Asian Journal of Information Technology 6 (3): 379-384, 2007
© Medwell Journals, 2007

Strategic Modeling and Analysis of Internet Connectivity Using Markov Process to Enhance Internet Performance

[1]O.O. Adeosun, [2]E.R. Adagunodo, [3]I.A. Adetunde and [4]T.H. Adeosun
[1]Department of Ladoke, Computer Science and Engineering, Akintola University of Technology,Ogbomoso, Nigeria
[2]Department of Ladoke, Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria
[3]Department of Applied Mathematics and Computer Science, University for Development Studies, Navrongo, Ghana
[4]Department of Osun State, Banking and Finance College of Technology, Esa-Oke, Nigeria

Abstract: Internet system is developed to satisfy a set of requirements that meet a need. A requirement that is important in Internet system is that it should be highly dependable. Fault tolerance is a means of achieving that dependability. This is our main focus in this study. This research presents a framework for optimizing Internet services using fault-tolerant approach. For effective monitoring, we describe an algorithm for accessing files over replicated proxy servers. The realization and actualization of this modeling is through the use of Markovian processes in which time is measured discretely, or counted.

Key words: Strategic, internet connectivity, actualizaton, algorithm

INTRODUCTION

The whole world is increasingly dependent on services provided by computer systems vis-a-viz Internet system and our vulnerability to computer failures is growing fast as a result. We are of the opinion that a system is said to have a failure if the service it delivers to the user deviates from compliance with the system specification for a specific period of time. The terms failure and fault are key to any understanding of system reliability and they both serve as treats to system reliability. Fault can be defined as the adjudged cause of a failure [3GPP, 2002]. Every fault is a failure from some point of view. The observable effect of a fault at the system boundary is called a symptom. The most extreme symptom of a fault is a failure. Pease *et al*. (1980) said it is possible for a fault to lead to other faults, or to a failure or neither. A system with faults may continue to provide its services (i.e. not fail). Such a system is said to be fault tolerant. This is our focus in this research, to provide network services to clients without any interruption even at presence of faults, thereby making the Internet system transparent to the clients.

To implement dependable and highly-available systems, a fault-tolerant system requires an architecture that includes necessary hardware and software redundancies (Arlat *et al*., 1989). Our approach in this research is simply to implement a service in Internet connectivity by replicating the proxy server and use appropriate algorithms to coordinate the replicas. The algorithm provides the abstraction of a single service to the clients but the replicated proxy server continues to provide correct service even when a fraction of the replicas fail. Therefore, the system is highly available and stable provided the replicas are not likely to fail all at the same time. This scenario would definitely enhance Internet System performance.

Aims of the study: The whole world is daily increasingly dependent on services provided by Internet system and we would like this system to be highly available and dependable at all times even at the occurrence of faults. This research considers the problem of providing high availability, reliability and performance of Internet system through development and deployment of fault-tolerant platforms.

Obviously, in a large system like Internet system under consideration, to achieve high availability cum performance, the reliability probability of the system must be very close to 100% thereby making the expectation of the system as a whole remaining fault-free. The present Internet system is far from realization of this fact.

Accessing large documents from geographically distributed servers can be difficult due to unreliable-network connections or faulty servers. Servers may

Corresponding Author: O.O. Adeosun, Department of Ladoke, Computer Science and Engineering, Akintola University of Technology,Ogbomoso, Nigeria

occasionally become overloaded, unresponsive, or they may crash. The network itself is subjected to transient congestion and individual links may break down. Some of the existing proposed solutions (Byers *et al.*, 1999; 1998) use error-correcting codes to allow clients to reconstruct missing or damaged data. These solutions require that both the servers and the clients perform computations in order to encode the data to be transferred and decode the received information. If the clients or servers have limited computational capabilities, this can be difficult to implement.

Kuhn (1997) opined that studies over the past decade have consistently shown that the reliability of Internet system falls far short of the five 9s (99.999%) of availability expected in the public-switched telephone network. Therefore, it is highly necessary to critically examine and thoroughly access the Internet and its performance with a mind to improve access to Internet and its services. Abiona (2005) opined that traditional approach to improving network performance e.g., upgrading severs and increasing bandwidth can only work for short time. In contrast to this traditional approach, this research adopt the fault tolerant approach which recognizes that the proxy server been the most active and busiest server in Internet connectivity, is a valuable resources that needs to be managed, conserved and shared as effectively as possible. Our research aimed at replicating this server in an attempt to provide spares in case there is a fault or failure, so that the system can switch to a spare of it thereby keep the Internet services going without any interruption. This approach will definitely provide a last long solution to this problem of Internet service performance. Hence, the introduction by this research, the fault tolerant technology approach to enhance the Internet services performance using Markovian Processes to model the Internet connectivity.

Performance: Performance can be measured in many ways, including transit time and response time. Transit time is the amount of time required for a message to travel from on device to another. Response time is the elapsed time between an inquiry and it response.

The performance of a network depends on a number of factors, including the number of users, the type of transmission medium, the capabilities of the connected hardware and the efficiency of the software (Rubino and Sericola, 1993).

The dominant protocol used on the Internet today is TCP, a reliable window-based protocol. Under ideal conditions, best possible network performance is achieved when the network pipe between the sender and the receiver is kept full of data.

Reliability: The domain of reliability engineering involves considerations of all aspects of design, development and fabrication, so as to minimize the chance of equipment breakdown. Neglect of reliability considerations can prove to be very costly, from the loss of vital information to the possibility of losing confidence in the system.

As Internet system become more complex the chances of system unreliability becomes greater, since the reliability of a piece of equipment in Internet system depends on the reliability of its components. The relationship between parts reliability and the Internet system reliability can be formulated mathematically to varying degrees of precision depending on the scale of the modeling effort. The mathematics of reliability is based on parts failure rate statistics and probability-theoretic relationships. The following list comprises a number of definitions in decreasing order of stringency.

- Freedom from faults.
- Tolerance of faults with no loss of performance.
- Partial fault-tolerance or graceful degradation.

Error-containment or fail-safe: Faults may occur but their consequences are localized and there is no runaway of the system.

For any structure, reliability is a function of the number and organization of the interconnections of various components to form a viable system. It is possible to reduce reliability as well as to increase it. There have been two distinct and viable approaches taken to enhance system reliability. One is based on component technology, i.e., manufacturing the component as intrinsically reliable as possible followed by parts screening, quality control, pre-testing to remove early failures (infant mortality effects) etc. The second approach is based on the organization of the system itself, e.g. fault-tolerant architectures where the architecture makes use of protective redundancy to mask or remove the effects of failure and thereby provides greater overall system reliability than would be possible by the use of the same component in a simplex or non-redundant configuration.

Fault-tolerance is provided by the application of protective redundancy: use of more resources to upgrade Internet system reliability. These resources may consist of more hardware, more software, or more time, or a combination of all of these. Extra time is required to re-transmit messages or to re-execute programs, extra software is required to perform diagnosis on the hardware, extra hardware is required to provide replication of units.

Mathematical theory of reliability: Some relationships between reliability parameters and the underlying probability-theoretic relationships are as follows. If a fixed large number N_0 of identical items is being tested, of which N_s is the number of items surviving after time it and N_f the number of items that failed during time it, then, for all t, $N_0 = N_s + N_f$. Now, for a sufficiently large iN_0, the reliability R(t) of an item is N_s / N_0. The failure rate $\lambda(t)$, which is defined to be the rate at which the population changes at time t, can be shown to be given by:
1dR(t)

$$\lambda(t) = -\frac{1}{R(t)}\frac{dR(t)}{dt} \quad (1)$$

$$R(t) = e^{-\lambda(t)\,dt} \quad (2)$$

The reliability function R(t) is often called the survival probability function since it measures the probability that failure of an item does not occur during the time interval [0,t].

FAULT-TOLERANT INTERNET CONNECTIVITY MODELING

The Markov process is used to model a novel fault-tolerant Internet Connectivity as shown in Fig. 1. It is a conceptual view of fault-tolerant Internet Connectivity that would actualize our intension of building an Internet system that would tolerate faults and failures, hence improving on Internet system reliability.

From Fig. 1, $S_0 \ldots S_2$ are replicated proxy servers. This type of arrangement illustrates protective redundancy and to be precise load sharing technique. The system is composed of functionally parallel proxy servers in such a way that if one of the proxy servers fails, the parallel proxy server will continue to do the system function.

Markovian methodology for modeling fault-tolerant internet system: The basic methodology presented in this work was developed from the work of Russian mathematician, A. A. Markov, around the beginning of the 20th century. Markov processes form a subclass of the set of all random processes-a subclass with enough simplifying assumptions to make them easy to handle. Markov processes deal with time measured discretely, or counted. The only mathematics used is algebra. Many of the concepts established in the discrete time case hold for continuous time Markov processes, which are the subject of this work.

Considering Fig. 1, where we have three replicated proxy servers arranged in parallel, one of them will be declared active at any point in time while the remaining two will be in look-up state with intention to assign any one of them to replace the service proxy server if faulty. From this type of arrangement, the proxy server to be selected if the service one is down depends to some extent on the immediate past proxy server used.

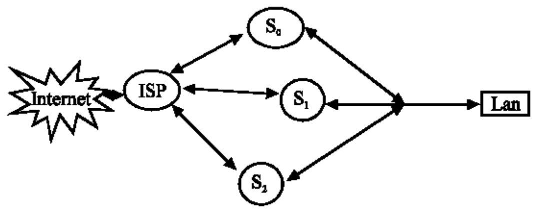

Fig. 1: Conceptual model of a fault-tolerant internet system

Many questions might crop up from this process. The following gives just a sample:

- If in Fig. 1, S_1 is the first proxy server to be selected, when is it going to drop S_1 for S_2?
- If the faulty S_1 is repaired or replaced with new one, when will it return to S_1?
- If there is no problem with S_1 selected throughout, what is going to be the importance of adding replicated proxy servers?

These are examples of some very vital practical questions which we might need answers to in order to evaluate potential satisfaction with a proxy server selected amidst the replicated ones. Some of these questions require probabilities as answers; others require expected values. Some pertain to the probability of selecting a particular server at a specified time; others pertain to the time required to select a specific server.

Because reassignment occurs only when there is failure of service proxy server, the passage of time may be recorded by counting the number of reassignments that has occurred. Starting at an arbitrary time, denote a particular 6-failure interval by the number zero. The first reassignment will occur at the end of that interval, so the following interval will be denoted by the number one and so on. The nth reassignment occurs at the beginning of the nth 6-failure interval.

It is convenient to number the Proxy Servers as shown in Fig. 2. The first Proxy server is numbered 1, the second one 2 and the third 3. This particular arrangement is obviously arbitrary.

A realization, or time history, of the process can now be drawn as a graph, as in Fig. 2. This shows that at failure 0 proxy server iS_0 is assigned, was then assigned to proxy server S_1, next to proxy server S_2 and so on. This is a record of the past; the future is unknown. If failure 0 is the present and we wish to predict the future, we would

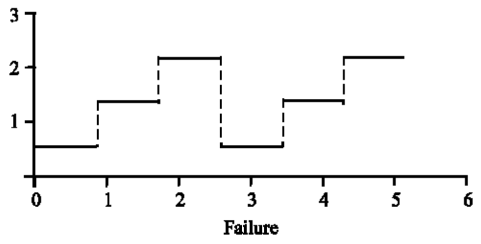

Fig. 2: A realization of assignment and reassignment of proxy servers

have to say that it is represented by the set of all possible such graphs, each graph having a certain probability associated with it.

If we specify a particular proxy server failure in the future and ask what proxy server will be assigned at that time, we would have to say that it is given by a random variable which can take on the values 1, 2 and 3, according to some probability distribution. One way to describe the future of the whole process is to say that it is given by a sequence of random variables, $\{X_1, X_2, X_3 ...\}$. The random variable X_n would be interpreted as the (uncertain) assignment that the proxy server will have during the nth failure.

A discrete time stochastic process is any sequence of random variables indexed by time. The set of all possible values for the random variables is called the state space: the value assumed by X_n is called the state of the process at time n. Changes of state are called transitions. The State Space is a pictorial map of the process in which states are represented by points and transitions by arrows. Proxy Servers are the states in this case. An arrow from S_0 to S_1, for example, would mean that if the process is in iS0 at some time in (no failure), then it is possible for it to be in iS_1 at time in + 1 (failure). In this research, there are three states with no restrictions on possible transitions, so the transition diagram would be as shown in Fig. 3.

It is visualize as the random walk of a particle over the transition diagram. Occupying a state is equivalent to the particle being at a particular point; a transition corresponds to a jump or an instantaneous movement of the particle along one of the arrows. Virtual Transition occurs when the new state is the same as the old and is represented by loops in Fig. 3.

The process is described by a sequence of random variables, $\{X_1, X_2, X_3, ...\}$ and the random variables are not independent. A complete specification of a stochastic process would require joint distribution functions, taken jointly over all of the random variables.

Markov's simplifying assumption is made that the proxy server to assign next depends only on the present

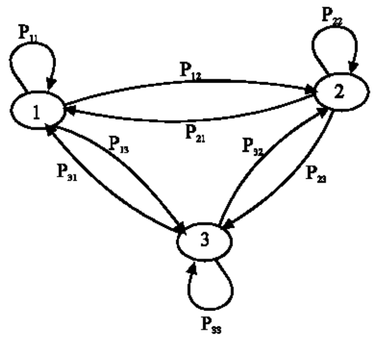

Fig. 3: Transition between the replicated proxy servers

proxy server assignment. We thereby eliminate all previous experience of the proxy server assignment as a factor in determining the next proxy server assignment. In general term, a Markov chain is a discrete time stochastic process in which each random variable, X_i, depends only upon the previous one, X_{i-1} and affects only the subsequent one, X_{i+1}. The term chain suggests the linking of the random variables to their immediately adjacent neighbours in the sequence.

Suppose that X_0 represents the proxy server's present assignment and we are interested in X_1, the next proxy server to be assigned. What we want is a probability distribution over the three possible values for X_1. But these probabilities depend on the proxy server on use at a particular point in time. Suppose S_1 is the assigned proxy server (i.e., $X_0 = 1$). Then, for its next assignment if there is failure, S_2 will be assigned (i.e., $X_1 = 2$) or S_3 in case S_2 is equally faulty (i.e., $X_1 = 3$). This is shown in Fig. 3.

Where there are faults that individual proxy servers can tolerate, we have this scenario: Suppose S_1 is the assigned proxy server (i.e., $X_0 = 1$). Then, for its next assignment if there is fault that can be tolerated, S_1 will be assigned (i.e., $X_1 = 1$), ditto for others as shown in Fig. 3.

The convenient way to record these different probabilities is in matrix form:

$$P = \begin{matrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{matrix}$$

The first row of this matrix represents the probabilities that the proxy server assignment will be in this order S_0, S_1, S_2, respectively, if the current assigned is S_0. The second row gives the same probabilities if S_1 is first assigned; the third row, if S_2 is first assigned. Each row is, individually, a probability distribution for X_1; the appropriate one to use depends on which value X_0 has.

Since each row is a probability distribution (as stated above), it implies that the elements of each row must sum to 1.

The element p_{ij} means the probability that $X_1 = j$ if we know that $X_0 = i$. This is the conditional probability, $p_{ij} = P\{X_1 = j \mid X_0 = 1\}$. This is a probability of going from state i to j. The ip_{ij} 's are called (one-step) transition probabilities and the matrix P is called the transition matrix.

We labeled the arrows of transition diagram in Fig. 3 instead of using the matrix. The arrow from i to j would receive the label p_{iij}, which can then be thought of as the probability that that arrow is used when the particle leaves point i. Row i of P would correspond to the set of arrows leaving point i, so the sum of the probabilities taken over this set must equal 1. If any of these p_{ij} are zero (indicating that a direct transition from i to ij is not possible) the arrow from i to ij would simply not appear in the diagram.

Algorithm for document requests on the design system: A Web document W is a sequence of |W| bytes, $(b_0, b_1, b_2, \ldots b_{|iW|-1})$. The document is assumed to be located at N different iPSs (Proxy Servers), $S_0, S_1, \ldots, S_{N-1}$. A chunk of W starting from position i and ending at position j, $0 \le i \le j = |iW| - 1$, is denoted as iW[i,ij] and is defined as

$$W[i,j] = (b_i, b_{i+1}, b_{i+2}, \ldots b_{j-1}, b_j). \qquad (3)$$

A request R made to a PS consists of any number of non-overlapping chunks of W. The size of a request iR is the total number of bytes it contains. We will use the term request to indicate both those generated by the client, containing basically the list of starting and ending positions for the chunks and the reply generated by a proxy server. The size of a request is thus the number of bytes that are contained in the portion of the document sent from one server to the client.

In Fig. 4 we consider the problem of accessing large documents by hosting them on multiple proxy servers and accessing the proxy servers in parallel. Suppose we want to access a document W on N different proxy servers $S_0, S_1, \ldots, S_{N-1}$, where each server has a copy of iW. This is shown in Fig. 4 above. Choosing the best proxy server for fetching the page should be nontrivial problem. Not all the replicas could be up and running at the same time and not all the replicas could provide the same bandwidth and/or the same latency. Moreover, the variable nature of the network may alter the parameters of each client-server connection, slowing down previously fast connections or speeding up congested ones.

```
Require: K, $1 \le K \le N$
Ensure: is the request for proxy server $S_i$, $0 \le i \le N-1$
  fragSize := $|W| / N$
  t := 0
  $R_0 := R_1 := \ldots := R_{N-1} := \varnothing$
  for i = 0 to N-1 do
      Wi := W[i x frag size, (i+1)x frag size-1]
      for j = 1to N-K + 1 do
      $R_t := R_t \cup W_i$
      t (t+1) mod N
      end for
  end for
```

Fig. 4: Algorithm for computation of $R_0, R_1, \ldots, R_{N-1}$

approach for accessing documents over multiple Proxy Server (PS) replicas. Our proposal is based on the iinformation dispersal technique described in Rabin (1989) and works as follows. Different subsets $R_0, R_1, \ldots, R_{N-1}$ of the original page iW are requested to the iN PS replicas (our iN in this study is taken to be 3) in such a way that any iK replies (where iK is a user-defined parameter, 1= iK = iN) are sufficient to reconstruct the page. The size of each request will be shown to be

$|W| / N\ (N - K + 1)$. Setting K = 1 means requesting the whole page to all replicas, while setting K = N is equivalent to distributing the requests among all N replicas in such a way that all of them must reply before it is possible to reconstruct the page. Tuning the parameter K allows the client to automatically select the K fastest replicas, at the price of requesting bigger subsets of the document iW as iK decreases. The performance of the proposed algorithm shown in Fig. 4 above, can be evaluated by computing the impact of the parameter K on the probability of completing a transfer in time less than t.

We define a set of N requests, $R_0, R_1, \ldots, R_{N-1}$ for a given document iW such that iR_i will be sent to server S_i, with the following properties:

- Any iK replies are sufficient to reconstruct the whole document, for a fixed K, $1 \le K \le N$
- All the requests have approximately the same size.

The requests R_i, i = 0, ..., N – 1 can be computed using algorithm in Fig. 4. The algorithm divides the page iW in iN non overlapping chunks $W_0, W_1, \ldots, W_{N-1}$, of size |iW|/ iN each. Each chunk is then cyclically inserted into N – K+1 different requests. It can be easily seen that this guarantees property (1) above: Any chunk W_j

will not appear in N – (N - K+1) = K – 1 requests, so any iK of them can be chosen with the guarantee that at least one will contain a copy of every chunk. To guarantee property (2), the chunks are evenly distributed among all the requests.

Since each chunk of size |W|/ N is present in N – K+1 different requests and there are iN chunks, the total size of all the requests is |W| (N – K + 1), the size of each request iR_i being $|R_i|$ = |W|(N – K + 1)/N.

CONCLUSION

In this research work we presented an algorithm for fault-tolerant retrieval of a Web document iW replicated among iN different PS replicas iS_0, iS_1, iS_{21}. The algorithm is evaluated analytically by using Markov model of network connection. The analysis showed that the proposed algorithm can be effective in reducing lost or incomplete transfer of a document iW , which is quite interesting as the algorithm does not require any feedback from proxy servers or from the client, nor it does monitor the network performances. The value of the parameter K, the number of replies which are sufficient to reconstruct W , must be carefully chosen in order to obtain good performance. Higher values of K are recommended if most of the network connections offer high throughput. Lower values of iK are recommended if the network connections are highly unbalanced and only few of them offer high throughput. The redundancy in this work would make the multiple proxy servers more resilient to network failures. The result would enhance Internet service dependability however, at the expense of substantial amount of money to procure the replicated proxy servers and their associated equipments.

REFERENCES

Abiona, O.O., 2005. Development of a Framework for Optimizing Access to Wide Area Network Resources, a Ph.D. Thesis submitted to the Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria.

Arlat, J., Y. Crouzet and J.C. Laprie, 1989. Fault Injection for Dependability Validation of Fault-Tolerant Computing Systems, Dig. Int. Symp. Fault Tolerant Computing, FTCS-19, pp: 348-355.

Byers, J.W., M. Luby, M. Mitzenmacher and A. Rege, 1998. A digital fountain apporoach to reliable distribution of bulk data, In: Proceeding SIGCOMM'98, pp: 56-67.

Byers, J.W., M. Luby and M. Mitzenmacher, 1999. Accessing multiple mirror sites in parallel: Using tornado codes to speed up downloads. In proceeding INFOCOM'99, pp: 275-283.

GPP, 2002. Network Architecture (Release 5), Technical Specification 3rd Generation Partnership Project, Technical Specification.

Kuhn, C., 1997. Improving round-trip time estimates in reliable transport protocols. Theoretical Computer Science, 4: 364-373.

Pease, M., R. Shostak and L. Lamport, 1980. Reaching Agreement in the Presence of Faults. J. ACM., 27: 228-234.

Rabin, M.O., 1989. Efficient dispersal of information for security, load balancing and fault tolerance, J. ACM., 36: 335-348.

Rubino, G. and B. Sericola, 1993. Interval availability distribution computation, IEEE. Trans. Computers, pp: 48-55.