CS 499 – B. Tech. Project

Reliability and Availability in Distributed Systems

by

Abhay Gupta (Y0002)

Santosh Kumar (Y0301)

Supervisors

Dr Bhaskaran Raman

Department of Computer Science and Engineering

IIT Kanpur

Dr Kishore S. Trivedi

Department of Electrical and Computer Engineering

Duke University

Acknowledgement

We would like to thank a number of people . We deeply appreciate valuable contribution of each of them. We are first of all, grateful to our guides Dr. Kishor S. Trivedi and Dr. Bhaskaran Raman. They volunteered many hours of their valuable time to help us put our best. Their patientence and encouragement have been the most important factors in meaningful completion of our project. Further, we sincerely appreciate Dr. S. Dharmaraja of Indian Institute of Technology, Delhi, who went out of his way in offering his full fledged support at some very crucial moments during this project. Our special thanks goes to all the researchers working with Dr. Trivedi, who from time to time offered us invaluable advice, ideas and corrections to our analysis. We would particularly like to mention, Wie Xie, whose work we have been extending, and Kesari Mishra, who offered us enormous help in installation and administration of polling agent at Duke. Finally, we are thankful to our evaluation committee members, Dr. Deepak Gupta and Dr. Dheeraj Sanghi, for helping us keep our focus and take this project to a meaningful completion.

Modeling of Web User Behavior and User Perceived Website Availability

Abstract

Depending upon the user behavior user perceived availability may differ from the actual system availability. Xei et al [5] studied the relationship between the system availability and the user-perceived availability and their distribution with change in system MTTF (Mean Time To Failure) and MTTR (Mean Time To Repair) using 3 different mathematical models of user-perceived unavailability,$\overline{A}_u$, called SUSH, SUMH and DPF models respectively as shown in Section 3. We corrected SUSH model and extended DPF model to bring it closer to the real world. We found out that unlike as claimed by Xie [5], CTMC analysis does not persistently overestimates $\overline{A}_u$. For lower repair rate, μ, CTMC analysis underestimates $\overline{A}_u$ and at higer repair rate it overestimates. We verified our claims from data obtained for user-server interactions with real web servers. We have obtained encouraging results, however, we need to go for more extensive data collection before drawing any conclusions. We modified DPF model to bring it more closer to real world by incorporating generally distributed transitions. This is a very complex model and not amenable to numerical solutions in its given form. So, we outlined solution for a simpler model leaving more extensive model's analysis for future work.

1 Motivation

The demand for high availability and performance has become critical to the success of e-business applications on the web. Depending upon the user behavior, e.g. repeat rate in the event of a failure, user perceived availability may differ from the actual system availability. Xei et al [1] studied the relationship between the system availability and the user-perceived availability and their distribution with change in system MTTF (Mean Time To Failure) and MTTR (Mean Time To Repair) using mathematical models of user-perceived availability.

The models built by Xie are Single-user-single-host model, Single-user-multiple-host model and Single-user-single-host model with different platform failures (namely near-user, in-middle and near-host failures). Henceforth, we call these models as SUSH, SUMH and DPF respectively. Depending upon whether the transitions between different states are taken to be all exponentially distributed or not, these models can be analyzed using CTMC or MRGP techniques.Xie in his study carried out both the analysis and concluded that CTMC models tend to overestimate the user perceived unavailability by 26% to 125% with MRGP as the reference, and that fast recovery after a failure is more important than higher component reliability in improving user-perceived availability.

Raymakers et al [6] simulated SUSH model and tried to verify the results. However, results of Raymakers et al [6] were quite different from those obtained by Xie [5]. We outline and compare their claims in Section 4. In our endeavor, we aimed to validate the hypothesis corresponding to the claims made by Xie et. al as well as results of Raymakers et al [6].

This report is organized as follows: Section 2 reviews the theory of stochastic processes which lie at the heart of mathematical modeling undertaken in this project. One can skip the rich mathematical details of this section and just refer to Section 5.1 where we outline the steps of solution as applicable to our case. In Section 3, we outline the models as given by Xie [5]. Later on we modify these models. A related work done by Raymakers [6], provided motivation for our work. We compare claims made by Xie vs Raymakers in Section 4. In Section 4.3, we have outlined our contribution amidst contributions of others. In Section 5, we begin our extensive analysis of SUSH model. One can safely skip the details in subsections 5.1 to 5.5 and directly go to Section 5.6. In Section 6, we have explained the issues pertaining to collecting data for real world user-server interactions. We outline our model of user-behavior in Section 6.1. We did this data collection from IITK server, so we account for behavior of proxy server in Section 6.2. In the next Section, i.e. 7, we estimate values of parameters from data we obtained. Results obtained from user-server interaction data have been shown in Section 8. Next, we explain various aspects and difficulty in extensive analysis of DPF model in Section 9. We outline solution for a simpler model in Section 9.1. We end up with summary of our work in Section **??** and give pointers for future work in Section 10.

2 Understanding of the Modeling Paradigms

We approached the problem of user percieved availability modeling from the beginning. First of all, we endeavored to understand the work done by Xie et. al. and Raymakers et. al.. A thorough understanding of analytical discrete modeling methodologies, such as CTMC, SMP and MRGP, is a precondition to this. Therefore, we studied relevant chapters from the red book, [1] and [2]. We review them below.

2.1 Stochastic Processes and Markov Process

Stochastic Process: *A stochastic process is a family of random variables* $\{X(t)|t \in T\}$*, defined on a given probability space, indexed by parameter t, where t varies over an index set* T.

The values assumed by the random variable $X(t)$ are called states, and the set of all possible values forms the **state space** of the process. If the state space of a stochastic process is discrete, then it is called a **discrete-state process** or a **chain**. Similarly, a continuous state space process is called **continuous-state process** or a **stochastic sequence**. We will always be concerned with Markov chains.

Markov Process: *A stochastic process* $\{X(t)|t \in T\}$ *is called a* Markov process *if for any* $t_0 < t_1 < t_2 < ... < t_n < t$*, the conditional distribution of* $X(t)$ *for given values of* $X(t_0), X(t_1), ..., X(t_n)$ *depends only on* $X(t_n)$*:*

$$P(X(t) \leq x|X(t_n) = x_n, ...X(t_0) = x_0) = P(X(t) \leq x|X(t_n) = x_n) \quad (1)$$

In the conditional distribution function in above has the property of invariance with respect to the time origin t_n:

$$P(X(t) \leq x|X(t_n) = x_n) = P(X(t - t_n) \leq x|X(0) = x_n), \quad (2)$$

then the Markov chain is said to be **(time-)homogeneous**.

2.2 Discrete and Continous Time Markov Chains

Discrete Time Markov Chain: *A stochastic process* $\{X_n, n \in I^N\}$ *with countable state space* Ω *is called a discrete time Markov chain(DTMC) with state space* Ω *if*

- *for all* $n \geq 0$, $X_n \in \Omega$ *with probability 1,*
- *for all* $i, j \in \Omega$ *and* $n \geq 0$, $P\{X_{n+1}=j|X_n = i, X_{n11}, ..X_0\}=P\{X_{n+1}=j|X_n = i\}$

If the above conditional probability is invariant with respect to time, n, then the Markov chain is said to be *homogeneous*. In this case, we denote by p_{ij} the probability P$\{X_{n+1}$=j$|X_n = i\}$ and **P**=$[p_{ij}]$ the transition probability matrix.

If the transitions between states can take place at any instant then, the Markov chain is known as *continuous time Markov chain(CTMC)*. The formal definition is as follows:

Continuous Time Markov Chain: *A stochastic process* $\{X_t, t \geq 0\}$ *with countable state space* Ω*={0,1,2....} is called a continuous time Markov chain(CTMC) with state space* Ω *if for any* $s, t \geq 0$, $j \in \Omega$,

P$\{X(t + s) = j|X(u); u \leq t\}$=P$\{X(t + s) = j|X(t)\}$

The probability,

$p_{ij}(t + s, t)$ =P$\{X(t + s) = j|X(t)\}$,

for $s, t \geq 0$, and $i, j \in \Omega$, is called the *transition probability* and the matrix

P(t+s,t)=$[p_{ij}(t + s, t)]$

is called the *transition probability matrix* of the CTMC.

Define,

$q_{ii}(t) \equiv \frac{\partial}{\partial t} p_{ii}(t + s, t)|_{s=0}$ for all $i \in \Omega$, and
$q_{ij}(t) \equiv \frac{\partial}{\partial t} p_{ij}(t + s, t)|_{s=0}$ for all $j \in \Omega$ and $i \neq j$.

The matrix **Q**(t)=$[q_{ij}(t)]$ is called the *infinitesimal generator matrix* or just *generator matrix*.
For time-homogeneous CTMC, the transition probability $p_{ij}(t + s, t)$ depends only on the time difference s. Therefore, the transition rates are independent of t. Hence, the *generator matrix* **Q** is a constant. Further, for a finite and *irreducible*(i.e. every state is reachable from every other in finite number of steps) time-homogeneous CTMC, it can be shown that the limits
$\pi_i = lim_{t\to\infty} p_{ij}(t) = lim_{t\to\infty} p_i(t)$
exist and are independent of the initial state j. These are called the steady state probabilities and if π represents the vector of these probabilities, then, it can be obtained by solving the equations

$$\pi Q = 0 \; and \; \pi e = 1 \quad (3)$$

where, e represents column vector with all enteries being 1.

In our case, we will encounter only finite, irreducible and time-homogeneous CTMCs and we will be interested in the steady state analysis. Hence, the strategy to solve

them is as follows:

1. Obtain the generator matrix Q, which will be a constant matrix.
2. Solve Eq. 3 to obtain the steady state probabilities.

2.3 Markov Regenerative Process

In this section we will provide a brief review of the theory behind Markov Regenerative Process and its solution techniques.

2.3.1 Markov Regenerative Theory

Consider a stochastic process in which there exist time points where the process satisfies the Markov property. These time points are called *regeneration points*. In a Markov regenerative process(MRGP) the stochastic evolution of the process between two successive regenerations depends only on the state at regeneration and not on the evolution before regeneration. Further, due to the time-homogeneity of the embedded Markov renewal process, the evolution of the MRGP becomes a probabilistic replica after each regeneration. As a result, all memory except the state must be reset at a regeneration point. The concepts of MRGP are elucidated in the following definitions.

Markov Renewal Sequence: A sequence of bivariate random variables $\{(Y_n, S_n),\ n \geq 0\}$ is called a Markov renewal sequence if

- $S_0 = 0, S_{n+1} \geq S_n; Y_n \in \Omega' \subset \Omega = \{0, 1, 2, ..\}$
- for all $n \geq 0$

$$\begin{aligned} P(Y_{n+1} = {} & j, S_{n+1} - S_n \leq t | Y_n = i, S_n, Y_{n-1}, ..., Y_0, S_0) \\ = {} & P(Y_{n+1} = j, S_{n+1} - S_n \leq t | Y_n = i) \\ & (MarkovProperty) \\ = {} & P(Y_1 = j, S_1 - S_0 \leq t | Y_0 = i) \\ & (Timehomogeneity) \end{aligned}$$

Then, the MRGP is defined as follows

Markov Regenerative Process:*A stochastic process, $\{Z(t), t \geq 0\}$ on Ω is called a MRGP if there exists a Markov renewal sequence$\{(Y_n, S_n),\ n \geq 0\}$ of random variables such that all conditional finite dimensional distributions of $\{Z(S_n + t), t \geq 0\}$ given $\{Z(u), 0 \leq u \leq S_n, Y_n = i\}$ are the same as those of $\{Z(t), t \geq 0\}$ given $Y_0 = i, i \in \Omega' \subset \Omega$.*

Note that the above definition implies that in this case$\{Z(S_n^+), t \geq 0\}$ or $\{Z(S_n^-), t \geq 0\}$ is an *embedded discrete time Markov chain(DTMC)* or just *embedded Markov chain (EMC)* in $\{Z(t), t \geq 0\}$, and also that S_n is a stopping time(regeneration points) of $\{Z(t), t \geq 0\}$.

We denote the conditional probability in Eq 2. by $K_{ij}(t)$, $i, j \in \Omega'$. The matrix **K(t)**=$[K_{ij}(t)]$ is called the *global kernel* of the Markov renewal sequence. Define the matrix **E(t)**=$[E_{ij}(t)]$,$i \in \Omega'$, $j \in \Omega$, as follows
$E_{ij}(t) = P(Z(t) = j, S_1 > t | Y_0 = i)$. This matrix describes the behavior of the MRGP between two transitions epochs of the EMC i.e over the time interval $(0, S_1)$. This matrix is known as the *local kernel*.

2.3.2 Steady State Solution of MRGP

In most of the problems as ours, steady state solution is of prime interest. Fortunately, this is also more tractable. We will consider the steady state behavior of MRGP by taking $t \to \infty$. For this we need to define two variables, $\alpha = [\alpha_{ij}]$ and the steady state probability vector $\nu = (\nu_k)$. α_{ij} represents the mean time that the MRGP spends in state j between two successive regeneration points, given that it started in state i after regeneration. Hence,

$$\alpha_{ij} = \int_0^\infty E_{ij}(t)dt \tag{4}$$

We define steady state probability vector as solution of the following set of equations

$$\nu = \nu P \ \ and \ \ \nu e = 1 \tag{5}$$

From these steady state probabilities one can obtain the user-perceived unavailability for SUSH model and for SUMH model as given in Section 3. where *e* is a vector with all enteries 1 and $P = K(\infty)$ is the one state transition probability of the EMC. The Theorem 1 of [2] describes the limiting behavior of MRGPs which we state below

Theorem 1: *Let $\{Z(t), t \geq 0\}$ be a MRGP on Ω with Markov renewal sequence $\{(Y_n, S_n), n \geq 0\}$ with Kernel K(.). Let N(t) denote the total number of state changes by time t, i.e. N(t)=sup$\{n \geq 0 : S_n \leq t\}$. Suppose that*

- *the sample paths of $\{Z(t), t \geq 0\}$ are right continuous with left limits,*
- *the semi-Markov process, $\{Y_{N(t)} \in \Omega' \subset \Omega, t \geq 0\}$ is irreducible, aperiodic and positive recurrent,*
- *$\nu = (\nu_k)$, is a positive solution of Eq. 5,*

then the steady state probability of the MRGP is given by

$$\pi_j = lim_{t\to\infty} P(Z(t) = j) = \frac{\sum_{k\in\Omega'} \nu_k \alpha_{kj}}{\sum_{k\in\Omega'} \nu_k \beta_k} \quad (6)$$
$$where\ \beta_k = \sum_{l\in\Omega} \alpha_{kl}$$

Limiting probabilities can, under some restrictions, be interpreted as the long run proportion of time the Markov chain spends in state j. Because those restrictions are trivially satisfied in our case, above theorem provides us with a method to obtain limiting probability vector from data collected for real world user-server interactions. From the data, we obtain the time spent by users in various states, sum them up and take ratios to obtain the limiting probabilities, which in turn are equal to steady state probabilities.

3 User Perceived Availability Models

In this section we briefly outline the models as given by Xie [5]. Later on we modify these models, however, here we give them as it was given by Xie. Xie et. al. [5] developed 3 models for capturing user-perceived availability two of them being MRGP models while the third being Stochastic Petri net(SPN) model. We briefly describe these models and unavailability definition applicable to each below.

3.1 Single-User-Single-Host: SUSH

This model captures the first scenario, where there is one online user, who is interested only in one host. Hence the name, Single-User-Single-Host(SUSH) model. Here, the user is *dedicated* to the host in a sense that, he will not switch to another site if the site of interest fails. He will keep retrying. In this model, failures occuring in different part of the system have not been distinguished. This is shown in Fig 1.Here, a state is designated by a tuple (server state, user state). Server state can be either **U**p or **D**own, while user state can be **A**ctive (when user has sent a request to the server and it is being fulfilled), **T**hinking (when user has recieved his desired objects and is going through them without making any further requests) and **F**ailure (when user request could not be fulfilled and the user *knows* that the server is down. We use the bold lettered alphabets to designate these states of server and user. We, thus, obtain state space to be $\Omega = \{(U,A),(U,T),(D,A),(D,T),(D,F),(U,F)\}$ for this model as shown in the figure. We also denote a particular state in Ω by its index in the set, starting at 1. Futher, the server's failure and recovery is assumed to have exponential time distribution with rates λ and μ respectively. These states are joined by transitions, F(.), G(.), R(.) and T(.). We explain these transitions and their nature in Section 8. The definition of availability used for this model is as follows:

$$\overline{A}_u \equiv \frac{\pi_{U,F} + \pi_{D,F} + \pi_{D,A}}{\pi_{U,F} + \pi_{D,F} + \pi_{D,A} + \pi_{U,A}} \quad (7)$$

where $\pi_{U,A}$ etc represent the limiting probabilities as explained in Section 2.3.2. Although, there are other definitions of availability, this seems to be the most suitable and widely accepted in the present context.

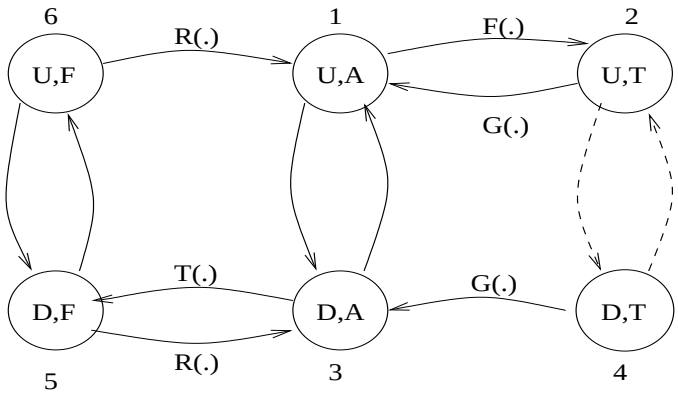


Figure 1: Single-User-Single-Host Model by Xie

3.2 Single-User-Multiple-Host: SUMH

This scenario includes one online user and a large number of hosts. Each of these hosts provide some service that the user is interested in. The user is not dedicated to a single host and therefore, in face of a failure of some host, he readily switches over to another host. Hence, the name single user multiple host(SUMH) model. To simplify analysis, it is assumed that the failure and recovery behavior of all the hosts in the pool is identical. In case an individual web host fails, the user may switch to another web host providing the service of interest. Switching time is assumed to be generally distributed with mean $1/\lambda$. Since, the unavailability of all web hosts is $\overline{A}_s$, the new web host may be down with probability of $\overline{A}_s$ and up with a probability of (1-$\overline{A}_s$). This model is shown in Fig 3.2. Note the transitions from (D,F) to (U,A) and (D,A) have aforesaid mentioned probabilities as a factor. The definition of availability used for this model is,

$$\overline{A}_u \equiv \frac{\pi_{D,F} + \pi_{D,A}}{\pi_{D,F} + \pi_{D,A} + \pi_{U,A}} \quad (8)$$

where $\pi_{U,A}$ etc represent the limiting probabilities as explained in Section 2.3.2.

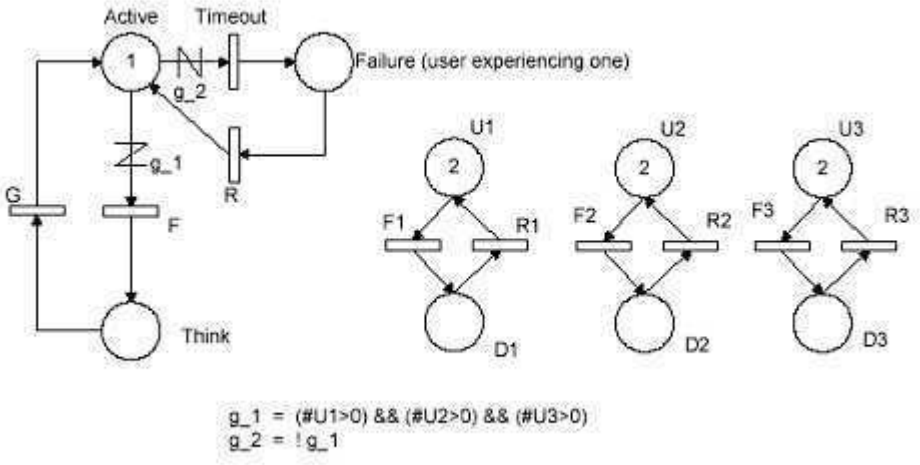


Figure 3: Single-User-Single-Host with Different Platform Failures

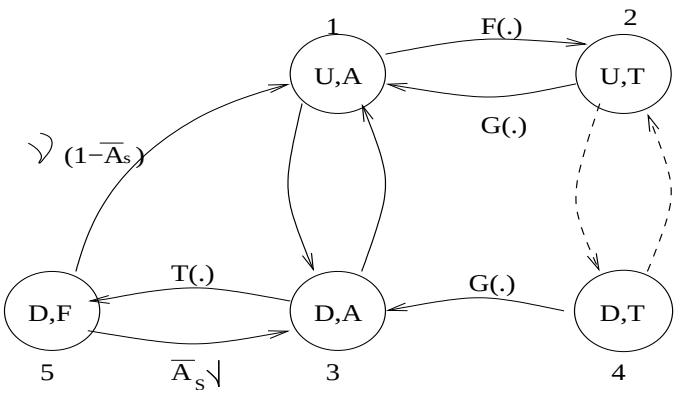


Figure 2: Single-User-Multiple-Host Model by Xie

3.3 Single-User-Single-Host with Different Platform Failures: DPF

In this model, we distinguish among failures occuring in different parts of the system namely, near-user, in-middle and near-host failures. This model of Xie is based on SPN and was an all exponential model. This is shown in Fig 3.3. Because of difficulty of MRGP analysis and large number of states in this model, we had to simplify this model in order to make it more amenable to MRGP analysis. The details on this model are given in Section 9.

4. Previous Work

In this section we state the conclusions of work done by Xie [5] and Raymakers [6] for objective comparision. Later we outline the work that we have done in this project.

4.1 Xie's Study

Following is the list of claims made by Xie et. al.

1. The user percieved system unavailability, $\overline{A}_u$, is different from actual system unavailability. User perceived system unavailability is a function of system's failure and recovery behavior as well as the user behavior.

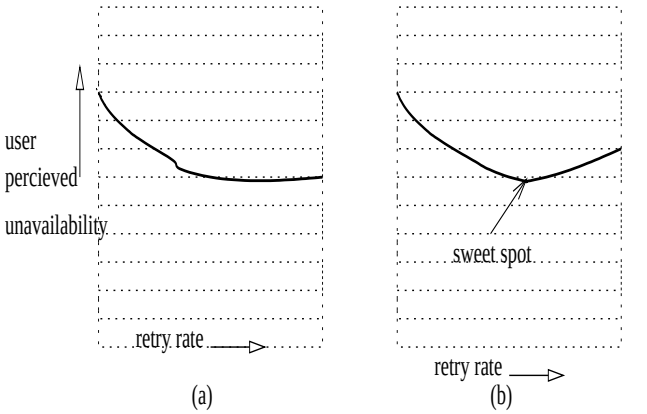
2. The relationship between user retry rate, λ_R, and $\overline{A}_u$ is of the nature as shown in Fig. 4.1(a).

3. For a given system unavailability $\overline{A}_s$, we have the following relationship

$$\lambda = \mu\overline{A}_s \tag{9}$$

where λ and μ represent failure and retry rate of the system. The equation says that to maintain the same actual platform unavailability, if μ is increased then λ must be increased and vice-versa. Xie et. al. concluded that for a given fixed $\overline{A}_s$, $\overline{A}_u$ decreases as mean time to failure, MTTF(λ^{-1}) & mean time to repair, MTTR(μ^{-1}) decrease. This conclusion can provide the web services administrators with handles to improve upon their system's user perceived availability while optimizing the costs.

4. Finally, CTMC analysis is not sufficient for user percieved platform availability results in much higher unavailability values. Therefore, MRGP analysis becomes essential.



(a) depicts the expected system unavailability as concluded by Xie et. al.
(b) depicts the observered system unavailability by Raymakers et. al.

for the case of fixed platform availability but variable MTTR and MTTF

Figure 4: Fig 1

4.2 Raymaker's Study

In their simulation experiments, Raymakers et. al. made the following claims:

1. We can find a "sweet spot", for a given system availability, beyond which higher user repair rates yield little benefit. $\overline{A}_u$ at a fixed $\overline{A}_s$ varies in the way shown in (b) above in fig 4(b), with decreasing MTTR i.e. the increasing μ.

2. For a given system, we can determine whether improving MTTR or MTTF will yield more user visible benefits.

4.3 Our Contributions

The three models of user interactions with servers were developed by Xie et al [5]. Xie et al [5] carried out the CTMC analysis as well as MRGP analysis of Model 1(SUSH) and Model 2(SUMH) and compared the results with CTMC analysis. For Model 3(DPF), they only carried out analysis using an all exponential assumption for the transitions between the states.
Raymakers et al [6] simulated Model 1(SUSH) with same parameters as those used by Xie et al [5]. They obtained results which were different from those of the later.

Our contributions are as follows:

- We have found out some problems with the Model 1(SUSH) by Xie et al [5]. We developed the new Model 1 as shown in Fig 5 and carried out MRGP analysis for the same. With this modification we were able to obtain results which were more similar to those of Raymakers [6].

- Further, we collected data from real world user and server interactions to evaluate the accuracy of claims made by Xie et al. [5].

- For DPF model, we have modified the model in a way so that it could be more tractable and carried out MRGP the analysis.

A quick summary of this is given in Table 1.

5 Single User Single Host (SUSH)

This model captures the first scenario where there is one online user who is interested only in one host. Hence the name, Single-User-Single-Host(SUSH) model. Here, the user is *dedicated* to the host in a sense that, he will not switch to another site if the site of interest fails. He will keep retrying. In this model failures occuring in different part of the system have not been distinguished.
An analysis of Xie's model 1(SUSH) revealed that certain facts were overlooked in the analysis of the model using MRGP. The fact that the exponential transitions with rates λ and μ between states **(U,T)** $\rightarrow$ **(D,T)** (refer to Figure 3.1 & 5) and **(U,F)** $\leftarrow$ **(D,F)** respectively, are of *concurrent* nature(refer to Appendix 11.3), modifies the *generator matrix* of the EMC of the MRGP. Hence, the *local kernel* $E(t)$ and the *global kernel* $K(t)$ also differ. We took notice of this fact which has been ignored in Xie's analysis and repeated the analysis.

5.1 Steps of Solution

We have carried out steady state analysis of the MRGP model 1 (SUSH) to find out the limiting probabilities from which the user perceived availability can be evaluated as shown in Section 3.1. Below we outline the steps to obtain the steady state probabilities:

1. Obtain the branching probability matrix Δ where $\Delta(i,k) = P$(next marking is $m_k \mid$ current marking is m_i and t which is an enabled transition in m_i, fires)

Table 1: Contributions of major groups in this project

	SUSH/Model 1	SUMH/Model 2	DPF/Model 3
Xie	MRGP Analysis	MRGP Analysis	CTMC
Raymakers	Simulation	–	–
Our Work	Correction & MRGP & Data Collection	-	MRGP

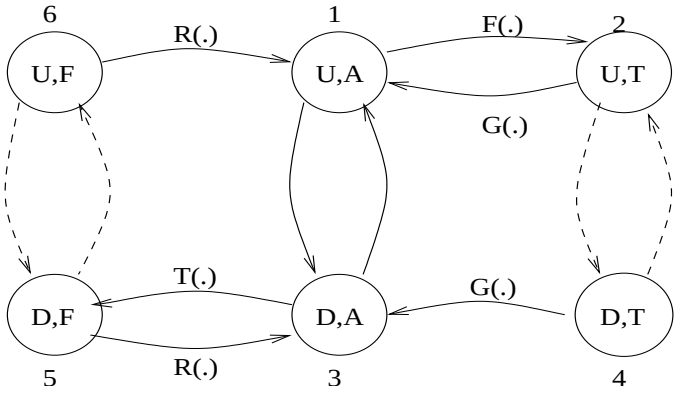


Figure 5: Corrected Single-User-Single-Host Model

2. Obtain the infinitesimal generator matrix Q of the subordinated CTMC of the MRGP.
3. Obtain the local kernel $E(t)$ using Theorem 4 of [1].
4. Obtain the global kernel $K(t)$ using Theorem 5 of [1].
5. Obtain matrix α where
   $\alpha_{mn} = E$[time spent in a state before a renewal $|$ $Y_0 = m$] = $\int_0^\infty E_{mn}(t)dt$
6. Obtain steady state probability vector $\nu = (\nu_k)$ using the equations
   $\nu = \nu P \quad \sum_{k\in\Omega'} \nu_k = 1$
7. Obtain the steady state probability of the MRGP given by
   $\pi_j = lim_{t\to\infty} P(Z(t) = j) = \frac{\sum_{k\in\Omega'} \nu_k \alpha_{kj}}{\sum_{k\in\Omega'} \nu_k \beta_k}$
   where $\beta_k = \sum_{l\in\Omega} \alpha_{kl}$

5.2 Branching Probability Matrix

There can be two kinds of states in MRGP: tangible, in which the process spends time > 0 with a non-zero probability and vanishing, in which the process spends time > 0 with zero probability. These vanishing marking are removed by merging all tangible states with vanishing states following it and reassigning the probabilities. This reassignment needs the branching probability matrix Δ. Since in our case, there are no vanishing states, therefore its a matrix with all enteries 1.

5.3 Infinitesimal Generator Matrix

Between any two renewals, the process behaves as a CTMC which can be solved after obtaining its generator matrix as explained in Section 2.2. This is one of the steps in obtaining the final solution of MRGP process as shown in Section 5.1. For this CTMC which underlies MRGP, the behavior will be dependent also on its initial state. The generator matrix with initial state m is called $Q(m)$. The various $Q(m)$s are given in 11.1. All these $Q(m)$s can be compactly represented as given below:

$$Q = \begin{pmatrix} -\lambda & 0 & \lambda & 0 & 0 & 0 \\ 0 & -\lambda & 0 & \lambda & 0 & 0 \\ \mu & 0 & -\mu & 0 & 0 & 0 \\ 0 & \lambda & 0 & -\lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & -\mu & \mu \\ 0 & 0 & 0 & 0 & \lambda & -\lambda \end{pmatrix} \tag{10}$$

5.4 Local Kernel E(t)

Local Kernel, as defined and explained in Section 2.3, for SUSH model, is the following matrix:

$$E(t) = \begin{pmatrix} E_{11}(t) & 0 & 0 & 0 & 0 & 0 \\ 0 & E_{22}(t) & 0 & E_{24}(t) & 0 & 0 \\ 0 & 0 & E_{33}(t) & 0 & 0 & 0 \\ 0 & E_{42}(t) & 0 & E_{44}(t) & 0 & 0 \\ 0 & 0 & 0 & 0 & E_{55}(t) & E_{56}(t) \\ 0 & 0 & 0 & 0 & E_{65}(t) & E_{65}(t) \end{pmatrix} \tag{11}$$

$E(t)$ represents the behavior of embedded stochastic process between two consecutive regenration points of an MRGP, which is a CTMC process.
A generalized method to obtain the local kernel is outlined in [1]. In our special case, it can also be obtained by analyzing the system as a two state continuous time Markov chain(CTMC). It's analysis as a CTMC yields the following results.

$$E_{11}(t) = e^{-\lambda t}(1 - F(t)) \tag{12}$$

$$E_{22}(t) = (1 - G(t)) \left(\frac{\mu}{\mu+\lambda} + \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)t} \right) \tag{13}$$

$$E_{24}(t) = (1 - G(t)) \frac{\lambda}{\lambda+\mu} \left(1 - e^{-(\lambda+\mu)t} \right) \tag{14}$$

$$E_{31}(t) = (1 - T(t)) e^{-(\mu)t} \tag{15}$$

$$E_{42}(t) = (1-G(t))\tfrac{\mu}{\lambda+\mu}\left(1-e^{-(\lambda+\mu)t}\right) \quad (16)$$

$$E_{44}(t) = (1-G(t))\left(\tfrac{\lambda}{\mu+\lambda}+\tfrac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t}\right) \quad (17)$$

$$E_{55}(t) = (1-R(t))\left(\tfrac{\lambda}{\mu+\lambda}+\tfrac{\mu}{\lambda+\mu}e^{-(\lambda+\mu)t}\right) \quad (18)$$

$$E_{56}(t) = (1-R(t))\tfrac{\mu}{\lambda+\mu}\left(1-e^{-(\lambda+\mu)t}\right) \quad (19)$$

$$E_{65}(t) = (1-R(t))\left(\tfrac{\mu}{\mu+\lambda}+\tfrac{\lambda}{\lambda+\mu}e^{-(\lambda+\mu)t}\right) \quad (20)$$

$$E_{66}(t) = (1-G(t))\tfrac{\lambda}{\lambda+\mu}\left(1-e^{-(\lambda+\mu)t}\right) \quad (21)$$

The generalized method for obtaining local kernel yields the following expressions:

$$E_{11}(t) = \left[e^{Q(1)t}\right]_{11}(1-F(t)) \quad (22)$$

$$E_{22}(t) = \left[e^{Q_2(t)}\right]_{22}(1-G(t)) \quad (23)$$

$$E_{24}(t) = \left[e^{Q_2(t)}\right]_{24}(1-G(t)) \quad (24)$$

$$E_{33}(t) = \left[e^{Q_3(t)}\right]_{33}(1-T(t)) \quad (25)$$

$$E_{42}(t) = \left[e^{Q_4(t)}\right]_{42}(1-G(t)) \quad (26)$$

$$E_{44}(t) = \left[e^{Q_4(t)}\right]_{44}(1-G(t)) \quad (27)$$

$$E_{55}(t) = \left[e^{Q_5(t)}\right]_{55}(1-R(t)) \quad (28)$$

$$E_{56}(t) = \left[e^{Q_5(t)}\right]_{56}(1-R(t)) \quad (29)$$

$$E_{65}(t) = \left[e^{Q_6(t)}\right]_{65}(1-R(t)) \quad (30)$$

$$E_{66}(t) = \left[e^{Q_6(t)}\right]_{66}(1-R(t)) \quad (31)$$

Here owing to their simplicity, $e^{Q(m)}$ can be exactly evaluated. We evaluated these matrices in 11.2. From them we re-evaluated the local kernel. We obtained results by this general method which were identical to those obtained previously, hence confirms the correctness of our analysis.

5.5 Global Kernel K(t)

Next, we proceeded to the *global kernel, K(t)*. This captures the behaviour of the embedded discrete time Markov chain formed by the regeneration instants, refer to Section 2.3.

Theorm 5 of [1] outlines the general method for obtaining global kernel. Using this method we obtain the following expressions for the global kernel

$$K_{12}(t) = \int_0^t \left[e^{Q(1)x}\right]_{11} dF(x) \quad (32)$$

$$K_{13}(t) = \left[e^{Q(1)}\right]_{13}(1-F(t)) + \int_0^t \left[e^{Q(1)x}\right]_{13} dF(x) \quad (33)$$

$$K_{21}(t) = \int_0^t \left[e^{Q(2)x}\right]_{22} dG(x)) + \int_0^t \left[e^{Q(2)x}\right]_{24} dG(x) \quad (34)$$

$$K_{23}(t) = K_{21}(t) \quad (35)$$

$$K_{31}(t) = \left[e^{Q(3)}\right]_{31}(1-T(t)) + \int_0^t \left[e^{Q(3)x}\right]_{31} dT(x) \quad (36)$$

$$K_{35}(t) = \int_0^t \left[e^{Q(3)x}\right]_{33} dT(x)) \quad (37)$$

$$K_{41}(t) = \int_0^t \left[e^{Q(4)x}\right]_{42} dG(x)) + \int_0^t \left[e^{Q(4)x}\right]_{44} dG(x) \quad (38)$$

$$K_{43}(t) = K_{41}(t) \quad (39)$$

$$K_{51}(t) = \int_0^t \left[e^{Q(5)x}\right]_{55} dR(x)) + \int_0^t \left[e^{Q(5)x}\right]_{56} dR(x) \quad (40)$$

$$K_{53}(t) = K_{51}(t) \quad (41)$$

$$K_{61}(t) = \int_0^t \left[e^{Q(6)x}\right]_{65} dR(x)) + \int_0^t \left[e^{Q(6)x}\right]_{66} dR(x) \quad (42)$$

$$K_{63}(t) = K_{61}(t) \quad (43)$$

In case, $R(t)$ is an exponential distribution then we have

$$K_{51}(t) = 0 \quad (44)$$

$$K_{53}(t) = \frac{\lambda_R}{\lambda_R+\mu}(1-e^{-(\lambda_R+\mu)t}) \quad (45)$$

$$K_{61}(t) = \frac{\lambda_R}{\lambda_R+\lambda}(1-e^{-(\lambda_R+\lambda)t}) \quad (46)$$

$$K_{63}(t) = 0 \quad (47)$$

With these expressions in place we compute the numerical values of various other matrices mentioned in 5.1 for various parameters and plot the graphs in the next section.

5.6 Results of Theoritical Analysis

The theoritical analysis of this model with same parameters as those used by Xie is being carried out. For this we first pin down the specific distributions assumed for $F(.)$, $G(.)$, $R(.)$ & $T(.)$. $F(.)$ represent the ON period for a user interacting with a server. This has been assumed to follow a *Weibull* distribution, refer to Section 6.1, with pdf and cdf given as follows:

$$f(t) = \tfrac{k}{\theta}\left(\tfrac{t}{\theta}\right)^{k-1}e^{-\left(\frac{t}{\theta}\right)^k} \quad t \geq 0 \quad (48)$$

$$F(t) = 1-e^{-\left(\frac{t}{\theta}\right)^k} \quad t \geq 0 \quad (49)$$

$G(.)$ represent the OFF period for a user interacting with a server. This has been assumed to follow a *Pareto* distribution, refer to Section 6.1, with pdf and cdf given as follows:

$$g(t) = \begin{cases} C\alpha m^\alpha/t^{\alpha+1} & m \leq t \leq n, \\ 0, & otherwise, \end{cases}$$

$$G(t) = 1-\left(\tfrac{m}{t}\right)^\alpha \quad m \leq t \leq n \quad (50)$$

$R(.)$ represents the user retry time distribution after he witnesses a failure. This has been assumed to have an exponential distribution with parameter λ_r. Finally, $T(.)$, which represents the HTTP retry time of the user agent, has been taken to be a constant with value $T = 10seconds$. Hence, its cdf is

$$T(t) = \begin{cases} 0 & 0 \leq t < T, \\ 1 & T \leq t \end{cases}$$

We obtain the solution of the MRGP model following the steps 5,6 & 7 of 5.1 using same values of parameters as those used by Xie. These parameters have been tabulated in Table 2. In Table 3, we tablulate the unavailabilities obtained for different μ and λ at a constant platform failure of $\overline{A}_u$=0.007.

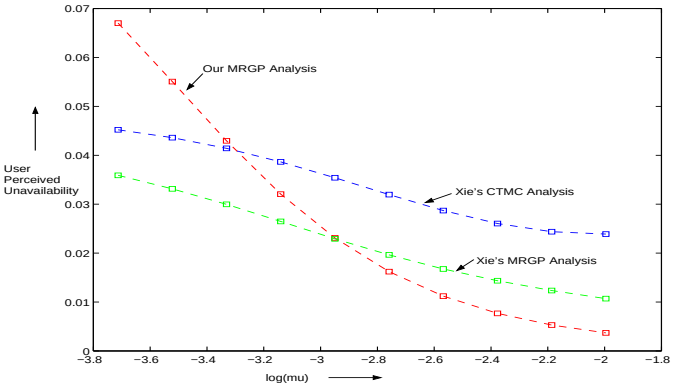


Figure 6: Unavailability Vs Repair Rate(log scale)

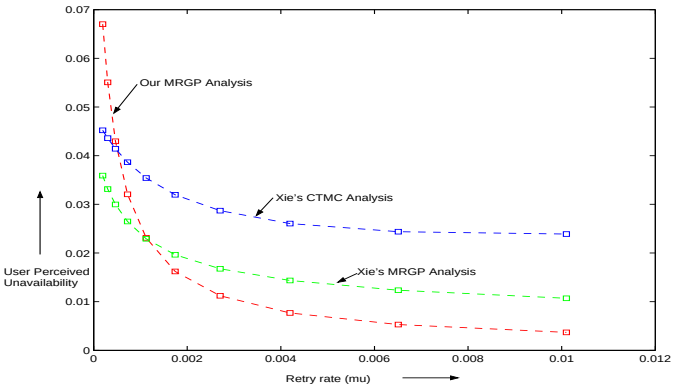


Figure 7: Unavailability Vs Repair Rate

The results obtained were markedly different from the results obtained by Xie. This is explained in 8

6 Data Collection for Experimental Verification of Results

Our next task was to collect data for verification of claims made by Xie [5] and Raymakers [6]. To accomplish this we decided to collect user data of the campus community for a set of 500 most highly accessed servers. We prepared this list on the basis of access logs of Vsnlproxy at IITK. We needed the following things for this purpose.

1. Knowledge of UP and DOWN states of servers being monitored
2. Knowledge of timing and duration of user requests to these servers

For the first part, it was necessary that we poll the servers from more than one location otherwise inaccessibility of the server from user's point of view will mostly coincide with the DOWN state of the server. Therefore, we set up pollers at Duke University. The working of the pollers is described in Section 6.3. We were able to collect 3 sets of data in this way, which we call data set 1, 2 & 3. There were many issues pertinent to real data collection for measuring user behavior. We throw some light on them in the following sections.

6.1 User Behavior

An empirical and tractable ON-OFF model of web user behaviour was proposed by Deng. The ON period follows a *Weibull* distribution with probability density function given by

$$f(t) = \frac{k}{\theta}\left(\frac{t}{\theta}\right)^{k-1} e^{-\left(\frac{t}{\theta}\right)^k} \tag{51}$$

Here constants k and θ are referred as the *shape parameter* and scale parameter of the Weibull distribution. Their typically used values are $k = (0.77 - 0.91)$ and $\theta = \left(e^{4.4} - e^{4.6}\right)$. The OFF period follows a *Pareto* distribution with pdf

$$g(t) = \begin{cases} C\alpha m^{\alpha}/t^{\alpha+1} & m \leq t \leq n, \\ 0, & otherwise, \end{cases}$$

Here, α, m, n are constants with typical values of $\alpha = (0.5 - 0.9)$ (α is called the shape parameter of the Pareto distribution.) m is called the "ON-OFF" threshold and is also called the shape parameter of Pareto distribution which means that a series of requests arriving with an inter-interval times within m will be considered to be an ON period while the requests seperated by more than m time constitute an OFF period between them. n is called session threshold which indicates that the requests separated by more than n time are considered to be of different sessions.

Table 2: Summary of SUSH Model Parameters

parameter	value	Comments
k	0.88	shape parameter of Weibull distribution
θ	$e^{4.5}$	scale parameter of Weibull distribution
α	0.5	shape parameter of Pareto distribution
m	60 seconds	scale parameter of Pareto distribution (ON-OFF threshold)
n	6000 seconds	truncation point of Pareto distribution (session threshold)
$1/r$	100 seconds	Mean time between user retries upon failures
T	10 seconds	HTTP Retry rate

Table 3: $\overline{A}_u$ Vs Platform Repair rate μ with $\overline{A}_S$=0.007 in SUSH model

μ	Our $\overline{A}_u$	Xie's MRGP $\overline{A}_u$	Xie's CTMC $\overline{A}_u$
0.00019372	0.067029	0.03589534	0.04520818
0.00030059	0.055061	0.03313437	0.04359795
0.00046642	0.042960	0.02997184	0.04142308
0.00072373	0.032062	0.02647339	0.03865829
0.00112299	0.023094	0.02291812	0.03541084
0.00174250	0.016211	0.01963503	0.03195471
0.00270377	0.011197	0.01676101	0.02869206
0.00419535	0.007683	0.01434502	0.02604848
0.00650979	0.005288	0.01234880	0.02437117
0.01010101	0.003690	0.01068826	0.02389412
0.02020202	0.003620	–	–

To start with, we will use this model with a slight modification.

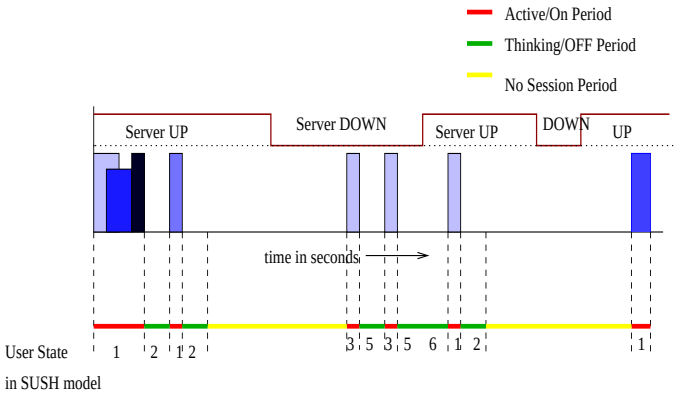


Figure 8: User State Determination Diagram

We will assume that an ON period is the period between two subsequent non-overlapping requests within a time interval of $< n$. We will merge all over-lapping requests into a single request. Hence, we do have $m = 1$ in our model. Two request separated by more than n units will be considered to be belonging to two different sessions and only a small portion of this time is considered to be the OFF period, and is called the thinking time threshold (it partly plays the role of m). For this we need to estimate two parameters i.e. n and the thinking time threshold, which we do in 7.3.

Now, it's quite simple to deduce the state transition sequence for a given user-server pair from the knowledge of timing and duration of user requests to these servers (obtained from access.log proxy files) and the knowledge of UP and DOWN states of servers being monitored (obtained from data of polling agents). We show this using diagram in Fig. 8. This is quite simple and intuitive and can be easily understood from this diagram. Here, states (U,A) etc are denoted by their index in set Ω, as explained in Section 3.1. In this figure, filled vertical bars, in shades of blue, represent user requests. This figure is not to scale however.

data collected

6.2 Summary of Proxy Server Behavior

In Appendix 11.4, we outline the behavior of proxy server to the three kinds of requests made by an user. From this analysis we conclude that we need to retain only the following enteries in the access log files obtained from the

proxy server

1. TCP_MISS
2. TCP_REFRESH_MISS
3. TCP_CLIENT_REFRESH_MISS

6.3 Server Polling Agent

We will call the agent that will poll the servers for their UP and DOWN state as Server Polling Agent or SPA. This agent will poll a set of predefined servers at regular intervals. We will use HTTP HEAD request to poll the servers. Previously, people have used the ubiquitous ping program, ICP ping, rpc.statd[1] to find out whether a particular server is up or down at a given instant of time. All these methods are much more efficient than polling using HTTP HEAD request. However, in each of these earlier studies the aim was just to see whether the system was up or down. We are interested in Web-User behavior modeling. In this context, UP state of a server is that state in which the user is getting a response from the server which belongs to a category of HTTP requests other than the 5xx response (5xx responses imply that there was some server error due to which the server could not complete the request of the user). Whether the server is in UP state or not can only be found out only after analyzing the response of the server (or browser in cases such as that of request time outs).
Its working is follows:

Send a HTTP HEAD request to the server
parse the response
if response code is obtained before time out **then**

  if response code is one corresponding to UP state **then**
    store the current time instant to be an UP instant for the server
  else
    store the current time instant to be a DOWN instant for the server
  end if
else
    store the current time instant to be an DOWN instant for the server (this corresponds to time out of responses such as Requested URL could not be retrieved)
end if

6.4 Servers To Be Polled

Our work is being is done within the boundary of an educational campus. This might not be the representative of the web user behavior in general. However, since most of the people using internet heavily are related to academics, therefore, the user behavior of the limited set of users on the campus will be representative of user behavior for some specific sites related to academics such as citeseer.com, dblp.com etc. However, this approach can be generalized to a wider study of web user behavior. We will outline some scalability issues with our approach and ways to deal with them later on. To get a list of servers to be polled we will analyze the web logs of proxy server at IIT Kanpur. From this a list of 500 most highly accessed sites will be shortlisted (these numbers are arbitarily decided and subject to changes). Time Synchronization

Table 4: Characterization of Poller's Behavior

Time Diff	Avg Time for HTTP Trans	Avg # of Conn
5	8.78	1.33
4	7.74	1.92
3	7.45	2.44

6.5 Issue of Time Synchronization

We installed the polling agent at different locations to obtain more reliable data regarding UP and DOWN states of the servers being polled. The log files of up and down times thus generated will be merged off-line in order to generate the more actual up and down time behavior of the server. This brings us to time synchronization problems as the systems on which we run our programs may differ in their time. To deal with this, we will use the Network Time Protocol and inquire about the UTC time at a chosenNTP Server. We will a poll a stratum 2 NTP server or a secondary NTP server as primary NTP servers take longer to respond and the differences between the two is of the order of a second at most, which is bearable in our case atleast for now.

6.6 Time difference between polling

We will use separate threads to poll different servers of interest. However, this might lead to a large number of simultaneous TCP connections. To prevent this we will run threads for different servers at a separation of a few seconds. Below in the table we give the mean time taken to get a response to HTTP HEAD request and average number of TCP connections for different values of time interval between starting of polling threads. The analysis was done for a about 500 servers and average values gathered over 100 runs. This table helped us in determining the optimum time interval. Its first column contains the difference between starting of consecutive request threads (in seconds), second column contains average time taken for HTTP HEAD request-response transaction to complete(in seconds) and the last column contains the average number of simultaneous TCP connections opened by SPA.

6.7 Scalability

The current system of server polling agents have not been optimized to scale to the size of the web. However, there are tools to do this well. [2] mentions of one such hardware tool that can effectively deal with huge amounts of traffic and corresponding log records without encurring much overheads. Hence, we hope that our efforts at modeling the user behavior can be scaled over larger networks by replacing this subsystem with a more efficient and dedicated hardwares.

7 Data Analysis & Results

In this section, we do the analysis of data collected and estimate their parameters. We have assumed that the data collected follows the distributions assumed for them. We have verified this for distributions of F(.) and G(.). For other distributions we did not have enough data. In the first subsection we estiamate parameters of exponential distributions involved i.e λ, μ and λ_R. In next subsection we estimate parameters of Weibull distribution F(.). Refer to chapter 10 of Trivedi's book for details [4]. We plot pdf using our estimates and show their extent of overlap in Fig. 9 for the first data set. We do the same for Pareto distribution in subsection 7.3 and plot the graph in Fig. 10

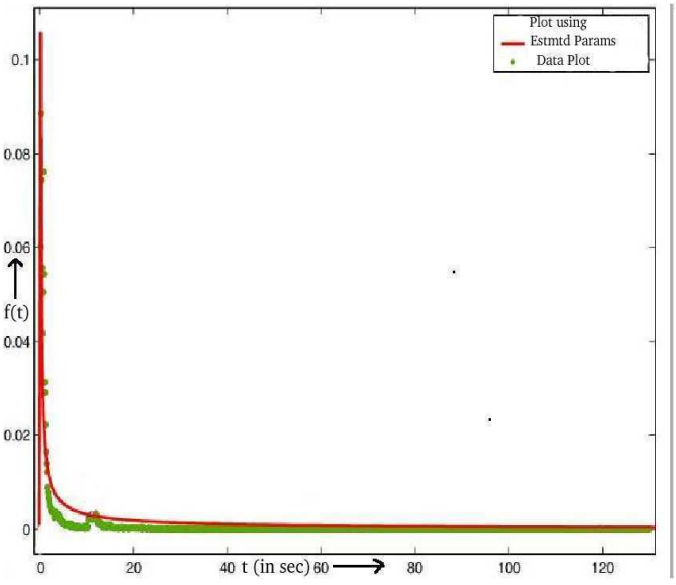


Figure 9: Distribution of User-ON periods, F(.)

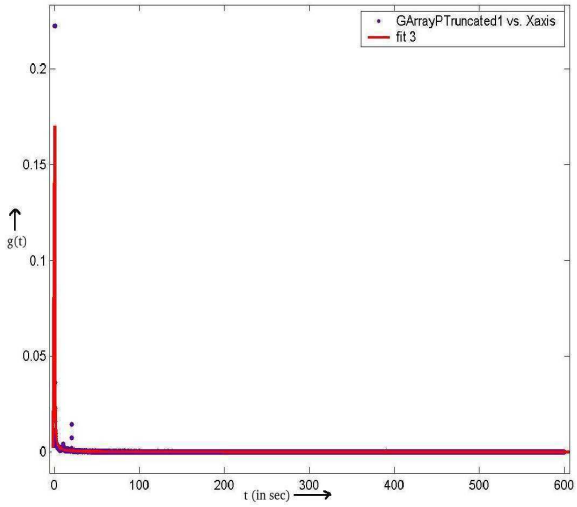


Figure 10: Distribution of User-OFF periods, G(.)

7.1 Estimating Parameters of Exponential Distributions λ, μ & λ_R

We use *maximum likelihood estimates* for obtaining these parameters. This involves defining a *likelihood function* and finding its maximum value by partially differentiating it with respect to the parameters, refer to chapter 10 of [4]. We, thus, obtain the following estimate for λ

$$\lambda = \frac{n}{\sum_{i=1}^{n} X_i} \tag{52}$$

This is applicable for estimating μ and λ_R, but not for λ. This is so because our period of data collection is not large and in this period we could not witness failure of most of the servers we polled. Therefore, we will use the method of *truncated life tests* for estimating these parameters as outlined in chapter of 10 of [4]. This analysis gives the following estimate of λ

$$\lambda = \frac{r}{\sum_{i=1}^{r} T_i + (n-r)T_r} \tag{53}$$

where $T_1, T_2, ..., T_r$ are the observed times to failure so that $T_1 \leq T_2 \leq ... \leq T_r$ and r is the number of servers which failed during the observation period out of the total n servers polled. The values of λ estimated using this method has been tabulated in Table 5. Similarly, estimates of μ and λ_R using Eq. 52 have been tabulated in the same table.

Table 5: Estimation of Parameters From Data

Set #	$\lambda(sec^-1)$	$\mu(sec^-1)$	$\lambda_R(sec^-1)$	k	θ	α	$\overline{A}_u$(Theory)	$\overline{A}_u$(Data)	η
1	$1.3596 * 10^{-4}$	$1.0392 * 10^{-2}$	$5.802 * 10^{-3}$	0.2738	$1.94 * 10^4$	0.04712	0.1734	0.1389	1.24838
2	$5.2348 * 10^{-4}$	$9.863 * 10^{-3}$	$8.242 * 10^{-3}$	0.2456	$4.55 * 10^4$	0.0615	0.1368	0.2015	0.67890
3	$7.984 * 10^{-3}$	$4.0238 * 10^{-2}$	$6.563 * 10^{-3}$	0.3456	$7.23 * 10^3$	0.0598	0.2568	0.2356	1.08998

7.2 Estimating Parameters of Weibull Distribution

For estimating the parameters of Weibull distribution, we follow the maximum likelihood estimation method outlined in chapter 10 of Trivedi's book [4]. We obtain the following set of equations in terms of parameters θ and k

$$\frac{n}{\theta^k} - \sum_{i=1}^{n} T_i^k = 0 \quad (54)$$

$$\frac{n}{k} + \sum_{i=1}^{n} lnT_i^k - \theta^k \sum_{i=1}^{n} T_i^k \, lnT_i = 0 \quad (55)$$

There are no closed-form solutions for θ and α. These have to be solved iteratively.

7.3 Estimating of Parameters of Pareto Distribution

For estimating the parameters of Pareto distribution, we follow the maximum likelihood estimation method outlined in Trivedi [4]. We obtain the following equation in terms of parameters m and α

$$\alpha = \left[\frac{\sum_{i=1}^{n} lnT_i}{n} - ln(k) \right]^{-1} \quad (56)$$

These are again to be solved using iterative numerical techniques. The values obtained for the different data sets have been tabulated in Table 5. For this distribution, the session threshold time has been assumed to be 6000 seconds as these values were used by Xie [5] in his analysis.

7.4 Results

We plotted the graphs corresponding to estimated parameters for F(.) and G(.) along with the real distribution obtained from data in Fig 7 and 7 respectively for data set 1. However, our values for the parameters are quite different from those given in Xie [5]. This should be anticipated as the campus community is not the representative of the web user community in general. The close agreements between them show that our assumptions regarding nature of F(.) and G(.) are valid.

Our next task was to compare the theoritically obtained user perceived unavailibities values with those obtained from real data. For this we made use of Theorem 1 of [2], which we have stated in Section 2.3. The theorem and the fact that limiting probabilities under suitable conditions, which are satisfied for our models, are same as the long run proportion of time a Markov chain spends in state j, can be used to evaluate limiting probabilities. We assume that our period of observation of data is long. Therefore, we obtain limiting probabilities by evaluating long run proportion of time spent in various states by users and from these using unavailability definition of Section 3.1, we obtain actual values of unavailibilities. This we have tabulated against the steady state probabilities obtained by theoritical analysis using parameters estimated from data in Table 5.We found good agreements betweent the user-perceived availability values obtained by theoritical analysis and from data. This is quite encouraging given the short time interval of data collection, which increases discrepancies. More extensive study can establish the fact that our methodology is indeed quite reliable.

8 Conclusions

We have plotted the graphs showing variation of user unavailability, $\overline{A}_u$, versus increasing repair rate, μ, in Fig. 6(log μ) and Fig. 7of section 5.6. We have also plotted the values obtained for MRGP and CTMC analysis by Xie. The results obtained by us are quite different from those of Xie [5]. Our analysis clearly establishes the following facts:

- Insufficiency of CTMC Analysis: CTMC analysis is not sufficient for the purpose of user-perceived availability analysis because of large difference in estimates obtained by the two methods. This establishes that we must do MRGP analysis for DPF model inspite of its difficulties.

- **Presence of sweet spot:** There does exist a sweet spot in the graph of $\overline{A}_u$ vs μ, which implies that efforts to improve MTTR (μ^{-1}) beyond sweet spot, at a given system availablity, does not provide much dividends, because afterall its the user-perceived availability thats important and not actual system avalibility. This can serve as a useful handle for web administra-

tors to fine tune their efforts to improve overall availability of webservers.

- **Close agreements with real data:** We were able to establish that distributions of F(.) and G(.) are indeed of the kind assumed in the theoritical analysis. Moreover, we found good agreements betweent the user-perceived availability values obtained by theoritical analysis and from data as shown in Table 5. More extensive study can establish the fact that our methodology is indeed quite reliable.

9 Single User Single Server With Different Platform Failures

This scenario includes one web user and one service host. The user is dedicated to the host, meaning that the user will not switch to other sites even in the presence of service outage. The model distinguishes the *near-user*, *in-middle* and *near-host* failures. Due to the complexity of the model, originally it was proposed and solved by Xei et al [5] as a Stochastic Petri Net (SPN) as shown in 3.3.

Assume there are two components in each of the near-user, in-middle, and near-host subsystems, which are independent of each other. Places U1, U2 and U3 represent the ”up” place of the near-user, in-middle and near-host components, respectively. Places D1, D2 and D3 represent the ”down” place of the near-user, in-middle, and near-host components, respectively. Transitions F1, F2 and F3 are the corresponding failure transitions, and transitions R1, R2 and R3 are the corresponding repair or recovery transitions. The user could be in any of the Active, Think and Failure places, and there are guard functions that control the transitions among them.
The limitation of a SPN is that we have to let all transitions have exponentially distributed firing times. To overcome this limitation in the originally proposed model, we have constructed a Markov Regenerative Process (MRGP) to represent the system. MRGP allows us to have at most one non-exponentially distributed transition from every state in the model. We denote a state of the MRGP by the 4-tuple (U1, U2, U3, U), where U1, U2, U3 denote the number of ”up” components in near-user, in-middle and near-host parts and U denote state of the web user. Since there are two independent components in each part of the network, there are three possible values for each of U1, U2 and U3, i.e. each can be 0, 1 or 2 depending upon the number of components ”up” in each part of the network. Also, user can be either Active, Thinking or seeing a Failure so there are three possible values for U, denoted A, T or F respectively. So the MRGP model has a total of 81 states and 405 transitions.
Due to complexity of the model, a simplified version is shown in the figure below. The simplified model assumes only 1 component in each platform, thus, leading to total number of states being reduced to 24 and transitions to 96. A state labeled 010A above denotes all components down in near-user part as well as near-host part, 1 component up in in-middle part and the user is in Active state. All transitions among the states, in which user is in thinking state, are concurrent transitions since user does not get to know about these transitions and hence, the clock is not reset on any of these transitions. A similar argument could be given for all transitions among states in which user is seeing a failure. The transitions among those states in which user is in active state, while all the components in one or more platforms are down, are also concurrent. This holds since the user’s request does not get completed and the clock for T(.) starts as soon as user enters any of these 7 states and it is not reset on any of these transitions. All of these concurrent transitions have exponential distribution. Hence, in the above model we have 3 subordinated CTMCs. But in the original model, there exists another such subordinated CTMC consisting of those states in which user is in active state and at least one component is up in each part of the network thereby allowing for the user request to be completed.

In the above model, the transitions from active states to corresponding thinking states are prohibited when at least one platform has all components down. This is in accordance with the guard function g1 in the original SPN. Similarly, the transitions from active to failure states when there is at least one path are disallowed by guard function g2 in the SPN.

In the above model, the only way to enter a thinking state from a non-thinking state is through transition from ”111A” to ”111T”. Since the distribution for this transition is general and for all other transitions with at least one platform with all components down, the distribution is exponential hence we conclude that, all the states in which user is in thinking state and there is at least one platform with all components down are Non-Regenerative (NR) states. Now, since any state in which user is thinking and at least one component is up in every platform, can be reached from a state having same number of components up in respective platforms and with user in active state, through a non-exponentially distributed transition, hence all such states are regenerative at the instant of occurrence of that transition and hence are Regenerative (R) states. These statements are also true for the original model with 81 states. A similar argument can be given for states in which user sees a failure, but then those states in which

there is at least one path from user to host are NR while all the remaining failure states are regenerative states. Any active state can be reached through a non-exponentially distributed transition from corresponding thinking or failure state with same number of components up in each part of the network. Hence, all the active states are regenerative.
Each of the subordinated CTMCs for the original model with 81 states is shown below. The model for subordinated CTMC within those states in which the user is in thinking mode is same as that in which user sees a failure.

A state labeled "xyz" here denotes the state when user is in thinking state (or failure for CTMC for failure states) and x components are up in near-user part, y components up in in-middle and z components are up in near-host platform in the network.
The CTMC above consists of all those states in which user is active and there is at least one path for user request to reach the host, and all the transitions among these states in the original model.
The CTMC above consists of all those states in which user is active and all components are down in at least one part of the network, and all the transitions among these states in the original model.

9.1 Solution of Model 3

To solve the model, we need to construct the local kernel matrix $E(t)$ and global kernel matrix $K(t)$ as for SUSH model. Since the MRGP model had 81 states, $K(t)$ and $E(t)$ are 81X81 square matrices. Although most of the entries in the kernel matrices are zero, there are 2059 non-zero entries in $K(t)$ and 1883 in $E(t)$. These entries in-turn depend upon distributions of $[e^{Q(i)t}]_{ij}$ for states i,j, which can be obtained by alternatively by solving Kolmogorov's equation

$$P'(t) = QP(t) \tag{57}$$

for each subordinated CTMC.

For instance, for the subordinated CTMC formed by the concurrent transitions in the thinking states in the simplified model (1 component in each platform), we obtain the following set of 8 differential equations:

$$\begin{aligned} P000'(t) &= -(r1+r2+r3)P000(t) + f1P100(t) \\ &\quad +f2P010(t) + f3P001(t) && (58)\\ P001'(t) &= -(r1+r2+f3)P001(t) + f1P101(t) \\ &\quad +f2P011(t) + r3P000(t) && (59)\\ P010'(t) &= -(r1+f2+r3)P010(t) + f1P110(t) \\ &\quad +r2P000(t) + f3P011(t) && (60)\\ P011'(t) &= -(r1+f2+f3)P011(t) + f1P111(t) \\ &\quad +r2P001(t) + r3P010(t) && (61)\\ P100'(t) &= -(f1+r2+r3)P100(t) + r1P000(t) \\ &\quad +f2P110(t) + f3P101(t) && (62)\\ P101'(t) &= -(f1+r2+f3)P101(t) + r1P001(t) \\ &\quad +f2P111(t) + r3P100(t) && (63)\\ P110'(t) &= -(f1+f2+r3)P110(t) + r1P010(t) \\ &\quad +r2P100(t) + f3P111(t) && (64)\\ P111'(t) &= -(f1+f2+f3)P111(t) + r1P011(t) \\ &\quad +r2P101(t) + r3P110(t) && (65) \end{aligned}$$

Similarly, we will get a system of 27 differential equations for corresponding subordinated CTMC for our original MRGP. On solving this system of equations, we can get the analytical solution for $P_{ij}(t)$ or equivalently $[e^{Q(i)t}]_{ij}$ for states i,j. This can be then used to calculate the kernels. But, finding analytical solution by solving such a system of equations for subordinated CTMCs with a large number of states does not seem promising. This coerced us to instead find the numerical solution for $P_{ij}(t)$, for which several tools are available. We used SHARPE for this purpose. We were able to construct the model using tool SHARPE and find $P_{ij}(t)$ for some states i,j. But solving the model completely could not be possible considering the immense number of transitions and time constraints.

10 Future Work

The problems with the structure of the models were discovered much later during the project and after that we had to start from almost the begining. There are many more things that we need to do. We outline them below:

1. **More Extensive Data Collection** We would like to carry out more extensive data collection and thereby verifying our models for the extent of accuracy with the real world. This should be easy now, as we have all the infrastructure and tools in place to do this.

2. **SUMH Model's Analysis** This model is very similar to SUSH model. However, due to short of time we could not carry out its analysis.

3. **DPF Model's Enhancements** We had to simplity this model to render it mathematically tractable. We would like to carry out its analysis in its original form using MRGP techniques.

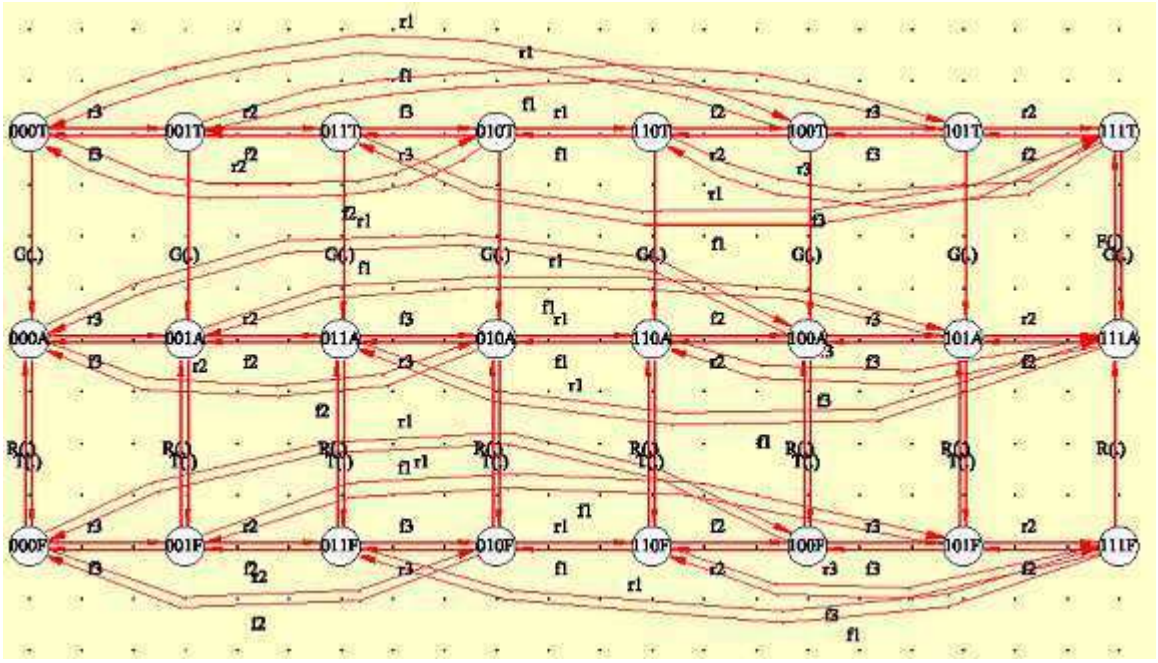


Figure 11: DFP model for reduced number of parameters (components in each part of the network).

11 Appendix

11.1 Generator Matrices

Between any two renewals, the process behaves as a CTMC. The generator matrix with initial state m is called $Q(m)$. Following the method outlined in [1], we obtain various $Q(m)$s to be as follows:

$$Q(1) = \begin{pmatrix} -\lambda & 0 & \lambda & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{66}$$

$$Q(2) = Q(4) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & -\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{67}$$

$$Q(3) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \mu & 0 & -\mu & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{68}$$

$$Q(5) = Q(6) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\mu & \mu \\ 0 & 0 & 0 & 0 & \lambda & -\lambda \end{pmatrix} \tag{69}$$

11.2 Exponentiation of Generator Matrices

In this section we show that owing to their simple structure $e^{Q(m)t}$ indeed can be obtained in a closed form for all the $Q(m)$ matrices. We use power series to define exponentiation of a matrix as follows:

$$e^{Q} = I + \frac{Q}{1!} + \frac{QQ}{2!} + \frac{QQQ}{3!} + ... \tag{70}$$

We show how we obtained $Q(2)$, rest can be done by following the reckoning.

$$Q(2) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\lambda & 0 & \lambda & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & -\mu & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{71}$$

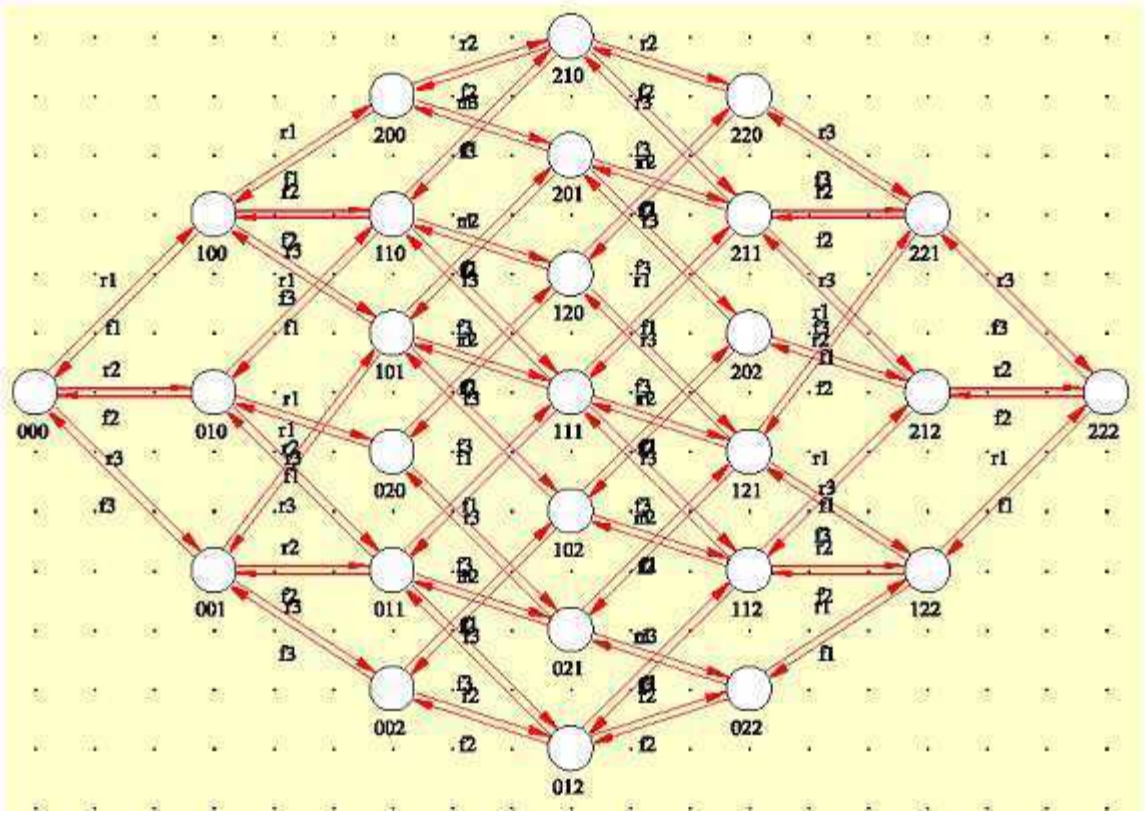


Figure 12: DPF model: Subordinated CTMC formed due to concurrent transitions in user thinking states. (Similar model for Failure States.)

$$Q(2)Q(2) = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \lambda(\lambda+\mu) & 0 & -\lambda(\lambda+\mu) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\mu(\lambda+\mu) & 0 & \mu(\lambda+\mu) & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{72}$$

We obtain the general formula for $Q(2)^n$ as follows

$$Q(2)^n = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q^2_{22} & 0 & Q^2_{24} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & Q^2_{42} & 0 & Q^2_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{73}$$

where,

$$Q^2_{22} = (-1)^n \lambda(\lambda+\mu)^{n-1} \tag{74}$$
$$Q^2_{24} = (-1)^{n-1} \lambda(\lambda+\mu)^{n-1} \tag{75}$$
$$Q^2_{42} = (-1)^{n-1} \mu(\lambda+\mu)^{n-1} \tag{76}$$
$$Q^2_{44} = (-1)^n \mu(\lambda+\mu)^{n-1} \tag{77}$$

Now, using equation 70 we have

$$e^{Q(2)} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & [e^{Q(2)}]_{22} & 0 & [e^{Q(2)}]_{24} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & [e^{Q(2)}]_{42} & 0 & [e^{Q(2)}]_{44} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \tag{78}$$

where,

$$[e^{Q(2)}]_{22} = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)} \tag{79}$$

$$[e^{Q(2)}]_{24} = \frac{\lambda}{\lambda+\mu}(1 - e^{-(\lambda+\mu)}) \tag{80}$$

$$[e^{Q(2)}]_{42} = \frac{\mu}{\lambda+\mu}(1 - e^{-(\lambda+\mu)}) \tag{81}$$

$$[e^{Q(2)}]_{44} = \frac{\mu}{\lambda+\mu} + \frac{\lambda}{\lambda+\mu} e^{-(\lambda+\mu)} \tag{82}$$

Similarly, we obtain other exponentiations.

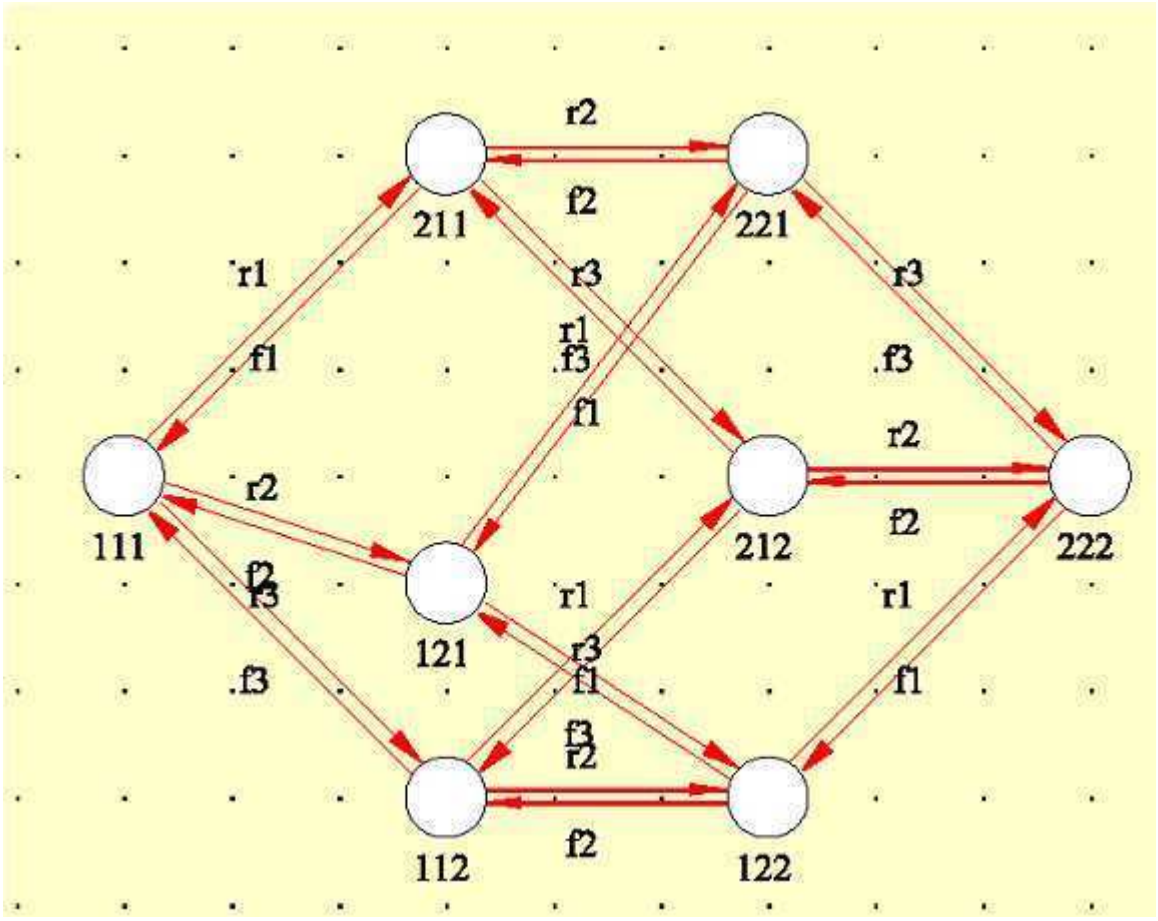


Figure 13: DPF model: Subordinated CTMC formed by concurrent transitions in Active Up states.

11.3 Concurrent and Competitive Transitions

11.4 Proxy Server Behavior

In IITK system, almost all computer users on the campus lan access Internet through the proxy server. Users route all their requests to servers outside IITK lan through this proxy server. Some user-server interactions get modified due to presence of the proxy server depending upon the kind of user request and also, whether the proxy server has the required data object in its cache or not. To capture real user-server interaction behavior, we need to avoid taking into consideration those interactions which are modified due to presence of the proxy server. We identify such requests and purge them from the access log files given to us. We outline the way to do it in this section.

There are three kinds of requests that clients send to the squid proxy server which are *normal tcp fetch requests*, *IMS (if-modified-since) fetch requests* and *requests involving pragma issued by client*. The motivation behind different kinds of request can be many such as the need to access data quickly or to access the latest version of data or to optimize the network resources. Corresponding to each such request there will be an entry made in the access log file, *access.log*, of the proxy server. An entry has many fields, one among which is the result code. The first of its two parts, i.e. the cache result of the request contains information on the kind of request, how it was satisfied, or in what way it failed. Please refer to section Squid result codes for valid symbolic result codes [3]. A study of these result codes and circumstances in which these are produced, provide us with a handle to purge many of the request enteries in *access.log* because these requests are served from the server's cache. We outline below various steps taken by a proxy server in response of each client request and give the description in an algorithmic form. We enclose comments between /* and */ . We have further marked whether an entery containing a particular squid result code will be useful for us or not. In

the access.log files that we obtained from IITK server, we retained only those enteries which have been marked useful in the algorithm below.

- **normal tcp fetch requests**

if desired object is present in cache **then**

if object present in the memory **then**
it's a TCP_MEM_HIT; /* not useful */
else
TCP_HIT; /* not useful */
end if
else
TCP_MISS; /* useful */
end if

- **IMS (if-modified-since) fetch requests**
Here, the client(browser on client's machine) has a locally cached copy of the object that it needs and this object contains its last modification time, LMT_client alongwith it. So, the client(browser) sends an IMS request that contains the time of last modification of the desired object. now, the proxy server checks for presence of the desired object on its cache. If present, then it checks last modification time, say LMT_squid, for the object in its own cache and the response is per the algorithm below:

if desired object is present in cache **then**

if locally cached object is NOT stale **then**

if LMT_client == LMT_squid **then**
TCP_IMS_HIT; /* not useful */
else
TCP_IMS_MISS; /* not useful */
/*since the proxy server always contains the latest of all copies that any of the clients can have, therefore, this ought to be the most recent copy.*/
send the locally cached copy on squid server to the client;
end if
else
send an IMS request to the origin server with LTM_squid embedded into it;

if request succeeds **then**

if LMT_origin_server == LMT_squid **then**
TCP_REFRESH_HIT; /* not useful as server DOES NOT need to send the newer copy of the object */
else
TCP_REFRESH_MISS; /* useful */
server sends the newest copy of the object to the squid server;
this new copy is delivered to the client;
end if
else
the IMS request to the origin server failed
TCP_REF_FAIL_HIT; /* not useful */
deliver the stale copy of the object present on the squid server to the client;
end if
end if
else
TCP_MISS;/* useful */
end if

- **the client issues some pragma such as "no cache" pragma, due to which the squid proxy server HAD to refetch the object.**
TCP_CLIENT_REFRESH_MISS; /* useful */

From above we can conclude that we need to retain only the following enteries in the access log files obtained from the proxy server

1. TCP_MISS
2. TCP_REFRESH_MISS
3. TCP_CLIENT_REFRESH_MISS

References

[1] H. Choi , V. G. Kulkarni , K. S. Trivedi, "Markov regenerative stochastic Petri nets," Proceedings of the 16th IFIP Working Group 7.3 international symposium on Computer performance modeling measurement and evaluation, p.337-357, May 1994, Rome, Italy

[2] S. Dharmaraja, K. S. Trivedi, D. Logothetis, "Performance modeling of wireless networks with generally distributed handoff interarrival times," Computer Communications 26(15): 1747-1755 (2003).

[3] *http://squid.bilkent.edu.tr/Doc/FAQ/FAQ-6.html* Section 6.7 Squid Result Codes.

[4] K. S. Trivedi, "Probability and Statistics with Reliability, Queueing and Computer Science Applications," John Wiley, second edition, 2002.

[5] Wei Xie, PhD thesis, "Availability and Performance Evaluation of E-Business Systems," Chapter 1,2 and 3 (through personal communications).

[6] J. Raymakers et al, "Impact of Failure and Recovery Times of Web Servers on User Perceived Availability",(through personal communications).

[7] A. Fox and D. Patterson, ”When does fast Recovery trump high reliability”, In Proceedings of the Second workshop on Evaluating and Architecting System Dependability, San Jose, October 2002.

[8] Howard F. Lipson & David A. Fisher, ”Survivability:A New Technical and Business Perspective on Security”.

[9] Dongyen Chen, PhD Thesis, ”Analysis and Mitigation of Failures in Communication Systems with Discrete and Fluid Models,” Chapter 1: Topics on Failures and Survivability and Chap 3:Network Survivability Performance Evaluation: A Composite Measure of Performance and Availability. (through personal communications).

[10] R. J. Ellision, D. A. Fisher, R. C. Linger, H. F. Lipson, T. Longstaff, N. R. Mead, ”Survivable Network Systems: An Emerging Discipline”.

[11] B. Randell, ”Facing Up to Faults,” Alan Turing Memorial Lecture, Jan 2001.

[12] J. Payne ,”Downtime Dangers: Stock Market Outages Highlight Software Availability Issues,” *The Payne Report*, Vol. 2, No. 6, July/August 2001. http://www.cigital.com/paynereport/archive/jul-aug2001.php.

[13] Merideth M. G. and Narsimhan P., ”Metrics for Evaluation of Proactive and Reactive Survivability,”

[14] Ellison R. J., Linger R. C., Longstaff T. and Mead N. R., ”Survivable Network Syste/pm Analysis: A Case Study,” IEEE Software, 16(4):70-77, July/August 1999.

[15] D. Long, R. Golding, A. Muir, ”Longitudnal Survey of Internet Host Reliability”, HPL-CCD-95-4, 22 February, 1995.

[16] RFC 2068: HTTP1.1 www.faqs.org/rfcs/rfc2063.html

[17] Network Time Protocol www.ntp.org

[18] www.kloth.net: Softwares related to Simple Network Time Protocol and traceroute.

[19] www.squid-cache.org/ FAQ, Configuration.

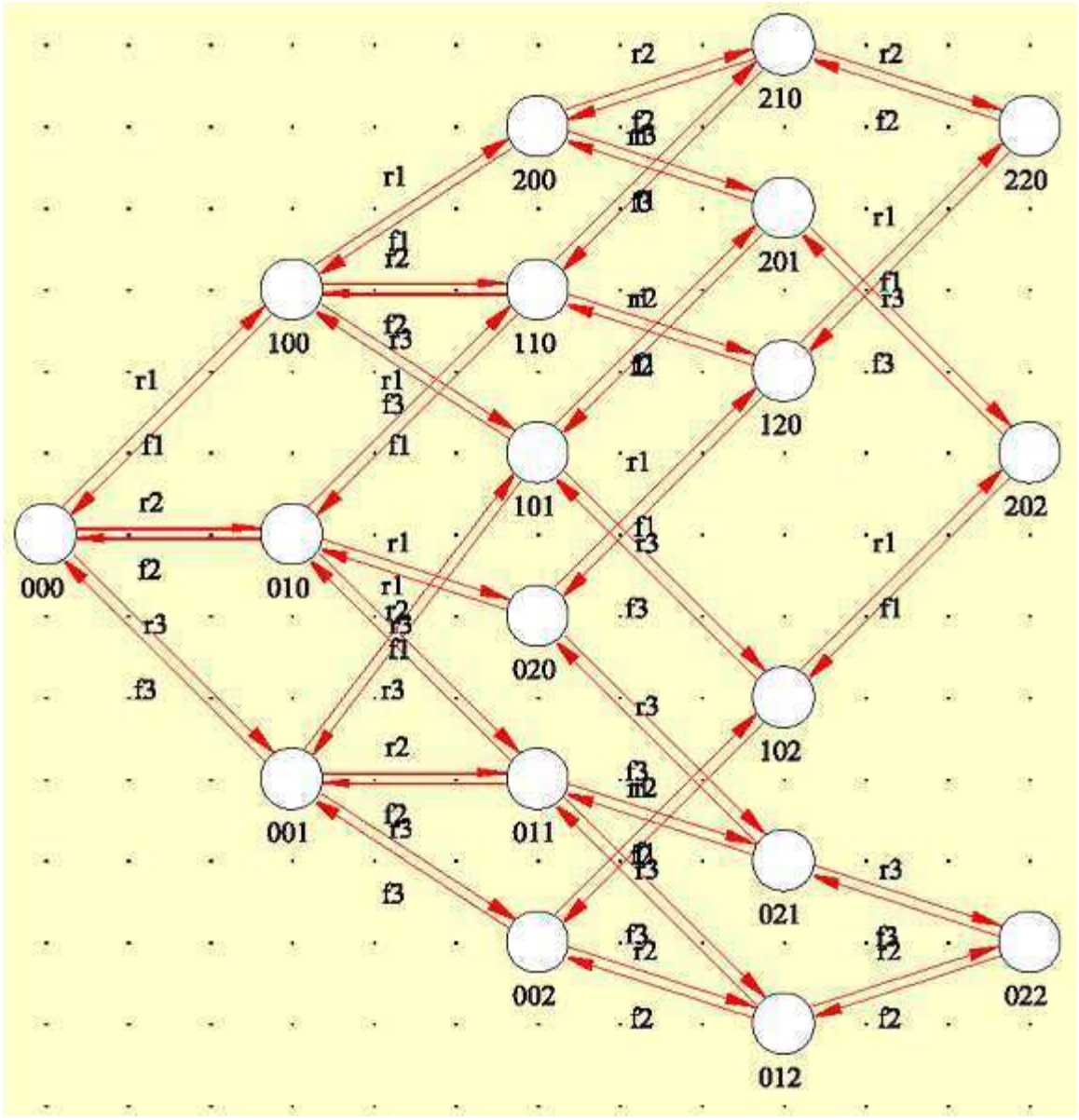


Figure 14: DPF model: Subordinated CTMC formed by concurrent transitions in Active Down States.