INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS AND ROBOTICS
ISSN 2320-7345

IMPROVING SERVER EFFICIENT AND QUALITY OF SERVICES BY INTELLIGENT SECURE SERVER MODEL

[1]*Narasimha Murthy S,* [2] *Dr. T.R. Mahesh*

[1] Scholar of Department of Computer Science Engineering, Singhania University Rajasthan, India
[2] Prof and Head of Dept Computer science T. John Institute of Technology. Bengaluru
{Mail: narasimhas.murthy@axa-tss.com, dr.maheshtr@gmail.com}

Abstract

This is the crucial time for the intermediate node named server, that plays an important role to manage and provides performance efficiency during the communication in the network, as we all know that it the biggest issue of discussion that how to manage the millions of request with high performance capacity networking at server level, network user and application are increasing, as result storage need and server performance degrades.

In proposed research paper author addressing the solution at server end by implementing a scheme that improves server working efficiency even in the case when server having high user and application data with millions of user interface. In this case traditional server scheme are not capable much enough to get improve server efficiency as required therefore the low performing server getting suffered with the well known problem as connection failure, data loss, congestion error, delayed response and low throughput, to get eliminate all the discussed problem at some extent level, author proposed a model named ISS, "Server efficiency model."

Keyword: ISS, data loss, throughput, congestion.

1. Introduction

According to the existing technique organization having a server where all the data is stored and managed by higher authority , any client if he want to access server they first need to take the permission from higher administrator and than he can directly access the server [10] . In this case the centralized server responsible for performing all the operations for user authentication and management of information. If any user who can have ability to break the authentication then he can directly access the server because server is secured by ID and Password which is not very effective because ID and Password can be shared by someone, can be stolen by someone or can be guess by someone so that confidentiality is not achieved at higher level . According to the past analysis one studied that many research scholars already have done research on Server Performance Issues , Many of the proposed protocols and technique to manage congestion , some of them are related to the detection of error due to the traffic it is obvious that the user of internet is increasing all around the world it does traffic so that one should

have some effective technique to provide secure service , Detection of error and flow control is not sufficient now one need to have some kind of approach that will not only cover the security aspects but also managing the information flow with accuracy and reliability [5,6] . In our proposed research we cover following:

1.1 For Authentication

One proposes biometric image identification code as a fingerprint which is performed by some other external server named intelligent secure server. In order to maintain higher level of security server performed authentication with secure code dynamically [12]. Now the entire client has to face ISS first and then it will have the direct access permission to the centralized organization server. ISS works on TCP connection at run time so that failure of connection will be reduces at run time, TCP is also support multi stream delivery services so that biometric image can be supported by the connection, One of the more advantage of TCP is that it provide buffering function so that now server has more strong buffering functionality at run time if server facing many users at same time of period.

1.2 For Managing Flow and Error Control

ISS also provides flow and error control at run time by introducing itself. Error and flow control problem is associated with traffic, when the bandwidth of channel is low than the bandwidth of data packet available at time T1 period than network has to face flow and error problem but ISS solve it at same time T1 period by introducing ISS storage here ISS working like a intelligent machine to have temporary storage to solve congestion flow and error problem [8,9].

2. ISS Quality Concept

Fundamentally, QoS enables you to provide better service to certain flows. This is done by either raising the priority of a flow or limiting the priority of another flow. When using congestion-management tools, you try to raise the priority of a flow by queuing and servicing queues in different ways [5]. The queue management tool used for congestion avoidance raises priority by dropping lower-priority flows before higher-priority flows. Policing and shaping provide priority to a flow by limiting the throughput of other flows. Link efficiency tools limit large flows to show a preference for small flows [04,03].

Server QoS is a tool box, and many tools can accomplish the same result. A simple analogy comes from the need to tighten a bolt [11]. You can tighten a bolt with pliers or with a wrench. Both are equally effective, but these are different tools. This is the same with QoS tools. You will find that results can be accomplished using different QoS tools. Which one to use depends on the traffic? You wouldn't pick a tool without knowing what you one are trying to do, would you? If the job is to drive a nail, you do not bring a screwdriver [7]. The basic architecture introduces the three fundamental pieces for QoS implementation identification and marking techniques for coordinating QoS from end to end between network elements

- QoS within a single network element (for example, queuing, scheduling, and traffic-shaping tools).
- QoS policy, management, and accounting functions to control and administer end-to-end traffic across a network.

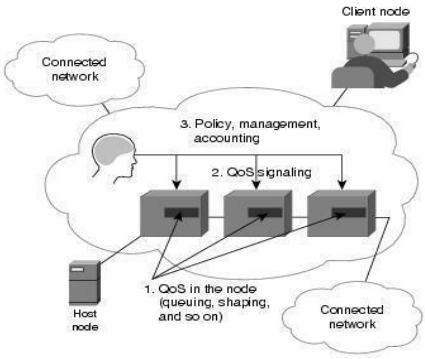


Figure1: Basic implementation of secured network

3. Making Biometric Systems More Efficient

This is a method of enhancing the security and privacy of biometric authentication. Instead of enrolling with a true finger (or other biometric), the fingerprint is intentionally distorted in a repeatable manner and this new print is used. If, for some reason, the old fingerprint is stolen then an essentially a new fingerprint can be issued by simply changing the parameters of the distortion process. This also results in enhanced privacy for the user, since the true fingerprint is never used anywhere and also different distortions can be used for different types of accounts. The same technique can also be used with other biometrics to achieve similar benefits. To implement the proposed one user following concept. Let D and T be the representation of the Database Template and Synthetic Template respectively. Each minutia may be described

by a number of attributes, including its location in the fingerprint image, orientation, type etc. Most common minutiae matching algorithms consider each minutiae as a triplet m={x,y, } that indicates the minutiae location coordinates and the minutiae angle.

- D= $\{m_1,m_2,\ldots\ldots m_n\}$ $m_i = \{x_i,y_i, _i\}$ i= 1….m.
- T= $\{ m'_1,m'_2,\ldots\ldots m'_n \}$ $m_j = \{x'_j,y'_j, '_j\}$ j= 1….n.

Di : The database template corresponding to user i , i =1, 2,3,....N , where N is the total number of users registered in the system. It is assumed that the attacking system knows the format of this template, but it cannot access the template itself.

Tij : The jth synthetic template generated by the attacking system for user i. This template has the same format as database templates; it can be represented as S (Di, Tij): The matching score between Di and Tij. For attacking a specific user account, the attacking system must follow the following five steps: [ULU, 2004] also shown in Figure 1.17.

- **Step 1** (Initial guessing): Generate a fixed number of synthetic templates (Ti^1, *Ti2, Ti3* …… Ti^{100}).
- **Step 2** (Try initial guesses): accumulate the corresponding matching scores [$S(Di, Ti^1)$,
- $S(Di, Ti^2)$, $S(Di, Ti^3)$,..., $S(Di, Ti^{100})$] for user *i.*
- **Step 3** (Pick the best initial guess): Declare the best guess Ti^{best} to be the template resulting in the highest matching score.

- **Step 4** Modify Ti^{best} by adding a new minutia, replacing an existing minutia. If for any one of these attempts, the matching score is larger than previous $S^{best}(Di)$ declare the modified template as Ti^{best} , and update $S^{best}(Di)$ accordingly.

- **Step 5** (Obtaining result): If the current best score is accepted by the matcher (namely, $S^{best}(Di)$ $S_{Threshold}$), stop the attack.

4. Implementation Policy for Different Server Ends

one use simulation results based on the following server characteristics: T_{OFF} = 200s, T_{SLEEP} = 60s, P_{OFF} = 0W, P_{SLEEP} = 10W, P_{IDLE} = 150W and P_{ON} = 240W. These parameter values are based on measurements for the Intel Xeon E5320 server, running the CPU-bound LINPACK [13] workload.

4.1 Optimal Single Server policy

As the first step towards our goal of finding policies for efficiently managing server pools, one analyzes the case of a single server system. Recall that our aim is to find the policy that minimizes ERP under a Poisson arrival process of known intensity. Below Equation states that for a single server, the optimal policy is included in the set (NEVEROFF, INSTANTOFF, SLEEP) and hence there is no need to consider any other capacity provisioning policy.

- Let π mixed denote the class of randomized policies whereby a server immediately transitions to power state S i (i ϵ (0……..N)) with probability pi on becoming idle. Given that, the server onent into power state S_i, with probability q_{ij} it stays in S_i and waits until j jobs accumulate in the queue, where $\sum_{j=1} q_{ij} = 1$. Once the target number of jobs has accumulated, the server immediately begins transitioning to the on state, and stays there until going idle.

$$E[T] = \frac{sum_{\sum pij \sum qij.rij}^{i=0}}{sum_{\sum_{i=0}^{N} Pi \sum qij(j+\lambda Tsi)}^{x}}$$

- Under a Poisson arrival process and general i, d. job sizes, the optimal policy lies in the set $\prod_{mixed}$. Consider a policy $\pi \prod_{mixed.}$

$$rij = \frac{j + \lambda Tsi}{\mu} - \lambda + \left[jTsi + \frac{j(j-1)}{2\lambda} + \frac{\lambda T}{2} \right]$$

- With parameters as in Equation 1. The mean response time for policy π under a Poisson (λ) arrival process with i..d. Exp(∏) job sizes is given by:

$$E[P] = fracsum \sum qij.rij \sum_{i.....0}^{j....0} xij(p-1)(q-1) + \lambda T siPon$$

4.2 Near-Optimal Multi-server policy

In this section, one extends our results for single server systems to the multi-server systems with a fixed known arrival rate, with the goal of minimizing ERP. where one found the best of NEVEROFF, INSTANTOFF and SLEEP to be the optimal policy, one intuit that in the multi-server case, one of NEVEROFF, INSTANTOFF and SLEEP will be close to optimal as one and all., one provide simple guidelines for choosing the right policy from among this set, depending on the system parameters.

- Let $\prod_{OFF}$ denote the class of policies which only involve the states on, idle and off. The ERP of the best of NEVEROFF and INSTANTOFF is within 20% of the ERP of the optimal policy in $\prod_{OFF}$ when p≥10. When p ≥ 20, the performance gap is smaller than 13%.
- Let $\prod_{Si}$ denote the class of policies which only involve the states on, idle and the S_i sleep state. For arbitrary S_i (that is P_{Si} and T_{Si}), the ERP of the best of NEVEROFF and SLEEP with sleep state S_i is within 30% of the ERP of the optimal policy in $\prod_{Si}$ when p≥ 10. When p≥ 20, the performance gap is smaller than 23%.
- The main idea behind Conjectures 1 and 2 is obtaining reasonably good lower bounds on the ERP for the optimal policy, and then numerically optimizing the performance gap with respect to the lower bound. The justification for Conjecture 2 is similar, and one omits it due to lack of space.
- One believe that in reality, the simple NEVEROFF, INSTANTOFF, and SLEEP policies are better than our Conjectures suggest. To justify this claim, one perform the following simulation experiment. One focus on the case in Conjecture 1 of policies involving on, idle and off states. Note that as one mentioned earlier, due to the metric of ERP, one can not utilize the framework of Markov Decision Processes/Stochastic Dynamic Programming to numerically obtain the optimal policy. Instead one limit ourselves to the following class of threshold policies.

5. Comparison of Different Threshold Computation

(n_1, n_2): At least n1 servers are always maintained in on or idle state. If an arrival finds a server idle, it begins service. If the arrival finds all servers on (busy) or turning on, but this number is less than $n_2 \geq n_1$, then the arrival turns on an off server. Otherwise the arrival waits in a queue. If a server becomes idle and the queue is empty, the server turns off if there are at least n_1 other servers which are on.

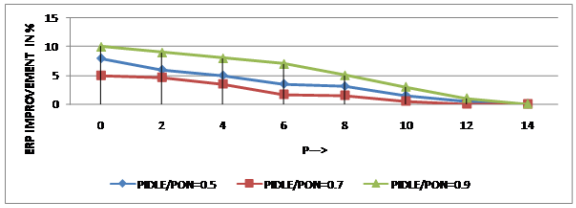
- To perform the optimum decision to precede request from i to j node one need to analyze the result with the help of following equation.

$$E[I] = \sum_{i=..n}^{j=....ni} pij.qij$$

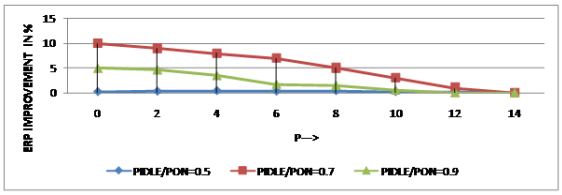
- The MXA parameter to get compute the buffering initial value from i=0 node , one need to organize the measurement value fist to observe the value of remaining nodes that specify that the actual required node buffer size.

$$Bf[T] = \int_{x=n}^{j=n} xij.rij \sum mxa(Nij.Nmin)$$

- In this way node buffer specifies its initial value with the actual parameter network node N take the upper bound and lower bound parameters.



(a) T_{OFF}=0.1E[S]



(a) T_{OFF}=2E[S]

Figure 2: Comparison of the performance of THRESHOLD policy against the best of NEVEROFF and INSTANTOFF policies. The y-axis shows the percentage improvement in ERP afforded by the THRESHOLD policy.

The THRESHOLD policy can be seen as a mixture of NEVEROFF with n1 servers, and INSTANTOFF with (n2 - n1) servers. Thus, THRESHOLD represents a broad class of policies (since n1 and n2 can be set arbitrarily), which includes NEVEROFF and INSTANTOFF. In Figure 2, one shows the gain in ERP afforded by the optimal THRESHOLD policy over the best of NEVEROFF and INSTANTOFF for various values of p, T_{OFF} and P_{IDLE} P_{ON}.

One see that if T_{OFF} is small (Figure 2 (A)), the ERP gain of the THRESHOLD policy over the best of NEVEROFF and INSTANTOFF is marginal (< 7%). This is because in this case, INSTANTOFF is close to optimal. At the other end, when TOFF is large (Figure 1.20 (c)), the ERP gain of the THRESHOLD policy over the best of NEVEROFF and INSTANTOFF are again marginal (< 6%), because now NEVEROFF is close to optimal. One expect the optimal THRESHOLD policy to outperform the best of NEVEROFF and INSTANTOFF when T_{OFF} is moderate (comparable to P_{IDLE}_E[S]/ P_{ON}). In Figure 2 (B), one see that this is indeed the case. However, the gains are still moderate (an improvement of 10% when p≥ 10 and at most 7% when ≥ 20 when PIDLE is high).

6. Improving Algorithm Efficiency

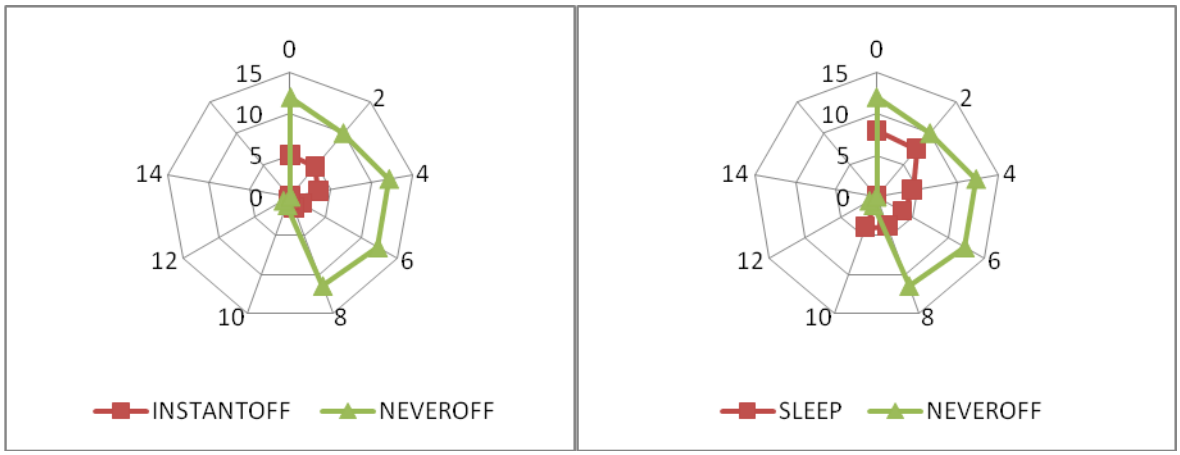
To provision a multi-server system with a fixed known arrival rate, it success to only consider the policies NEVEROFF, INSTANTOFF and SLEEP. The goal of this section is to develop a series of simple rules of thumb that help a practitioner choose between these policies. The specific questions in this section are. One justifies the heuristic rule of thumb by proposing approximations for the ERP metric under INSTANTOFF, NEVEROFF, and the SLEEP policies. One expect the INSTANTOFF policy to outperform NEVEROFF and SLEEP when TOFF is small enough compared to E[S], so that the penalty to turn on an off server is negligible compared to the necessary cost of serving the job. In this regime, one can approximate the ERP of INSTANTOFF by

$$\mathbf{ERP^{INSTANTOFF} \approx \lambda P_{ON} (E[S] + T_{OFF})^2}$$

Which is an upper bound obtained by forcing every job to run on the server that it chooses to turn on arrival. The ERP of NEVEROFF with optimal number of servers is approximated by Eq. (11), with P_n = P and ε = ε *(P_{IDLE}/P_{ON}). For SLEEP, one again expect SLEEP (S_i) policy to outperform NEVEROFF when T_{Si} is small enough so that almost all jobs turn on a sleeping server and get served there. In this regime, one can approximate the ERP of SLEEP, with ε = ε * (P_{Si}=P_{ON}). Using the above approximations for ERP, one can choose between the INSTANTOFF, NEVEROFF and SLEEP policies.

If one compare INSTANTOFF and NEVEROFF, Rule of Thumb, says that if T_{OFF} is sufficiently

Small compared to E[S] then one should choose INSTANTOFF. Figure 3(A) verifies the accuracy of the above rule of thumb. Observe that in the region where our rule of thumb mis predicts the better policy, the gains of choosing either policy over the other are minimal. Similarly, the dashed line in Figure 3(B) indicates that the theoretically predicted split between the NEVEROFF and SLEEP policies is in excellent agreement with simulations.



(A) (B)

Figure 3: Verification of accuracy

This figure shows the relative performance of NEVEROFF, INSTANTOFF and SLEEP policies for a multi-server system, as a function of load, p and mean job size, E[S], based on simulations. Figure 3(A) shows NEVEROFF vs. INSTANTOFF. The crosses indicate the region of superiority of INSTANTOFF over NEVEROFF. Figure 3(B) shows NEVEROFF vs. SLEEP. The crosses indicate the region of superiority of SLEEP over NEVEROFF. The numbers associated with each point denote the % improvement of the superior algorithm over the inferior. The dashed lines indicate the theoretically predicted split based on Rule of Thumb #3.

7. CONCLUSION

The goal of Research work was to characterize the intelligent way of distributing the traffic load in such a way that one can manage the tradeoff between the rate of control and network congestion for flow control policies. So that one can achieve best server level service to make network performance high and to reduces data loss problem which is having due to the congestion of high traffic volume. Specifically, one deal with the question of how often congestion notifications need to be sent in order to effectively control congestion. To our knowledge, this was not the first attempt to analytically study this particular tradeoff. Since this tradeoff is difficult to analyze in general networks, one considers a simple model of a single server queue with optimal server scheme.

The different techniques are evaluated primarily with respect to compatibility to one standard, geographical scalability, and to what extent they achieve load balancing. One did not consider other Internet entities that may dispatch client requests, such as intelligent routers or intermediate name servers, because they are affected by the same problem that limits the portability of client-based approaches.

8. REFERENCES

[1] A congestion-aware access control mechanism in wireless networks. Broadband Network and Multimedia Technology (IC-BNMT), 2010 3rd IEEE International Conference.
[2] A Fast Congestion-Aware Flow Control Mechanism for ID-Based Networks-on-Chip with Best-Effort Communication. Digital System Design (DSD), 2011 14th Euromicro IEEE Conference.
[3] An improved TCP congestion control mechanism based on double-windows for wireless network. Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium IEEE Conference.

[4] Adaptive congestion control mechanism of TCP flows for performance optimization in mobile heterogeneous wireless networks. Computer, Control and Communication, 2009. IC4 2009. 2nd International IEEE Conference.
[5] MAC Channel Congestion Control Mechanism in IEEE 802.11p/WAVE Vehicle Networks. Vehicular Technology Conference (VTC Fall), 2011 IEEE Conference.
[6] Improving Secure Server Performance by EAMRSA SSL Handshakes
Industrial Control and Electronics Engineering (ICICEE), 2012 International IEEE Conf.
[7] Modeling TCP/IP stack in a single custom processor, with secure data transmission to an Altera-based Web server Southeast on, 2011 Proceedings of IEEE.
[8] Mobile Agent for Fast and Secure Services in Client-Server Environment.
Advanced Computing and Communication Technologies (ACCT), 2013 Third International IEEE Conference.
[9] Platform and experimentation of secure service location with P2P/Client-Server over ad hoc networks International journal published in IEEE 2009.
[10] A Load Balancing Scheme for Cluster-based Secure Network Servers International journal IEEE 2005.
[11] Genetic Algorithm Based Secure Authentication Protocol with Dual Central Server and Token Authentication in Large Scale Mobile Ad-Hoc Networks, IEEE publication, 2010.
[12] Enhancing the efficiency of secure network monitoring through mobile agents IEEE journal publication, 2010.

Authors Biography

Narasimha Murthy S Received the B.E. degree in Computer science and Engineer from Gulbarga University, Karnataka, India, in 2000 and the MBA Degree in marketing from IGNOU, Delhi, India, in 2007. He is currently pursuing Ph.D. Degree in Computer Science &Engineering, Singhania University Rajasthan, India his current research interests lie in the areas of Wireless communications and Networks and algorithms, mobile computing, and multimedia Communication.

Dr. Mahesh T.R has done B.E., in Computer Science & Engineering from Mysore University, M.tech in Computer Science & Engineering from DR. MGR Educational & Research Institute, Chennai and PhD from University of Allahabad in the area of Data Mining for IDS. He has published several articles in national as well as International journals. He is Prof and Head of Dept Computer science T. John Institute of Technology. Bengaluru His current research Interests lie in the areas of Wireless communications, Networks and algorithms, mobile computing, and multimedia and Communication.