and communication links having each a certain probability of being operational, there is a certain probability with the extent that it can be successfully executed, also the distribution of the programs and data files affect a program's reliability as well as the overall system's reliability.

Distributed System's Reliability (DSR), which is a reliability measure of the entire system, is defined as the probability that all distributed programs of a given set can run successfully without any fault or error. A unified approach based on a graph model has been developed to generate all the required subgraphs for successful execution of all the programs under consideration. These subgraphs are then used to evaluate the reliability using any terminal reliability algorithm based on path enumeration.

1. Introduction

The development of computer networking and low cost Very Large Scale Integrated Circuits (VLSI) processing devices has led to an increased interest in "Distributed processing systems" (DPSs). In DPSs the computations are distributed among many interconnected processing elements (PE's)[3]. Variety of dis-

tributed systems are created as research projects at various universities and research institutions. For example Amoeba is a distributed systems created at Vriji University of Amsterdam, Athena is another distributed system developed at MIT, Mach distributed system was developed at Carnegie Mellon University, and V Systems was designed at Stanford University[1].

If one site fails in a distributed system the remaining sites can continue operating. If enough redundancy exists in the system, both in hardware and software, the system can continue with its operation, even in the presence of some failure. When the failure is recovered mechanisms are available to integrate it back into the system smoothly[2].

In section 2, factors involved in reliability analysis are mentioned. In section 3, fuzzy set tables are derived and in section 4, algorithms for distributed systems reliability analysis will be given.

2. Reliability Analysis

A distributed task may require one or more resources for the successful execution. For example, a query task may require the cooperation of multiple processing elements and access to a remote database. It may also require the channels for communication among the involved objects[3].

For higher reliability and greater resource sharing many objects are replicated *i.e.* redundant copies of a database file etc. In this environment, a distributed job can be executed when all the required objects (processing elements, communication links, jobs, data files, etc) are available. They may be several possible combinations[4]. For the required object, any combination can be determined based on the availability of objects at a given time and some performance criteria. Aside from processing elements and communication links each having a certain probability of being operational, there is a certain probability associated with the event that it can be successfully executed. Also the distribution of jobs and data files affect a job's reliability as well as overall system's reliability.

3. Derivation of Tables

Degree of membership of the nodes in the distributed system will be calculated by using the concept of fuzzy set theory. This degree of membership will be used in the proposed algorithm in the expansion step.

Consider "Resource Duplication" as a fuzzy set. The elements of this set are resources needed and their degree of membership depend on the number of times the same resource is accessed and is called the occurrence of resource. For example, a resource whose occurrence is 0 might have degree 0.00 and the

resource whose occurrence is 3 might have degree 0.06 and the resources of intermediate occurrences might have intermediate degrees as given in Table 1.

TABLE 1. Degree of resource duplication.

Occurrence of resource	Degree of membership
0	0.00
1	0.02
2	0.04
3	0.06

The maximum number a resource can occur is 3 because it has been assumed as maximum occurrence of any resource in our distributed system. According to this representation the fuzzy set "Duplication" is defined by its domain – the range of values of occurrence (0, 1, 2, 3) and the degree of membership (0.00, 0.02, 0.04, 0.06) respectively.

Consider another fuzzy set "Resource Distribution". The elements of this set are resources needed. In this case degree of membership depends on the availability of the resources. If no copy of a resource is available then its accessibility is zero and if one copy is available then its accessibility is 1 and so on. For example, a resource whose accessibility is 0 might have degree 0.00 and a resource whose accessibility is 3 might have degree 0.90 and the resources of intermediate accessibility might have intermediate degrees. A proposed representation for our distributed system is given in Table 2.

TABLE 2. Degree of resource distribution.

Accessibility	Degree of membership
0	0.00
1	0.30
2	0.60
3	0.90

While searching the graph, it is also possible that one may come across some of those resources which are not required at that time. To model this fact another table (Table 3) known as "Resource Other-occurrence" is derived. The table can be established as follows:

TABLE 3. Degree of resource other-occurrence.

Other-occurrences	Degree of membership
0	0.000
1	0.002
2	0.004
3	0.006

While giving Degree of membership to any node, functions namely, F_{dis}, F_{dup} and $F_{o\text{-}occ}$ will be used. These functions can be described as below:

F_{dis}

I. Gets the programs assigned to the current active vertex.
II. Gets the list of files needed (FN) to run the programs assigned to active vertex.
III. Gets the list of files accessible (FA) through the current active vertex.
IV. Computes the degree using the table "Resource Distribution".

F_{dup}

I. Uses the list of files accessible (FA) through the current active vertex.
II. Computes the degree using the table "Resource Duplication".

$F_{o\text{-}occ}$

I. Uses the list FN to run the programs at the current active x.
II. Uses the list FA through the current active vertex.
III. Computes the degree using the table "Resource Other-occurrence".

4. Algorithm Design

Under this section fuzzy set theory is used to solve the reliability problem of such distributed systems. In this approach the following notations and definitions are used:

AE	Number of edges in distributed system.
AP	Number of programs assigned to the system.
AR	Number of resources available in the system.
AV	Number of vertices included in the system.
AE(f)	Adjacent edges in the current forest f.
Dm(f)	Degree of membership of f.
Dm(Vi)	Degree of membership of the vertices accessible using Vi as root vertex.

E(f)	Set of edges included in the forest f.
EI(Vi)	Set of edges incident with ith vertex.
EI(Vf)	The set of edges incident on the vertices of f which are represented by the set Vf.
Ei, j	An edge between ith and jth vertex.
FA(Vi)	Set of files accessible using vertex Vi as root vertex.
FN(Vi)	Set of files needed for the programs at the ith vertex.
FSF	File spanning forest.
MFSF	Minimum file spanning forest.
Min-Dm	Minimum degree of membership.
NV(f)	Number of vertices in forest f.
PA(Pi)	Set of processors that are designed the ith program.
RR(Vi)	Set of resources required at ith vertex.
Waf	Weight assigned to a file.
X i / V i	ith vertex.

4.1 Model Construction

A distributed system is represented by matrices, graph theoretic notations and fuzzy set notations. The problem is to evaluate execution reliability of a given system using the fuzzy sets approach. For example Fig. 1 represents a distributed system with following characteristics:

DS < V, E, SJ, SR >

V = { V1, V2, V3, V4 } WHERE AV = 4

E = { $E_{1,2}$, $E_{1,3}$, $E_{2,3}$, $E_{2,4}$, $E_{3,4}$ } WHERE AE = 5

SP = { P1, P2, P3 } WHERE AP = 3

SF = { F1, F2, F3, F3 } WHERE AF = 4

PX = < Pid, Rnx >

For example, for x = 1

P1 = < P1, RN1 > where P1 = Prg1, RN1 = { F1, F2, F3 }

Vx = < Vid, PAx, Rax >

For example, for x = 1

V1 = < V1, PA1, RA1 > where Vid = V1 = vertex 1

PA1 = { P1 }, RA1 = { R1, R2 }

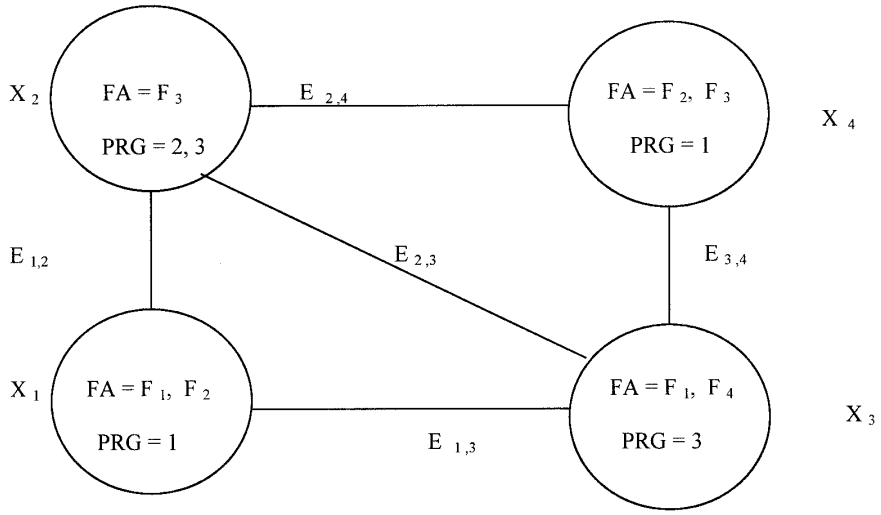

FIG. 1. A distributed system.

4.2 Distributed System Reliability Algorithm (DSR)

The algorithm DSR is used to generate all the MFSFs for the jobs assigned to the system. First in the initialization step all subsets of computers that jointly run all programs are obtained and stored in a list TRY. Then it computes the degree of membership of each forest at root node. This step is conducted to check whether the request is satisfied at the root node or not.

In the next step the algorithm checks whether the generated forest is a MFSF. If yes then MFSF is added to the list FOUND and removed from the list TRY. Then the forest whose request is not satisfied yet is enlarged with the hope that the request may be satisfied after enlargement of the forests. In the enlargement step Fuzzy set theory is used to get the most appropriate edges to include in the forest. Then it removes the duplicate forest from the current list which may be generated during enlargement step. It will continue until the list TRY is empty.

Algorithm: DSR

Input: DS < V, E, SJ, SR >
Output: MFSFs
Begin

Step 1: Initialization

Let TRY = Set of all f = { P1, P2, ... , Pm }

such that P1 ∈ PA(P1), ... Pm ∈ PA(Pm)
where PA(Pi) denotes the set of computers that can run PRG1.

Step 1.1: Remove duplicates

For all f ∈ TRY do
  for all (VI, Vj ∈ f) do
    if Vi = Vj then
      f ← f – Vi
  od
od

Step 1.2: Remove supersets

For all (f_i, f_j ∈ TRY) do
  if $f_i \cap f_j = f_i$ then
    TRY ← TRY – f_j
  if $f_i \cap f_j = f_j$ then
    TRY ← TRY – f_i
  od

Step 1.3: Finding degree of membership

For all f ∈ TRY do
  Dm(f) = Degree _ of _ membership (f)
od

Step 1.4: Finding minimum _ degree _ of _ membership required

For any f ∈ TRY do

$$Min - Dm = \sum_{i=1}^{NV(f)} (Waf * FN(Vi)$$

od

Step 2: Generating all MFSFs

Repeat

Step 2.1: Checking step

For all f ∈ TRY do
  if CHECK(f) then

```
    begin
        FOUND ← FOUND + f
        TRY ← TRY – f
    end
  od
```

Step 2.2: Expanding step

```
NEW = ϕ
  For all f ∈ TRY do
    NEW ← NEW = EXPAND(f)
    od
    TRY = CLEANUP(NEW)
  Until (TRY = ϕ)
end.
```

Function: Check

Discussion

This algorithm is used to check whether the generated forest is a MFSF. First it will get the degree of membership of the current forest, and compares it to the Min _ Dm. If it is equal or greater than the Min _ Dm then it means that generated forest is already a FSF. In next step it checks that whether this FSF is superset of any already found MFSFs. If it is not then it is added to the list FOUND as a newly found MFSF.

Function: Check

Input: Forest
Output: Boolean value

Function: Check(f)

```
Begin
CHECK = false
if (Dm(f) > = Min _ Dm) then
begin
if (FOUND = ϕ) OR (f_i ∈ FOUND , f_i ∩ f not = f_i) then
        CHECK = true
else
        TRY ← TRY – F
end
end.
```

Function: Check _ Unique

Discussion

This algorithm is used to check whether the generated forest is a unique MFSF, that is it not already included in the list FOUND.

It compares the given forest to each fores in the list FOUND, which is the list of MFSFs, and if it can not find a duplicate in the list then it declares the given forest as a unique forest.

Function: Check _ Unique

Input: Forest
Output: Boolean value

Function: Unique(f)

```
Begin
   UNIQUE = false
      if (FOUND = ϕ) or (all f_i ∈ FOUND, f_i ∩ f not = f) then
         UNIQUE = true
end.
```

Function: Cleanup

Discussion

This algorithm is used to remove the duplicate forests generated in the expand step. It checks from the current set of forests if many of the forest is equal to any other forest in the current set then it removes one of them.

Function: Cleanup

Input: A list of forests
Output: A list of forests without duplicates

Function: Cleanup(New)

```
Begin
   For all (f_i, f_j ∈ NEW) do
            if f_i – f_j = ϕ then
            NEW ← NEW – f_j
   od
end.
```

Function: Expand

Discussion

This algorithm is used to enlarge those forests which are not yet FSFs. In the first step it gets all the possible edges which can be used to expand the given forest. Then it selects each of them and computes their degree of membership. It also selects those edges only whose inclusion will not make a cycle. Then their degrees of membership are compared to Min _ Dm to get the FSF. In the next step the forests with higher degree of membership are selected.

Function: Expand

Input: Forest
Output: A list of forests

Function: Expand(f)

```
Begin
tmp = ϕ
tmpDm = ϕ
   EI (f) = EI (Vf) – E(f)
   kount = EI (f)
   For all (xij ∈ AE(f), Vi ∈ Vt ∩ Vj not ∈ Vt) do
   begin
      Newt = f + { Xi,j }
   If UNIQU(f) THEN
   Begin
      Dm (Newt) = Degree _ of _ membership(Newt)
      tmp = tmp + Newt
      tmpDm = tmpDm + Dm(Newt)
   end
   else
      kount = kount – 1
   end
tmp1 = f
   If kount > 1 then
   begin
      For all f ∈ tmp
      if Dm(f) > = Min-Dm then
      begin
         tmp1 = tmp1 + f
         tmp = tmp – f
         tmpDm = tmpDm – Dm(f)
```

```
        kount = kount – 1
    end
      if kount > 0 then
      begin
        limit = kount
        For I = to limit
        begin
        select Max _ value(f)  from  tmpDm
        tmp1 = temp1 + Max _ value(f)
        tmpDm = tmpDm – Max _ value(f)
    end
    end
    EXPAND = tmp1
    end
    else
      EXPAND = tmp
  end.
```

4.3 Algorithm Degree _ of _ Membership

Discussion

This algorithm is used to compute the degree of membership of a given forest.

In first step it finds out the active vertices of the given forest. Next it finds the degree of membership of each active vertex. Finally in last step the degree of membership is assigned to the forest.

Algorithm Degree _ of _ Membership

Input: Forest
Output: Degree
Begin

Step 1: Finding degree of each vertex in forest f

For all Vi ∈ f

/ * NV(Vi) = Number of vertices in forest f

FA (Vi) = Files accessible through Vi */

$$Dm(Vi) = \sum_{K=1}^{|FA(Vi)|} (F_{dis} + F_{dup} + F_{o-occ})$$

Step 2: Assigning degree _ of _ membership to forest

$$Dm(f) = \sum_{i=1}^{NV(f)} Dm(Vi)$$

end.

5. An Example

Let us consider the distributed system shown in Fig. 1. It has four nodes and five communication links. The allocation of programs and resources is as follows:

Files assigned to each node are:

$FA_1 = \{ F_1, F_2 \}$

$FA_2 = \{ F_3 \}$

$FA_3 = \{ F_1, F_4 \}$

$FA_4 = \{ F_2, F_3 \}$

Jobs assigned to each node are:

$JA_1 = \{ Prg_1 \}$

$JA_2 = \{ Prg_2, Prg_3 \}$

$JA_3 = \{ Prg_3 \}$

$JA_4 = \{ Prg_1 \}$

The files required by each program are:

$FA(V1) = \{ F_1, F_2, F_3 \}$

$FA(V2) = \{ F_1, F_2, F_4 \}$

$FA(V3) = \{ F_1, F_2, F_3, F_4 \}$

When we applied DSR algorithm on this distributed system, it generated a set of Minimum File Spanning Forest (MFSFs). These MFSF are shown in Fig. 2.

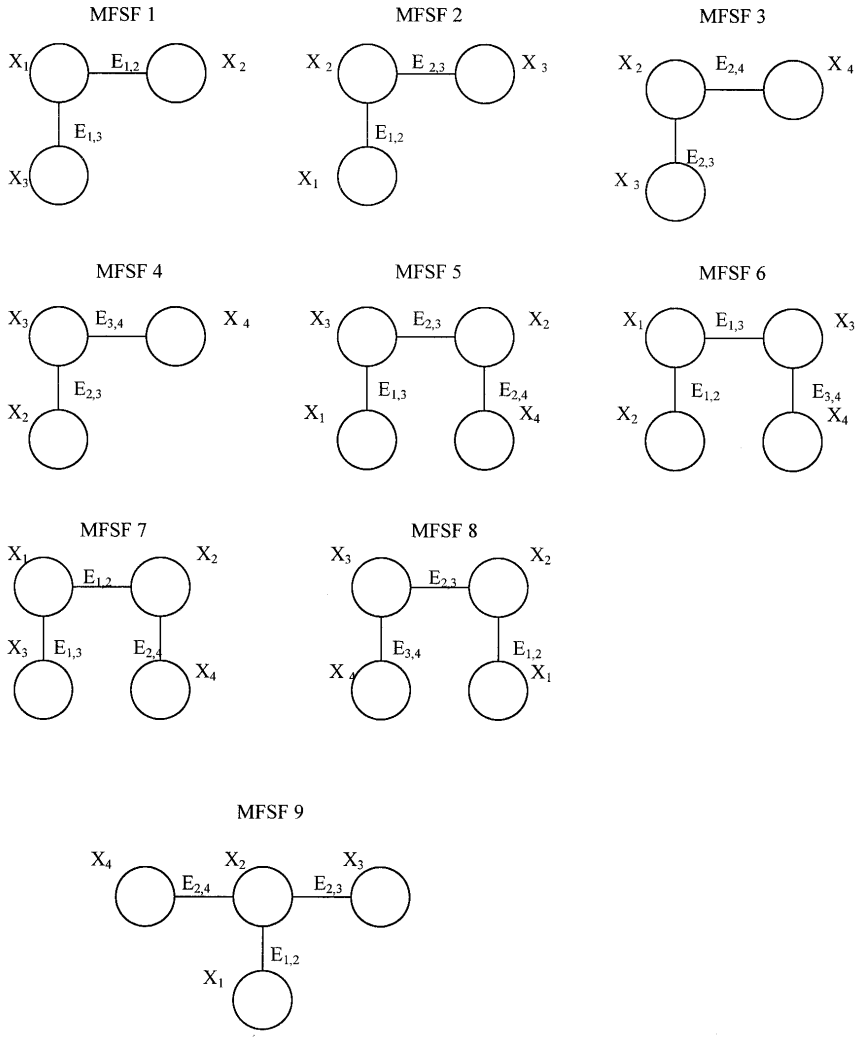

FIG. 2. Minimum File Spanning Forests (MFSFs).

6. Conclusion

In this paper, the reliability issues in distributed systems are addressed and a new reliability measure is presented. An efficient algorithm for evaluating reliability is developed. It enumerates all the possible sub-graphs (MFSFs) where the system successfully operates. Fuzzy set theory is used to find the possible sub-graph in the distributed system. As a result, a new approach is introduced to solve the reliability program.

In the expanding step of the proposed algorithm instead of adding each adjacent edge to generate MFSFs, restrictions were imposed to control immeasurable combinatorial superset growth.

Since all the methods of terminal reliability are NP-hard, our proposed algorithm is not applicable to all types of graphs. It works well for:

1. Fully connected networks
2. Binary structured networks
3. Star networks
4. Single link ring networks
5. Double links networks

References

[1] **Shay, W.A.,** *Introduction to operating systems,* Harper Collins Publisher, (1993) 31-34.
[2] **Peterson, J.L., Silberschatz, A.,** *Operating System Concepts,* Addision Wesley, (1986).
[3] **Prasanna Kumar, V.K.** and **Hariri, S.,** Distributed Program Reliability Analysis, *IEEE transaction on software engineering,* (Jan. 1986).
[4] **Hariri, S., Raghavendra, C.S.,** Reliability Analysis in Distributed System, *6th International Conference on Distributed Computing Systems, (1986)* 564-571.
[5] **Hariri, S., Raghavendra, C.S.,** SYREL: A Symbolic Reliability Algorithm based on path and cut set method, *IEEE transaction on reliability,* (October 1986).
[6] **Tanenbaum, A.S.** and **Woodhull, A.,** *Operating Systems: Design and Implementation,* Prentice-Hall Inc. (1977) 12-13.

تحليل الإعتمادية في أنظمة التوزيع باستخدام نظرية فزي (Fuzzy)

ذو الفقار أحمد ساتي
كلية ينبع الصناعية ، قسم الكهرباء والإلكترونيات
ينبـــع الصناعية – المملكة العربية السعودية

المستخلص . يمكن تمثيل أنظمة التوزيع كمجموعة من مصادر مختلفة مرتبطة فيما بينها بشبكة اتصالات محكومة بنظام التشغيل وأمثلة ذلك : أميبيا (Amoeba) ، أثينا (Athena) ، ماك (Mach) ، ونظام (V System) . ولهذه الأنظمة تأثير مباشر في توفير التكلفة نظرًا للمشاركة في مصادر المعلومات والقابلية للامتداد والتوسع .

تركز هذه الدراسة على طريقة الاعتمادية في أنظمة التوزيع وتطوير كفاءة الخوارزميات (البرامج) لتقييم الاعتمادية وذلك باستخدام نظرية فزي (Fuzzy) . وبتشغيل كل عنصر واستخدام وسيلة الاتصالات المناسبة يكون هنالك احتمال محدد ودقيق لكل حدث ممكن تشغيله بنجاح . وكذلك برامج التشغيل وملفات المعلومات لها التأثير الملحوظ على الاعتماد على هذه البرامج كما هو الحال في النظام بأكمله .

تعتبر الاعتمادية لأنظمة التوزيع كمقياس للنظام كاملاً وهي عبارة عن احتمال أن كل برامج أنظمة التوزيع لأي مجموعة معطاة ممكن أن تعمل بصورة ناجحة بدون حدوث أي أخطاء . وطريقة الموحد (Unified) تعتمد على الرسم الكروكي والذي يمكن الحصول من خلاله على كل الرسومات الفرعية والجزئية والتي توفر الضمان الكامل لنجاح تشغيل جميع البرامج المستخدمة والخاضعة للفحص والاختبار . هذه الرسومات الفرعية تستخدم لاحقًا في تقييم الاعتمادية باستخدام الخوارزميات لأي وحدة طرفية اعتمادًا على طريقة الإحصاء .