# The Partition Algorithm of an Equivalent Queuing Model for Serial-Parallel Connection Servers

CHUNG-PING CHEN*, YING-WEN BAI** and CHENG-YU CHUNG**
Graduate Institute of Applied Science and Engineering, Fu Jen Catholic University*
Department of Electronic Engineering, National Taipei University of Technology*
Department of Electrical Engineering, Fu Jen Catholic University**
510 Chung Cheng Rd. Hsinchuang, Taipei County 24205
TAIWAN, R.O.C.
491598038@mail.fju.edu.tw, bai@ee.fju.edu.tw, tsauyow168@hotmail.com

*Abstract:* - This paper aims at analysis efficiency in estimating the performance of serial-parallel connection servers. We use a queuing model to represent the equivalent performance and service quality of the serial-parallel connection servers. The queue types of the connection servers can be classified as serial, parallel and tree connections which are represented by queues with serial, parallel and tree connections. We design the algorithm to simplify the equivalent serial-parallel queues. According to the equivalent queues we estimate the system response time computing of serial-parallel connection servers. We use a network simulation and analytical software tool to represent the equivalent performance of the queue. Our simulation uses various different service rates and arrival rates of the queue models and finds the system response time by linear regression. We also measure the average system response time in comparison with the simulation result to find out the service rates of the actual servers and evaluate the accuracy of the algorithm.

*Key-Words:* - Serial-parallel Network, Web Servers, Service Rate, Serial-parallel Connection Servers.

## 1 Introduction

When the network structure becomes more and more complicated, and Web service requests become bigger and bigger, this overall result generates a high blocking probability of the Web service and lengthens the system response time, with unacceptable results. In the designs of server architecture many improved methods have been put forth, as stated in the following examples. Upgrading the equipment using a multiple connection method increases the service rate by making use of a load balance mechanism in order to strengthen the Web service capability [1]. A Gigabit Ethernet between the network interface controller and the corresponding CPU raises the network flow and reduces the network transmission time [2]. Also, the use of a preemption mode improves the utilization of the network and handles the high priority class information in advance. Then the low priority class makes use of the blank in the idle period of flow, thus reducing the whole blocking probability of the network [3].

This paper uses a serial-parallel method to improve the service capacity and to reduce the problem of a high blocking probability, and, by making use of the results of experiments, predicts the performance and efficiency of the serial-parallel connection servers [4, 5].

The server performance can be enhanced by a multi-layer with serial-parallel connection. This method analyzes the error margin of the system response time between the simulation and the physical measurement of the serial-parallel connection servers [4, 5]. We use a serial-parallel queue to imitate the system response time of Web requests of both the simulation and the measurement. We also compare the error margin between the measurement and the simulation.

A single server measures the quality of service (QoS) of the response time, which can be divided into three kinds: Less than 0.1s, a low blocking probability, from 0.1-10s, a medium blocking probability and greater than 10s, a high blocking probability [6].

As for the network system response times, their factors of influence which induce sorting are as follows [7]:

Network system response time = Network Time + Web Site Time + Time of DNS + Web Page Size + Multiple clicks.

Network Time = Node Latency + Transmission Time.

Web Site Time = Queuing Time + Service Time.

Service Time = function (CPU performance + disk driver performance + network processing performance)

The organization of this paper is as follows: In Section 2 we explain the process of the partition algorithm of an equivalent serial-parallel queue model to represent a local area network. In Section 3 we use the queue model for serial-parallel connection servers. In Section 4 we use multiple serial-parallel Website servers and make some physical measurements of the experiment and obtain the simulation results from the software tool. In addition, we disern their error margin, if any. In Section 5 we draw our conclusions according to the results of the experiment.

## 2 The Queue Representation of the Serial-Parallel Connection Server

Fig. 1 shows the campus network connection. Each department links the teachers' and the students' PCs by the hub or the switch, and the hub of the department links to each switch of the building. The switch of each building links to the switch of each college. The switch of each college then links to a computer center high-speed switch. Fig. 2 shows the queue representation that configures the campus network connection.

The LAN connection has many switches from the user to the servers. The connection type of the Internet presents a dendroid structure. The server is a root, the switch is a tree branch, and the user is a leaf. We want to know the system response time and the number of users that connect in order to establish particular relationship. The user's waiting time for Web page service is related to the size of the Web page, and we can estimate the amount of the exchange by the size of the Web page and the number of servers and thus predict the system response time of the network and the service rate of the servers.

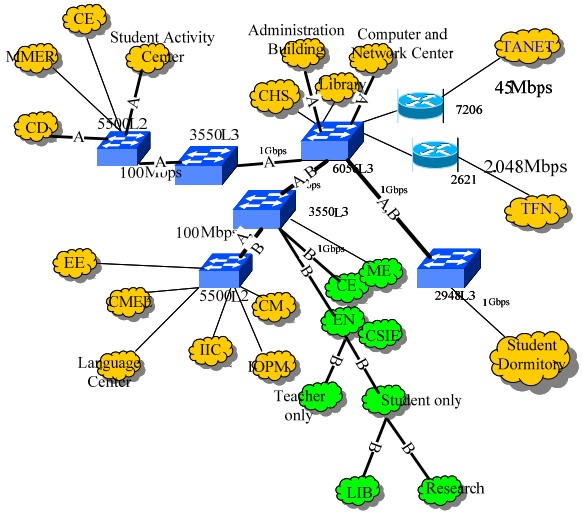


**Fig. 1 The campus network connection**

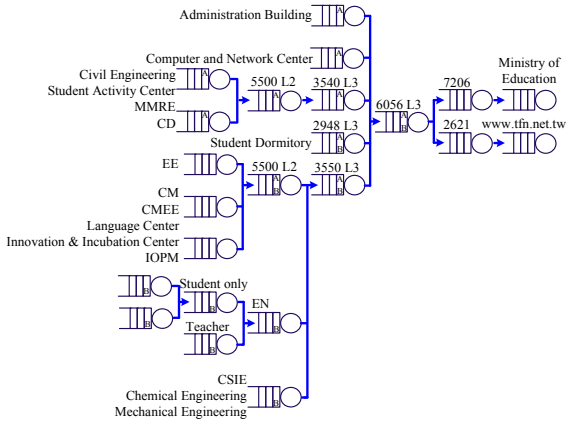


**Fig. 2 The equivalent queuing network for the campus network connection**

### 2.1 The Tree Structure of the Equivalent Queuing Network

We use the tree structure as the foundation of the equivalent queuing network in this paper. The tree is constituted by the node and branch elements. The node-related number of branches is the branch degree of this node. When a branch connects to a node, it provides a branch degree. When the direction of this branch is restricted with a node, the direction of each branch is outward from the respective node. The root of the inward branch degree is a null. In addition to a root, all nodes of the tree must just have an inward branch degree. The leaf called external node has a zero outward branch degree. Neither a root nor a leaf can call an internal node. As tThe father node's outward branch degree is null, the leaf has an inward branch degree. Nodes which have the same father are called brother nodes. The ancestry of any node in the path that goes to this node comes from the root, and the posterity is any node below this father node. The path consists of a continuous sequence of nodes. Among them, each node tightly answers the next node. The stratum of the node is the distance between the node and the root. The height of the tree is the stratum from the leaf to the root. Any conjoint structure is under the root of the subtree. The binary tree has all leaves without two subtrees.

### 2.2 The Extrovert Tree Conversion is a Binary Tree

We simplify a local area network into a single queue model to investigate the performance of the whole local area network. Because the tree for a large area network may have an uncertain diversity in a hierarchical binary tree, we need to transfer a general tree into a binary tree by means of the follow steps:

1. In addition to the branch of the leftest subtree, delete the branch of general tree rest.
2. Link all brothers' node usages branches together.
3. Turn the brother nodes into a right subtree, and turn the father and son nodes into a left subtree, thus forming a binary tree.

From the tree structures of a local area network, the left subtree will become a serial server node; the right subtree will become a parallel server node.

## 2.3 The Algorithm of the Equivalent Queues

After the binary tree transformation we simplify the root direction by the leaf. The node of the right subtree is a parallel node and that of the left subtree is a serial node. The father and son in the right subtree are two nodes which gradually move into a queue with a parallel connection. The father and son in the left subtree also are two nodes which gradually move into a queue with a serial connection. When the father node merges with the first subtree with two leaves in the left node, then we are computing a right node. Based on this procedure, we keep the simplification till the last whole large network is equalized into a single equivalent queue. Fig. 3 shows the flow chart of the algorithm.

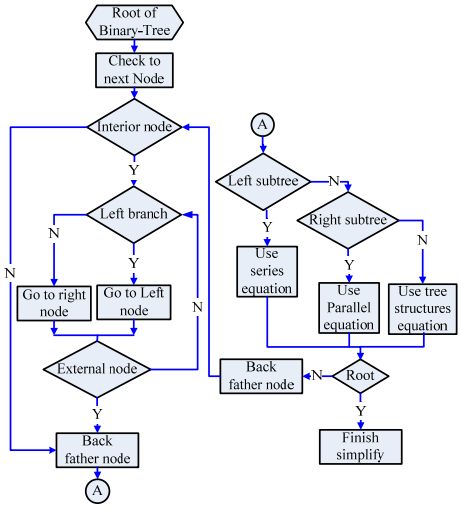


**Fig. 3 The flow chart of the partition algorithm.**

Figure 1 shows the original connection network. Part (a) of Fig. 4 shows the simplification steps of the equivalent queue. Step 2 shows the branch of the subtree farthest to the left, deletes the branch of the rest, and then links all brother node usage branches together, as shown in part (b) of Fig. 4. Step 3 shows the transformation of the binary tree, as shown in part (c) of Fig. 4. Step 4 shows the GKL and the PRS, both of which have three nodes, with a tree-like serial-parallel link, and use an equation to merge them as the G' and the P'. The MNO has three nodes with a parallel connection, and we therefore use a parallel equivalent queue to merge them as M', as shown in part (d) of Fig. 4. Step 5 establishes DG' and HM' by an equivalent queue to merge them as D' and H', and the three nodes of the JP'Q are tree-like serial-parallel links and use the serial-parallel equivalent queue to merge as J' , as shown in part (e) of Fig. 4. Step 6 shows the parallel H'IJ', and we use a parallel equivalent queue to merge them as F' with F again, as shown in part (f) of Fig. 4. Step 7 shows the parallel BCD'EF', and we use this parallel equivalent queue to merge B', as shown in part (g) of Fig. 4. Finally, this subtree serial merges with A and then gets the equivalent queue for these serial-parallel connection servers.

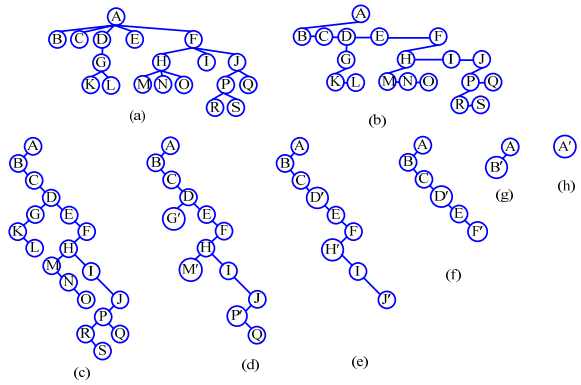


**Fig. 4 The simplification steps for the equivalent queue.**

Fig. 5 shows the pseudo code that simplifies a queueing network to an equivalent queue.

```
(Root of Binary-Tree)
(Check to next Node)
IF (Interior node) -> "Y" THEN
        IF (Left branch) -> "Y" THEN
                (Go to Left node)
        ELSE
                (Go to Right node)
ELSE
        (Back father node)
I F (External node) -> "Y" THEN
        (Back father node)
ELSE
        (Interior node)
IF (Left subtree) -> "Y" THEN
        (Use series equation)
ELSE
        IF (Right subtree) -> "Y" THEN
                (Use Parallel equation)
        ELSE
                (Use tree structures equation)
IF (Root) -> "Y" THEN
        (Finish simplify)
ELSE
(Back father node)
```

**Fig. 5** The pseudo code for the simplification of a queueing network.

## 3 The Queueing Model for the Serial-Parallel Connection Servers

To perform the simulation and measurement for the performance of the serial-parallel connection servers, we install one physical server computer network environment and use our algorithm to simplify this queueing network to obtain a single equivalent queue. We use the network software simulation system response time gained by the equivalent model for the measurement. The system definition and model parameter are shown in Table 1. The unit of measurement here is msec.

**Table 1 System definition and model parameters**

| Parameter | Description | Definition |
|---|---|---|
| $\lambda$ | Web request rate | Requests/sec |
| $\mu_{P_n}$ | Service rate of the n$^{th}$ parallel connection server | Requests/sec |
| $\mu_{Peq_n}$ | Equivalent service rate of the n$^{th}$ parallel connection queue | Requests/sec |
| $P_n$ | Probability of the n$^{th}$ parallel queue | |
| $\mu_{S_n}$ | Service rate of the n$^{th}$ serial connection queue | Requests/sec |
| $\mu_{Seq_n}$ | Equivalent service rate of the n$^{th}$ serial connection queue | Requests/sec |
| $\mu_{eq}$ | Service rate of the equivalent queue | Requests/sec |
| $E_{S_n}(T)$ | System response time of the n$^{th}$ serial connection queue | msec |
| $E_{P_n}(T)$ | System response time of the n$^{th}$ parallel connection queue | msec |
| $E_{Peq_n}(T)$ | Equivalent system response time of the n$^{th}$ parallel connection servers | msec |
| $E_{Seq_n}(T)$ | Equivalent system response time of the n$^{th}$ serial connection servers | msec |
| $E_{eq}(T)$ | Equivalent system response time of servers | msec |

We use the idea of a serial-parallel equivalent electric circuit as our analytical foundation and verify the performance by measuring, and we use approximate equations for the multiple stages of the serial-parallel network. At the beginning we use queueing networks and Markov Chains [8] to analyze serial-parallel networks with the following assumptions.

- All requests are first in first out of the system.
- The total number of requests in the system is unlimited.
- The requests can leave the system from another node.
- All service times are exponentially distributed.

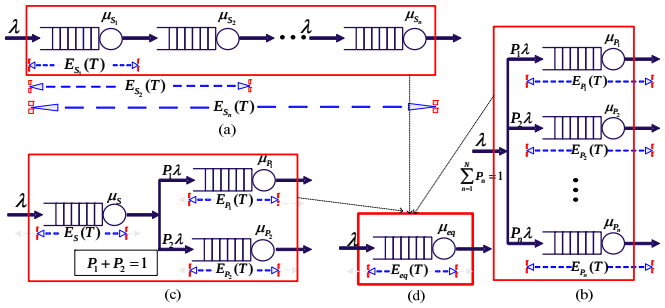


**Fig. 6 Equivalent model of serial-parallel connection servers**

If a customer demands a packet, we set the arrival rate at λ. As the packet arrives at the server set, it is assigned into each serial-parallel node one after another. Because there is no need to detect the network status or operation, the required service time is so short that we can neglect it. In Figs. 6(a), (b), (c), (d) the Web service, after connecting with the node, will have one small segment of waiting time in the buffer, and then this service is handled by the processor before it leaves the queue node.

The multiple serial queues under the same service rate are shown in Fig. 6(a).

Eq. (1) shows the total system response time.

$$E_{S_{eq_n}}(T) = E_{S_1} + E_{S_2} + \ldots + E_{S_n} = \frac{1}{\mu_{S_{eq}} - \lambda_{S_{eq}}} \quad (1)$$

When the service rates of servers are different, the equivalent service rate of a multiple serial queue is as shown in Eq. (2).

$$\mu_{S_{eq}} = \frac{1 + E_{S_{eq}}\lambda_{S_{eq}}}{E_{S_{eq}}} = \frac{1 + E_{S_{eq}}\lambda_{S_1}}{E_{S_{eq}}} = \frac{1 + \lambda_{S_1}(E_{S_1} + E_{S_2} + \ldots + E_{S_n})}{(E_{S_1} + E_{S_2} + \ldots + E_{S_n})} \quad (2)$$

If the serial servers have the same service rate, then the system response time is as shown in Eq. (3).

$$E_{S_{eq_n}}(T) = \frac{n}{\mu_{S_1} - \lambda_{S_1}} = \frac{1}{\mu_{S_{eq}} - \lambda_{S_{eq}}} = nE_{S_1} \quad (3)$$

If the serial servers have the same service rate, then the equivalent service rate is as shown in Eq. (4).

$$\mu_{S_{eq}} = \frac{\mu_{S_1} + (n-1)\lambda_{S_1}}{n} = \frac{1 + n\lambda_{S_1}E_{S_1}}{nE_{S_1}} \quad (4)$$

The service rates of the multiple parallels are the same. The system response time is as shown in Fig. 6(b).

$$E_{P_{eq_n}}(T) = \frac{1}{n}(E_{P_1} + E_{P_2} + \ldots + E_{P_n}) = \frac{1}{\mu_{P_{eq}} - \lambda_i} \quad (5)$$

Arrival rate

$\lambda_i = \lambda_{P_1} + \lambda_{P_2} + \ldots + \lambda_{P_n}$ , $\lambda_i = \lambda_{input}$

Dispatch probability

$P_1 + P_2 + \ldots + P_n = 1$

System response time

$$E_{p_{eq_n}}(T)=\frac{1}{n}\left(\frac{1}{\mu_{p_1}-P_1\lambda_i}+\frac{1}{\mu_{p_2}-P_2\lambda_i}+\ldots+\frac{1}{\mu_{p_n}-P_n\lambda_i}\right)=\frac{1}{\mu_{p_{eq}}-\lambda_i} \quad (6)$$

The servers have different service rates; their parallel equivalent service rate is as shown in Eq. (7).

$$\mu_{p_{eq}}=\frac{n+\lambda_i(E_{p_1}+E_{p_2}+\ldots+E_{p_n})}{(E_{p_1}+E_{p_2}+\ldots+E_{p_n})} \quad (7)$$

If the servers have the same service rate, then the system response time is as shown in Eq. (8).

$$E_{p_{eq_n}}(T)=\frac{1}{n}\left(\frac{n}{\mu_{p_1}-\lambda_{p_1}}\right)=\frac{n}{n\mu_{p_1}-\lambda_i}=\frac{1}{\mu_{p_{eq}}-\lambda_i} \quad (8)$$

If the servers have the same service rate, then the equivalent service rate is as shown in Eq. (9).

$$\mu_{p_{eq}}=\frac{n\mu_{p_1}-\lambda_i}{n}+\lambda_i=\frac{n\mu_{p_1}+(n-1)\lambda_i}{n} \quad (9)$$

The serial-parallel connection has the same service rate, as shown in Fig. 6(c).

The system response time of serial-parallel connection servers is shown in Eq. (10).

$$E_{T_{eq}}(T)=E_S+\frac{1}{2}(E_{p_1}+E_{p_2})=\frac{1}{\mu_{T_{eq}}-\lambda_{T_{eq}}} \quad (10)$$

The service rates of servers are different in serial-parallel connection servers.

$$\mu_{T_{eq}}=\frac{2+2\lambda_S E_S+\lambda_S(E_{p_1}+E_{p_2})}{2E_S+E_{p_1}+E_{p_2}} \quad (11)$$

If the service rates are the same, and the dispatch probabilities are the same, then the system response time is as shown in Eq. (12).

$$E_{T_{eq}}(T)=\frac{1}{\mu_S-\lambda_S}+\frac{1}{2}\left(\frac{2}{\mu_S-\frac{1}{2}\lambda_S}\right)=\frac{1}{\mu_{T_{eq}}-\lambda_S} \quad (12)$$

The equivalent service rate of serial-parallel connection servers with the same service rate is as shown in Eq. (13).

$$\mu_{T_{eq}}=\frac{(\mu_S-\lambda_S)(2\mu_S-\lambda_S)}{4\mu_S-3\lambda_S}+\lambda_S \quad (13)$$

# 4 The Comparison between the Simulation and the Measurement

To verify the system performance of the serial-parallel connection servers of a real network we use a local area network as the measuring environment. We use the WebServer Stress Tool measurement software to obtain the milt-node serial-parallel service rate of the ASP network. The service time is made up of the system response time of the serial-parallel connection servers. We set up the servers' IP addresses as shown in Fig. 7.

The client port requests the network to run a Web page service. We can neglect the Web page access time and only look at the serial-parallel connection server performance, so the Web page provides just simple data in mathematical operation.

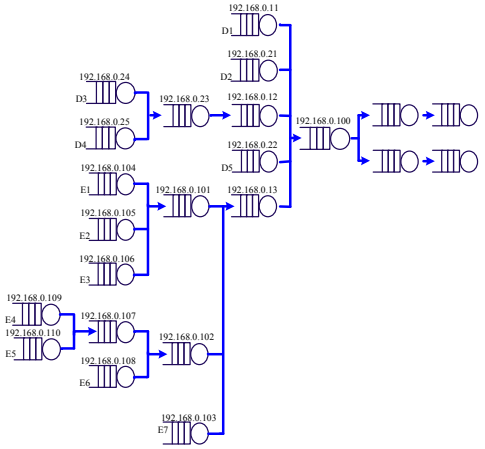


**Fig. 7 IP addresses of the serial-parallel connection servers**

In the actual measurement, in order to both adjust the service rate and transfer the simulation data of the server, we use the ASP Web page execution multiplication loop. The ASP Web page controls the service time of the server CPU through the execution multiplication loop. The multiplication loop has 1, 1 K, 10 K, 20 K, …, 190 K, 200 K times and both calculates and compares the system response time for every case.

## 4.1 Analytical Process and Measurement Results

Fig. 7 shows F networks, which uses a total of 19 servers, and there are 12 sets of serial loops. F1 by the IP of the server is 192.168.0.100 with 192.168.0.11, and the two servers are serial, F2 by the IP of the server is 192.168.0.100, with 192.168.0.21, and the two servers are serial. F3 by the IP of the server is 192.168.0.100 with 192.168.0.12, 192.168.0.23, 192.168.0.24, and the four servers are serial. F1-F12 altogether have 12 sets of serial loops. We carry out a parallel measurement with 12 sets of loops, ASP by 1 K, 10 Ks, 20 Ks, … , 190 Ks, 200 Ks. Each time the number of ASP includes 1 user, 5 users, … , 100 users, total with a measure of 441 times, at the rate of 20 minutes every time.

We collect measurement data to compare with the computation results of the value of the system response time from Eq. (1) to Eq. (5). The computed value of the system response time of each serial connection is from 1-100 users and takes the average value of the serial connection again. The value of the system response time of the measurement is from the formula of 1-100 users' system response time. The error margin between computing by equation and by measurement is smaller when CPU loads are low; this is due to the service rate and the arrival rate. The Web packet will not stay in the buffer, and the system response time is short. The error margin is bigger

when CPU loads are high; this is because the service rate approaches an arrival rate and has already had a Web packet to stay around the buffer, so the system response time increases. Fig. 8 shows the comparison.

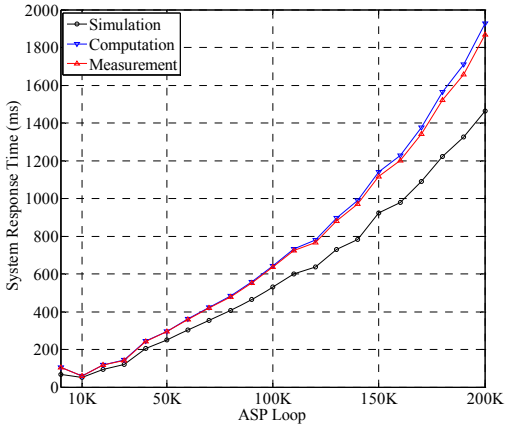


**Fig. 8 The system response times of simulation, computation and measurement**

Our experiment uses 19 servers; in addition there is a server that imitates 1-100 users, so that altogether we have at the same time 12 sets of serial loops that are parallel. Fig. 8 shows three curves, respectively from the equation, the simulation and the measurement. When the parallel number increases, the system response time is ASP=200 K, a high blocking probability, obviously from the 1463.196 ms of the F network. In Fig. 9, three curves show the simulation, computation and measurement system response times. Their average error margins are 19.27%, 18.20%, and 1.37% respectively.

## 5 Conclusion

We establish a parallel tree-like simple equation of equivalent queues to calculate the response time and the service rate of the system. Because of the tree characteristics of the Intranet, the network connectivity method can simply be classified as multiple-serial, multiple-parallel and serial-parallel connection. These characteristics make use of the algorithm of the binary tree in the Graph Theory and create the equivalent queuing model, the left subtree creates the equivalent serial equation, and the right subtree creates an equivalent parallel equation. The father and son are two groups. If we visit the first left subtree and then the right subtree and use a serial-parallel equivalent tree equation we make a simplification. The sequence of the simplification is as follows. First, the leaves of the left side have the initialization; the leaves of the right side go one after another toward the direction of the root. Second, we create the simplification procedure of the equivalent model.

In the low blocking probability, the network performance analysis shows that the error margin of its service rate is limited to 17.40%. The campus B network is analyzed as having a low blocking probability; the error margin of its service rate is limited to 16.25%.

When the Web service is busy, we use the parallel connection server to reduce the system response time. When user demand is larger than the system capability, we have a medium high blocking probability. By using parallel connection servers we can significantly reduce the system response time.

*References:*

[1] Ying-Wen Bai, Chia-Yu Chen, and Yu-Nien Yang, "A Two-Pass Web Document Allocation Method for Load Balance in Multiple Grouping of a Web Cluster System," *ICON'04, 12th IEEE International Conference on Networks,* Vol. 1, 2004, pp. 177-181.

[2] Nathan L. Binkert, Lisa R. Hsu, Ali G. Saidi, "Performance Analysis of System Overheads in TCP/IP Workloads," *Proceedings of the 14th International Conference on Parallel Architectures and compilation Techniques,* 17-21 Sept., 2005, pp.218-228.

[3] Zhen Zhao, Bryan Willman, Steven Weber and Jaudelice C.de Oliveira, "Performance analysis of a parallel link network with preemption," *Proceedings of International Conference on Information Sciences and Systems,* 22-24 March, 2006, pp.271-276.

[4] Ying-Wen Bai and Yu-Nien Yang, "An Approximate Performance Analysis and Measurement of the Equivalent Model of Parallel Queues for a Web Cluster with a Low Rejection," *Proceedings of the 14th IEEE International Conference,* 2006, pp.1-6.

[5] Chung-Ping Chen, Ying-Wen Bai and Yin-Sheng Lee, "Approximate Analysis and Measurement of Equivalent Model for Tree Connection of Web Server Systems," *Proceedings of the 19th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2007),* Cambridge, MA, USA, Nov. 19-21, 2007, pp.91-96.

[6] http://www.paessler.com/webstress/

[7] Keith W. Ross, James F. Kurose, "*Computer Networking: A Top-Down Approach Featuring the Internet*" Reading, MA. Addison-Wesley, 2001.

[8] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S. Trivedi,"*Queueing Networks and Markov Chains*" Wiley-Interscience, 1998.