Development of Reliability Model for Complex Distributed Software System Considering Erasure Code, Physical and Logical Interlink

B.Ashwin [1] and Rajakannu Amuthakkannan[2]

***Abstract*—** Distributed System is the one where there are many computer systems that are connected via a network. The computers interact with each other in order to achieve a common goal. Software systems are constructed by integrating software subsystems by logical and physical interlinks. Software subsystem within a network suffers from the logical and physical interlink reliability where some papers have shown the reliability of the system by using it. But reconstruction of the erasure code in the distributed systems due to previous failure will also affect the reliability of the distributed software system. So, in this paper, a new model is proposed considering logical, physical interlinks reliability along with the erasure code reconstruction to find the reliability of the complex distributed subsystem.

***Keywords*—** Distributed systems, Erasure code, Logical interlink, Physical interlink

I Introduction

RELIABILITY is the ability of the computer program to perform its intended function and operation in a system environment without expecting a failure. Various models are available to assess the reliability of individual components. Shukla et al., [6] proposed a framework for assessing the reliability of individual components by test case execution and evaluation techniques. Wang et al [6] proposed a reliability model for complex distributed software system considering physical and logical interlink but without considering the sub system. The extension of Wang et al model is done by Amuthakkannan et al.,[9] by considering subsystem's physical and logical reliability. But Amuthakkannan et al did not consider the reliability change due to erasure code reconstruction. Actually, recent technology of networking allows designing distributed software system consisting of a number of software sub systems at various hierarchy levels which are physically interlinked to provide a solution for any software application. Normally erasure code is occurring due to transmitting a fixed set of data to multiple clients over unreliable links. So, erasure code correction is necessary to recover lost data in a complex distributed system because to rebuild the whole system is loss of money and time consuming. So, erasure code reconstruction concept is widely followed in the software industries to recover the failure data. But reconstruction of code may also affect the reliability of the system which was not considered previously by the researchers. In this paper the reliability changes due to erasure code reconstruction is the core content. By considering this, it is proposed to extent the Architecture based software reliability model to the distributed software system by incorporating interface and erasure code reliability. Enterprise resource management is taken as the case study and assessed using the newly proposed approach.

B.Ashwin is with Department of Computer Science and Engg, Kumaraguru College of Technology, Coimbatore, Tamil Nadu, India (corresponding author's phone: +917708467183; e-mail:b.ashwin1991@gmail.com).

Rajakannu Amuthakkannan is working as senior faculty at Department of Mechanical and Industrial Engineering, Caledonian College of Engineering (affiliated to Glascow Caledonian University,Scotland,UK), Muscat, Sultanate of Oman (e-mail:amuthakkannan@caledonian.edu.om phone:+968-98893272)

II Problem Background

The functioning of computers depends on the correct operation of their software. Unlike computer hardware, whose reliability has improved dramatically with the advances in integrated circuit technologies, the improvements in the reliability of software has been lagging. The engineering of reliable software is still a young and immature discipline. The main reason for the system failures are poor development practices, incorrect assumptions with regard to system requirements, Poor user interfaces, faulty hardware, inadequate user training /user error, Poor fit between systems and organisations. In distributed systems, there are many opportunities to fall the reliability because of complex software and too many hardware. So, in-depth research work is required to improve the reliability of system. The frequent software failure will lead to erasure code reconstruction in the concept of distributed system reliability which engineering is in infant stage at present.

III Literature Review

Few Models are available for estimating the reliability of component based software systems. Various approaches are available for assessing the reliability of individual components. Dolbec et al.,[1] provided a model which estimates the reliability of component based software system based upon component usage ratio. Krishnamurthy et al. [3] assessed the reliability of component based reliability estimation. This approach does not consider component interface faults. Wang et al.[6] proposed an approach which helps in estimating the reliability of a heterogeneous architecture using Markovian-chain properties. Hamlet, et al., [2] presented a theory which describes how a component developer can design and test their components to produce measurements that are later used by system designer to assess

the composite system reliability. May [4] derived a new model that describes how test based software reliability estimation depends on the component structure of the code. Yacoub et al., [7] proposed a new technique called scenario based reliability estimation which builds on scenarios used in the analysis phase of component based system development. Shukla et al.,[5] proposed a theoretical framework for assessing the reliability of component based software system which incorporates test case execution and output evaluation. Amuthakkannan et al[9] considered the architecture based software reliability model proposed by Wen-Li Wang, et al.[6] and extended for physical and logical interlink reliability of subsystems.

IV Architecture Based Model

The state diagram shown in figure 1 is a directed graph in which node S_i and S_j represents adjacent states and the transition from state S_i to S_j is represented by a directed edge(S_i , S_j).

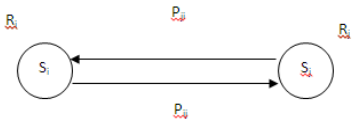

Fig. 1, Simple State Diagram

Based on the state diagram, it is defined M as transition matrix and the value of the entry M(i,j), which can be computed as $R_i * P_{ij}$, indicates the successful arrival at state S_j from S_i , i.e., the correct execution of the S_i and the occurrence of the transition from S_i to S_j .

$$M(i,j) = \begin{array}{c} \\ S_1 \\ S_2 \\ \\ S_n \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & & S_n \end{array} \\ \begin{bmatrix} 0 & R_1*P_{12} & \cdots & R_1*P_{1n} \\ R_2*P_{21} & 0 & \cdots & R_2*P_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ R_n*P_{n1} & R_n*P_{n2} & \cdots & 0 \end{bmatrix} \end{array}$$

Let $S = \{ S_1 , S_2 , \ldots, S_n \}$ be the set of states in the state diagram where S_i is the initial state and S_n is the final state. $M^k(i,j)$ represents the probability of reaching state S_j from S_i through k transitions. Therefore, the reliability R beginning from Si to S_j with total k transitions is represented as

$R=M^k(i, j)*R_j$

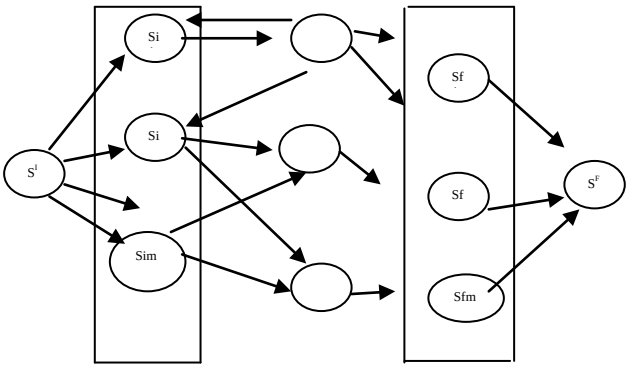

Fig. 2, State diagram with multiple states

From initial state S_1 to final state S_n , the number of transitions k may vary from 0 to infinity, where 0 means that the initial state is also the final state and infinity means that a cyclic loop may occur indefinitely among the states. Therefore, it is necessary to consider every possible outcome of state transitions. Let T be a matrix such that

$$T=I+M^1+M^2+M^3+\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots=\sum_{K=0}^{\infty} M^K$$

This can be written as

$T= (I-M)^{-1}$; $T=1/I-M$

Where I is the identity matrix of size $n*n$. The overall system reliability can be computed as follows:

$R=T(1, n)*R_n$,Where $T(1, n) = (-1)^{n+1}(|E|/|I-M|)$ is the determinant of the remaining matrix excluding the nth row and first column of the matrix $(I- M)$.

$$I(i,j) = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \cdots & \cdots & \cdots & \cdots \\ 0 & \cdots & \cdots & 1 \end{bmatrix}$$

$$I-M = \begin{array}{c} \\ S_1 \\ S_2 \\ \\ S_n \end{array} \begin{array}{c} \begin{array}{cccc} S_1 & S_2 & & S_n \end{array} \\ \begin{bmatrix} 1 & -R_1*P_{12} & \cdots & -R_1*P_{1n} \\ -R_2*P_{21} & 1 & \cdots & -R_2*P_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ -R_n*P_{n1} & -R_n*P_{n2} & \cdots & 1 \end{bmatrix} \end{array}$$

$$E = \begin{array}{c} \\ s_1 \\ S_2 \\ \\ S_{n-1} \end{array} \begin{array}{c} \begin{array}{ccc} S_2 & & S_n \end{array} \\ \begin{bmatrix} -R_1*P_{12} & \cdots & -R_1*P_{1n} \\ 1 & \cdots & -R_2*P_{2n} \\ \cdots & \cdots & \cdots \\ -R_{n-1}*P_{n-1,2} & \cdots & 1 \end{bmatrix} \end{array}$$

Model is based on one single initial state and one single final state, whereas a large system may have a set of initial states $I = \{ S_{i1} , S_{i2} , \ldots, S_{im} \}$ and a set of final states $F = \{ S_{1f} , S_{2f} , \ldots, S_{nf} \}$. In such a situation, it is revised the problem of multiple initial and final states to one initial state and one final state problem by introducing a *super-initial* and a *super-final* state to the state diagram. Figure 4 shows how a state diagram with multiple initial states and final states in the dotted rectangular area can be converted to a state diagram with only a single initial state and a single final state. A *super-initial* state S^I and a directed edge (S^I , S_{ij}) is added with its transition probability P_{ij} , observed from the operational profile, for each j = 1, 2, …., m. Similarly, a *super-final* state is created S^F and a directed edge (S_{fj} , S^F) with transition probability 1 for each j = 1, 2, …. , n. The reliabilities for both states S^I and S^F are assigned as 1.

A. *Proposal for Extension of Reliability Assessment Model to Complex Distributed Software System*

Distributed software systems are constructed by integrating software subsystems by logical and physical interlinks. Software subsystems within a physical system can be interconnected by logical interlinks. While the subsystems on different physical systems can be interconnected by physical interlink and logical interlinks. When assessing the reliability of such distributed software system, the physical interlink reliability between software subsystems must be taken into consideration. So the model provided by Wen-Li Wang[6] and Amuthakkannanet al [9] is extended to complex distributed system, by considering logical and physical interlink reliability with erasure code reconstruction.

Let $SS = \{SS_1, SS_2 \ldots SS_n\}$ be the set of sub systems. Where SS_{i*} are the initiating sub systems and SS_{f*} are the Terminating sub systems. SS^I is the super initial subsystem. SS^F is the super final subsystem. M_D is the Transition Matrix for distributed system. SP_{ij}=Transmission Probability from sub system i to j. SR_i=Sub System i's Reliability. IR_{ij} = Interface Reliability. PIR_{ij} = Physical Interlink Reliability between subsystem i and j.LIR_{ij}=Logical Interface Reliability. $IR_{ij}=PIR_{ij} * LIR_{ij}$. Logical Interface reliability is defined as the probability of two interacting systems has matching interfaces. A system interface defines how the system interacts with other systems. Interfaces describe the import and export relationship. A set of exported interfaces specifies the services that the systems can provide. A set of imported interfaces specifies the services which this system requires from the other systems. A mismatch is an interface can result in the incompatibility of the structure or the sequence of messages exchanged between systems, incompatibilities in data formats, types and message protocols, or misunderstood roles in interaction. Physical interlink reliability is essential in case there is a system distribution across networks. A message exchanged between systems in a distributed environment exposed to possible problems with the operating system call, hardware technology, communication physical subsystems and the physical network layer. Problems such as delays, congestion, physical failures, timing, and protocol affect link reliabilities [7].

Model is based on one single initial state and one single final state, whereas a large system may have a set of initial states $I = \{ S_{i1}, S_{i2}, \ldots, S_{im} \}$ and a set of final states $F = \{ S_{1f}, S_{2f}, \ldots, S_{nf} \}$. In such a situation, it is revised the problem of multiple initial and final states to one initial state and one final state problem by introducing a *super-initial* and a *super-final* state to the state diagram. Figure 1 shows how a state diagram with multiple initial states and final states in the dotted rectangular area can be converted to a state diagram with only a single initial state and a single final state. A *super-initial* state S^I and a directed edge (S^I, S_{ij}) is added with its transition probability P_{ij}, observed from the operational profile, for each $j = 1, 2, \ldots, m$. Similarly, a *super-final* state is created S^F and a directed edge (S_{fj}, S^F) with transition probability 1 for each $j = 1, 2, \ldots, n$. The reliabilities for both states S^I and S^F are assigned as 1.

$R_D = T(1,n)*SR_n$, Where $T(1,n)=(-1)^{n+1}$ ($|E_D| / |I-M_D|$)

R_D = Reliability of the Distributed software system

E_D = the matrix which is excluding the n^{th} row and first column of the matrix $I-M_D$

n = Number of Sub systems.

$$M(i,j) = \begin{array}{c} \\ SS_1 \\ SS_2 \\ \\ \\ \\ SS_n \end{array} \begin{array}{c} \begin{array}{cccc} SS_1 & SS_2 & & SSn \end{array} \\ \begin{bmatrix} 0 & SR_1*SP_{12}*PIR_{12}*LIR_{12} & \ldots & SR_1*SP_{1n}*PIR_{1n}*LIR_{1n} \\ SR_2*SP_{21}*PIR_{21}*LIR_{12} & 0 & \ldots & SR_2*SP_{2n}*PIR_{2n}*LIR_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots \\ SR_n*SP_{n1}*PIR_{n1}*LIR_{n1} & SR_n*SP_{n2}*PIR_{n2}*LIR_{n2} & \ldots & 0 \end{bmatrix} \end{array}$$

$$I\text{-}Md = \begin{array}{c} \\ SS1 \\ SS2 \\ SSn \end{array} \begin{array}{c} \begin{array}{ccc} SS1 & SS2 \ldots & SSn \end{array} \\ \begin{bmatrix} 1 & -SR1*SP12*IR12*ER12 & -SR1*SP1n*IR1n*ER1n \\ -SR1*SP12*IR12*ER12 & 1 & -SR2*SP2n*IR2n*ER2n \\ -SRn*SPn1*IRn1*ERn1 & -SRn*SPn2*IRn2*ERn2 & 1 \end{bmatrix} \end{array}$$

$$I\text{-}Md = \begin{array}{c} \\ SS1 \\ SS2 \\ SSn \end{array} \begin{array}{c} \begin{array}{ccc} SS1 & SS2 \ldots & SSn \end{array} \\ \begin{bmatrix} 1 & -SR1*SP12*IR12*ER12 & -SR1*SP1n*IR1n*ER1n \\ -SR1*SP12*IR12*ER12 & 1 & -SR2*SP2n*IR2n*ER2n \\ -SRn*SPn1*IRn1*ERn1 & -SRn*SPn2*IRn2*ERn2 & 1 \end{bmatrix} \end{array}$$

$$Ed = \begin{array}{c} \\ SS1 \\ SS2 \\ SSn-1 \end{array} \begin{array}{c} \begin{array}{ccc} SS2 \ldots & SSn \end{array} \\ \begin{bmatrix} -SR1*SP12*IR12*ER12 \ldots & -SR1*SP1n*IR1n*ER1n \\ 1 \ldots & -SR2*SP2n*IR2n*ER2n \\ -SRn*SPn2*IRn2*ERn2 \ldots & 1 \end{bmatrix} \end{array}$$

V CASE ANALYSIS

Enterprise resource management system is a software based system which is used for attaining the objective of best utilization of resources for supporting organization profitability. It is a complex distributed system which is built by integrating the sales and marketing support software(subsystem1), Inventory control support software (subsystem2), Total quality management support software(subsystem3),Production planning and control Support software(subsystem4), Human resource management support software(subsystem5), Project management support software (Subsystem 6) (Figure 3) by means of physical interlink. When the reliability of the individual sub systems is known, the overall system reliability can be calculated by considering the interface reliability and overall system architecture.

SR_1 = Reliability of the super initial subsystem, SR_8 =Reliability of the terminating subsystem, SR_2 =Reliability of the sales and marketing support software, SR_3= Reliability of the inventory control support software, SR_4= Reliability of the

total quality management support software, SR_5= Reliability of the production planning and control support software, SR_6=Reliability of the human resource management support software, SR_7= Reliability of the project management support software. SP_{ij} =Probability of transmission from subsystem i to subsystem j. IR_{ij}=Interlink reliability from subsystem i to subsystem j. ER_{IJ} = Erasure code Reliability from subsystem i to subsystem j.

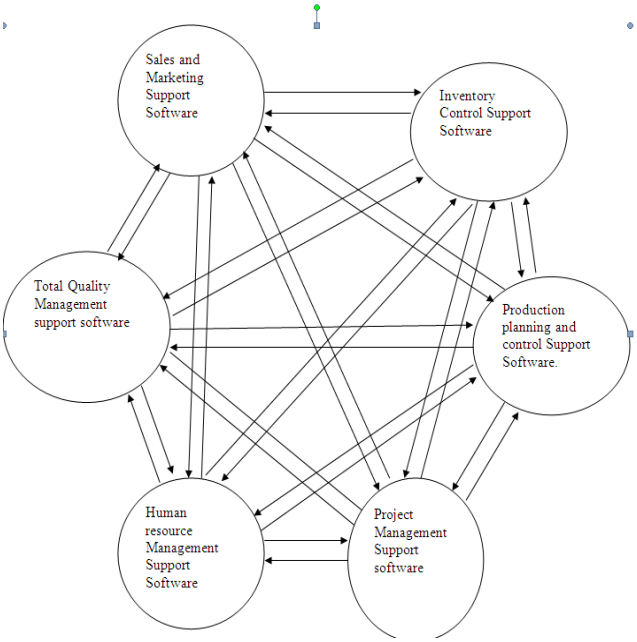

Fig. 3, Enterprise Resource Management Systems

A. Input Data

The probability of transmission from one subsystem to another subsystem, subsystems reliability, interlink reliability between subsystems are assumed to be available. The data are listed as follows.

TABLE I,PROBABILITY OF TRAMSMISSION AND INTERLINK RELIABILITY VALUES

Probability of transmission

SP11 = 0	SP37 = 0.2	SP65 = 0.15
SP12 = 0.21	SP38 = 0.17	SP66 = 0
SP13 = 0.31	SP41 = 0	SP67 = 0.25
SP14 = 0.13	SP42 = 0.2	SP68 = 0.28
SP15 = 0.16	SP43 = 0.2	SP71 = 0
SP16 = 0.21	SP44 = 0	SP72 = 0.2
SP17 = 0.31	SP45 = 0.15	SP73 = 0.15
SP18 = 0.22	SP46 = 0.3	SP74 = 0.2
SP21 = 0	SP47 = 0.15	SP75 = 0.2
SP22 = 0	SP48 = 0.14	SP76 = 0.25
SP23 = 0.15	SP51 = 0	SP77 = 0
SP24 = 0.3	SP52 = 0.2	SP78 = 0.19
SP26 = 0.2	SP53 = 0.2	SP81 = 0
SP27 = 0.2	SP54 = 0.3	SP82 = 0
SP28 = 0.25	SP55 = 0	SP83 = 0
SP31 = 0	SP56 = 0.15	SP84 = 0
SP32 = 0.2	SP57 = 0.15	SP85 = 0
SP33 = 0	SP58 = 0.32	SP86 = 0
SP34 = 0.15	SP61 = 0	SP87 = 0
SP35 = 0.15	SP62 = 0.25	SP88 = 0
SP36 = 0.3	SP63 = 0.2	
	SP64 = 0.15	

Interlink Reliability

IR12 = 0.76	IR27 = 0.88	IR44 = 1
IR13 = 0.87	IR28 = 0.72	IR45 = 0.75
IR14 = 0.74	IR31 = 0	IR46 = 0.68
IR15 = 0.64	IR32 = 0.85	IR47 = 0.87
IR16 = 0.65	IR33 = 0	IR48 = 0.78
IR17 = 0.69	IR34 = 0.62	IR51 = 0.79
IR18 = 0.85	IR35 = 0.79	IR52 = 0.77
IR21 = 0	IR36 = 0.67	IR53 = 0.84
IR22 = 0	IR37 = 0.83	IR54 = 0.78
IR23 = 0.97	IR38 = 0.69	IR55 = 0
IR24 = 0.76	IR41 = 0	IR56 = 0.64
IR25 = 0.88	IR42 = 0.83	IR57 = 0.87
IR26 = 0.74	IR43 = 0.74	IR58 = 0.62

TABLE II,SUBSYSTEM AND ERASURE CODE RELIABILITY VALUES

IR61 = 0	IR71 = 0	IR82 = 0
IR62 = 0.78	IR72 = 0.56	IR83 = 0
IR63 = 0.93	IR73 = 0.65	IR84 = 0
IR11 = 0	IR74 = 0.75	IR85 = 0
IR64 = 0.86	IR75 = 0.67	IR86 = 0
IR65 = 0.78	IR76 = 0.78	IR87 = 0
IR66 = 0	IR77 = 0	IR88 = 0
IR67 = 0.58	IR78 = 0.91	
IR68 = 0.81	IR81 = 0	

Subsystems Reliability

SR1 = 1
SR2 = 0.94
SR3 = 0.95
SR4 = 0.92
SR5 = 0.98
SR6 = 0.93
SR7 = 0.95
SR8 = 1

Erasure code Reliability

ER12 = 0.75	ER37 = 0.72	ER11 = 0
ER13 = 0.78	ER38 = 0.71	ER64 = 0.77
ER14 = 0.81	ER41 = 0	ER65 = 0.74
ER15 = 0.82	ER42 = 0.81	ER66 = 0.82
ER16 = 0.77	ER43 = 0.83	ER67 = 0.84
ER17 = 0.73	ER44 = 0.85	ER68 = 0.79
ER18 = 0.74	ER45 = 0.76	ER71 = 0
ER21 = 0	ER46 = 0.84	ER72 = 0.76
ER22 = 0.77	ER47 = 0.78	ER73 = 0.81
ER23 = 0.77	ER48 = 0.83	ER74 = 0.79
ER24 = 0.79	ER51 = 0	ER75 = 0.82
ER25 = 0.82	ER52 = 0.82	ER76 = 0.77
ER26 = 0.84	ER53 = 0.74	ER77 = 0.84
ER27 = 0.77	ER54 = 0.81	ER78 = 0.83
ER28 = 0.85	ER55 = 0.72	ER81 = 0
ER31 = 0	ER56 = 0.74	ER82 = 0
ER32 = 0.76	ER57 = 0.86	ER83 = 0
ER33 = 0.73	ER58 = 0.71	ER84 = 0
ER34 = 0.82	ER61 = 0	ER85 = 0
ER35 = 0.85	ER62 = 0.81	ER86 = 0
ER36 = 0.84	ER63 = 0.79	ER88 = 0

Intermediate Calculations

$$E_D = \begin{bmatrix} & -0.1197 & -0.2104 & -0.0779 & -0.0788 & -0.1051 & -0.1561 & -0.1384 \\ & 1.0000 & -0.1053 & -0.1693 & -0.1018 & -0.1168 & -0.1274 & -0.1438 \\ & -0.1227 & 1.0000 & -0.0725 & -0.0957 & -0.1604 & -0.1135 & -0.0791 \\ & -0.1237 & -0.1130 & 1.0000 & -0.0787 & -0.1577 & -0.0937 & -0.0834 \\ & -0.1237 & -0.1218 & -0.1857 & 1.0000 & -0.0696 & -0.1100 & -0.1380 \\ & -0.1470 & -0.1367 & -0.0924 & -0.0805 & 1.0000 & -0.1133 & -0.1666 \\ & -0.0809 & -0.0750 & -0.1126 & -0.1044 & -0.1426 & 1.0000 & -0.1364 \end{bmatrix}$$

$$I\text{-}M_D = \begin{bmatrix} 1.0000 & -0.1197 & -0.2104 & -0.0779 & -0.0788 & -0.1051 & -0.1561 & -0.1384 \\ 0 & 1.0000 & -0.1053 & -0.1693 & -0.1018 & -0.1168 & -0.1274 & -0.1438 \\ 0 & -0.1227 & 1.0000 & -0.0725 & -0.0957 & -0.1604 & -0.1135 & -0.0791 \\ 0 & -0.1237 & -0.1130 & 1.0000 & -0.0787 & -0.1577 & -0.0937 & -0.0834 \\ 0 & -0.1237 & -0.1218 & -0.1857 & 1.0000 & -0.0696 & -0.1100 & -0.1380 \\ 0 & -0.1470 & -0.1367 & -0.0924 & -0.0805 & 1.0000 & -0.1133 & -0.1666 \\ 0 & -0.0809 & -0.0750 & -0.1126 & -0.1044 & -0.1426 & 1.0000 & -0.1364 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1.0000 \end{bmatrix}$$

The determinant value of *E*D and *I – M*D is calculated and noted as follows.

|Ed|=-0.2598

|I-Md|=0.7337

Rd=T(1,n)SRn

$T(1,n)=(-1)^{n+1}$ ($|E_D|$ / $|I\text{-}M_D|$)

Here n=8

$T(1, n) = (-1)^{8+1}$ ($|E_D|$ / $|I\text{-}M_D|$)

T (1,8)=0.3541
Rd=T (1, 8) SR8
Rd=0.3541*1
Rd=0.3541
Reliability of the given Enterprise Resource Management System = 35.41%

VI DISCUSSION

Amuthakkannan, et al.[9] discussed about the reliability of distributed software for the enterprise resource management system is 95.9% where physical and interlink reliability was considered. The same data considered for this model with the inclusion of Erasure along with the physical and logical interlink reliability. The Reliability of the Enterprise Resource Management system is received from the proposed model is 35.41% (Figure 4).This shows the reliability of the system has directly come down with the consideration of erasure code reconstruction. The big difference in reliability is because of consideration of erasure code reconstruction.The data considered for this case study is based on assumption only. However, it shows the erasure code reconstruction will affect the reliability of the total system. The model is quite good and gives adequate confidence about the system because the reliability of all the components has been considered.

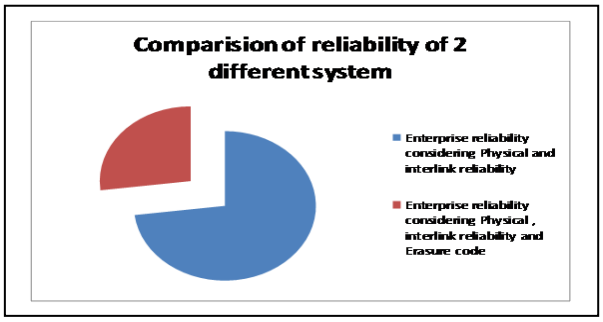

Fig. 4, comparison between Amuthakkannan et al[10] model with the proposed model

Reliability of the system is considered to be the highest concurrency, consistency, optimized for the common storage is all at its highest when the erasure code is considered [8]. Proper use of erasure codes provides greater space efficiency and fine network tuneable levels of protection, at the cost of greater complexity. It is possible to improve the model by considering Heterogeneity, Error Control, Testing, Synchronization, Measurement, debugging and Resource Management

VII CONCLUSION

An architecture based model for assessing the reliability of the distributed system is analysed. The interlink reliability is one of the important property of overall system reliability in distributed software systems. So it is proposed to extend the architecture based model to complex distributed system by incorporating the interface reliability and hierarchy of subsystems. The failure of erasure code technique may restrict the reliability of the distributed system. It is also considered to find the final reliability of the distributed system. An enterprise resource management system is considered as example of distributed system and the reliability of the enterprise resource management system is assessed using this proposed approach.

There is a chance of improving the reliability of the system by defect management system and also by providing effective erasure code for the distributed system. It is the future extension of this research work.

REFERENCES

[1] Dolbec,J.,Shepard,T,'A Component Based Software Reliability', Available at :http://citeseer. ist. psu. edu/ 136325. html,1995
[2] Hamlet,D, Mason.D, Woit.D, Theory of Software Reliability Based on Components', 23rd International conference on Software Engineering,Toronto,Canada,May,2001.
[3] Krishnamurthy , S. and Mathur,A.P. 'On the estimation of Reliabilty of a software system using reliabilities of its components',Proc. 68th International Symp. Software Reliability Engineering (ISSRE '97), Albuquerque,New Mexico,Nov.1997,pp 146-155.
[4] May,J.'Component-BasedSoftwareReliabilityAnalysis', Availableat: http://www.cs.bris. ac.uk /Publications/ Papers /1000623. pdf,2002
[5] Shukla,R.,Strooper,P, Carrington,D 'A Framework for Reliability Assessment of Software Components',Availableat: http://eprint.uq.edu.au/ archive/ 00001873/01/framework-cbse7.pdf,2005
[6] Wang,W., Wu,Y., Chen,M 'An Architecture-Based Software Reliability Model', proceedings of Pacific Rim International Symposium on Dependable Computing,1999, pp 143-150.
[7] Yacoub,S.,Cukic,B and Ammar, H., 'A Scenario-Based Reliability Analysis of approach for Component-Based Software' , IEEE Transactions on Reliability,Vol.53, No.4,2004,pp 465-48
[8] Dimitris S. Papailiopoulos and Alexandros G. Dimakis 'Repairing erasure code' Available a:http://static.usenix.org/event/fast11/posters_files/Papailiopoulos.pdf,2010
[9] Amuthakkannan, R., Kannan, S.M., Vijayalakshmi, K. and Jayabalan, V 'Managing change and reliability of distributed software system', Int. J. Information Systems and Change Management, Vol. 2, No. 1,2007, .30–49.

Study of QoS Management Techniques for Voice Applications

J.Faritha Banu[1] and V.Ramachandran[2]

***Abstract*—**In recent years, Voice over IP has influences on global telecommunication. Voice and multimedia applications delivery services remain a major and challenging task for network researchers and designers. VoIP packets are sensitive to packet losses and transmission delay to assure correct communicatio Quality of Services provisioning for coexisted and traffic flows in networks is still demanding problem. discusses Qos management techniques for VoIP applicati important techniques like flow classification, load balanc policies, congestion control and packet loss recovery studied. Finally, three routing policies like Single path r phase routing and multipath routing with forward erro are simulated using Ns2 simulator and the results ar identify the best routing policy.

***Keywords*—** Multipath routing, MPLS, QoS, Two phase routing, VoIP.

I. Introduction

VOICE over IP (VoIP) enables the voice and other multimedia real time sessions over IP-based networks with low cost. The critical problem is performing load balancing and assuring Quality of Service (QoS) for VoIP applications. Quality of VoIP applications is affected by basic network behavior, codec type, packet loss, delay and jitter. [1] QoS are managed by variety of mechanisms such as flow classification, load balancing, routing policies, admission control and traffic shaping, etc [2]. By classifying flows, it enables the users to learn more about the traffic and identify characteristics that are common to a set of flows. Due to this, the operator is able to manage flows from one class differently from another.

Common QoS Mechanisms for VoIP network and other real time applications are best effort services, integrated services (IntServ), differentiated services (DiffServ) and traffic engineering MPLS network [3]. Best effort service means that each user gets a fair share of the available network resources with no promise of QoS guarantees like delay, throughput and jitter. Intserv is based on IETF RSVP (Resource ReserVation Protocol) for QoS between end user nodes. Diffserv works among intermediate nodes on L2-L7 layers of OSI. The most important feature of "coloring" the IP traffic is the DSCP (Diffserv Code Point) field in the IP packet header. It provides services to aggregates classes and defines per hop behavior. A single flow receives the same QoS as of all the other flows in that aggregates. MPLS- TE networks provide the QoS level of service and best meets the VoIP nts of voice users.

gress interface needs classification, marking, haping (e.g. FIFO, FQ-Fair Queue, WFQ-Weighted e, WRED-Weighted Random Early Detected, "tail-Q-Low Latency Queuing), while the egress interface gestion avoidance, policing, and shaping tasks, y. The voice is classified as the traffic with the highest priority. Modeling packet switched networks is mainly characterized by manipulating only the packet arriving time series as a stochastic process [4].

QoS requirements of VoIP include packet loss, delay, and delay jitter. The current H.323 and SIP frameworks support some kind of interfaces to QoS management, but they do not provide functional QoS management mechanisms. To provide the QoS for multimedia applications several important techniques including packet classifier, buffer management, scheduling, loss recovery, and error concealment techniques, admission control, resource provisioning, traffic engineering, and connection management techniques are identified. Admission control plays a critical role in QoS for VoIP [5].

II. Related Approaches

A. Flow classification and Congestion Control Techniques

Traffic flows are categorized into distinct classes in order to control and manage highly aggregated Internet traffic flows efficiently. It also has the potential to support ISP provides to assure the required QoS for different users. Furthermore, real-time traffic classification is the core component of emerging QoS-enabled products and automated QoS architectures. To classify the flows in AQM queues an Active queue management (AQM) for non responsive traffic is proposed [6]. It is based on statistic measurement on the incoming traffic rate and the AQM packet loss rate. This scheme is implemented by estimating the average periodic cycle of the responsive traffic using a traffic estimator. Then, a wavelet de-noising filter is applied to remove the non-responsive traffic bursts before they enter the AQM queue. A wavelet de-noising filter uses a threshold function to remove the high peaks of traffic changes at different time scales. The thresholds should be set such that the resultant de-noised traffic is exactly the same as the responsive traffic. The

J.Faritha Banu is a research scholar, Faculty of Computer Science and Engineering, with Sathyabama University, Chennai, India. (e-mail: banujahir@gmail.com).

V.Ramachandran is working as a professor, Department of IST,with Anna University, Chennai, India (e-mail:rama@annauniv.edu).