Optimizing Scheduling Policy of Queuing Systems in Heterogeneous Environments using Fuzzy Reasoning

Essam Natsheh† and Khalid A. Buragga†

†College of Computer Sciences & IT, King Faisal University, Hofuf 31982, Saudi Arabia

Summary
This paper considers a queuing model with batch Poisson input and two heterogeneous servers, where the service times are exponentially distributed. The faster server is always on, but the slower server is only used when the queue length exceeds a certain level. Traditionally, this was done through scheduling policy. In this paper an enhanced algorithm, called fuzzy scheduling policy, is suggested using fuzzy reasoning to achieve the benefits of heterogeneity in service rates. Uncertainty associated with queue congestion estimation and lack of mathematical model for estimating heterogeneity in service rates makes the fuzzy scheduling algorithm the best choice. Extensive performance analysis via simulation showed the effectiveness of the proposed method for congestion detection and avoidance improving overall queuing systems.

Key words:
Queuing theory, heterogeneous service rates , optimal control , optimal customer allocation, fuzzy reasoning

1. Introduction

Multi-server queuing systems are of interest in applications such as multiprocessor systems as well as communication networks. There are two kinds of server heterogeneity: one where the servers have different mean service rates and another where the servers have different service functions, as when some servers serve only a certain class of customers. For clarity, we call the two kinds of service *server heterogeneity in service rates* and *server heterogeneity in service functions*. This paper tackles the problem of optimal control of queuing systems with heterogeneous servers using fuzzy reasoning algorithm. The system objective is to assign customers dynamically to idle servers in order to minimize the average cost of holding customers [1-2].

We examine the case of queuing systems with server heterogeneity in service rates. This case has been examined in the works of Nobel and Tijms [3] and Viniotis and Ephremidis [4]. The fuzzy reasoning approach is totally different from classical approaches.

In addition to the papers noted above, results on optimal routing of customers to multiple servers also have been reported elsewhere. In [5] and [6], Liu described an approach to improve the performance of the M/M/2 queue (a queue with two parallel servers and exponential inter-arrival times (iat) and service times) by replacing its homogeneous servers with heterogeneous servers. The author showed that there exists an optimal ratio at which the first order and second order metrics reach their optimums.

The problem of optimal allocation of customers in a two server queue with heterogeneous service rates and re-sequencing is addressed by Varma [7] and Xia and Tse [8]. The re-sequencing constraint ensures that the customers leave the system in the order in which they entered it. A comprehensive discussion on optimal service control of queuing systems can be found in the survey papers of Stidham [9] and Dragicevic and Bauer [10].

Most of the work mentioned thus far employs conventional stochastic threshold policy. In this paper, we propose an entirely new threshold policy using *fuzzy reasoning* and show via simulation that this new threshold policy efficiently solves cases intractable with classical threshold policy.

The rest of this paper is organized as follows. Section 2 summarizes problem description. Followed by the fuzzy scheduling policy as a new scheduling policy for queuing systems, performance analyzes of the proposed algorithm, and finally the conclusions.

2. Problem Description

The queuing system considered here is as follows: Customers arrive at the buffer in a Poisson stream with constant rate. The buffer has limited capacity and the order of service is irrelevant. The buffer is served by two exponential servers with *different* mean service rates μ_1 and μ_2 where $\lambda < \mu_1 + \mu_2$. Without loss of generality, it is assumed that: $\mu_1 > \mu_2$.

The problem is to assigns customers to idle servers dynamically so as to minimize the sum of waiting time in queue and service time of the customers.

The sojourn time is the sum of waiting time in queue and service time. By Little's theorem [11], the system objective is equivalent to minimizing the mean number of customers in the system. Another objective could be to minimize the average holding cost if we assume a holding

Manuscript received April 5, 2010
Manuscript revised April 20, 2010

cost per customer per unit time. This process is a continuous time Markov decision process.

Lin and Kumar [12] prove that there exists an optimal policy and it is of the *threshold* type. Specifically, the faster server should be fed a customer from the buffer whenever the server is available for service, but the slower server should be utilized if and only if the queue length exceeds a critical threshold value *n*. Walrand [13] gives the same result a simpler proof, using a probabilistic argument. Viniotis and Ephremidis [4] extend the same result with less restrictions. The last two papers, however, do not facilitate the calculation of the threshold. A threshold policy is of the type of *faster server first allocation* [2].

3. Fuzzy Scheduling Policy for Queuing Systems

In this section, concepts and rules of the proposed fuzzy scheduling policy algorithm for queuing systems are introduced. In the following two subsections, we studied the effect of some queue parameters on scheduling policy algorithm. These parameters are used in subsection 3 to create the rules of the proposed fuzzy scheduling policy. Method to design their membership functions is presented in the later subsection. Overall system design and its implementation complexity are presented in subsection 5 and 6.

3.1 Effect of Current Queue Size on Scheduling Policy

Current queue size q_c is the most used indicator in scheduling policy for estimating the probability of dropping the incoming packets. The drop probability p_d can be calculated as [14]:

$$p_d = \frac{2N^2}{(CT_p + q_c)^2} \qquad (1)$$

where N is a load factor, C is a transmission capacity (in packets/seconds) and T_p is a propagation delay (in seconds). Assuming a 10 Mbps (2500 packets/sec) transmission capacity with a 100 msec propagation delay, Fig. 1 shows the relation between p_d and the load for various queue sizes. It is evident that the probability of a packet dropping increases as the load increases. More packets in the queue wait for processing as load increases. Thus, it can be stated that when the used space of the queue is high, *srv2Status* must be *Yes* and vice versa. Consequently the following rules are proposed:

R1: If q_c is low then *srv2Status* is *No*
R2: If q_c is medium then *srv2Status* is *No*
R3: If q_c is high then *srv2Status* is *Yes*

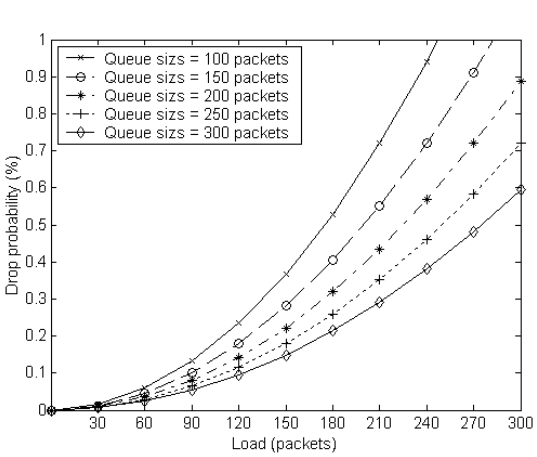

Fig. 1 Drop probability for the coming load.

3.2 Effect of Node Neighborhood Density on Scheduling Policy

In computer networks, the traffic is categorized as: data packets and control messages. The control messages are used to continuously update the nodes about the topology changes (new created or lost links). For example, if a node has two neighbors that means it will receive two Hello messages every second from them. Besides, receiving a route request messages, a route breaks messages, or data packets. If that node has ten neighbors, this means it will receive, in every second, ten Hello messages beside bulk amount of control messages and data packets. Hence, it is clear that the traffic pass through the nodes with few neighborhoods is less than the others with many neighbors.

In Eq.(1), the load N can be written as:

$$N = \sum_{i=0}^{n} \lambda_i \qquad (2)$$

where λ_i denote flow's rate from the neighbor node i and n is the number of neighbors. The congestion will happen at:

$$p_d = 1 \quad if \quad \sum_{i=0}^{n} \lambda_i > C \; and \; q_c = q_m \qquad (3)$$

where q_m is the maximum queue size. Hence, if the neighbors' density (n_d) of a node's is high, the node's queue will be full quickly and increases the probability of congestion and vice versa. Consequently the following rules are proposed:

R4: If n_d is low then *srv2Status* is *No*
R5: If n_d is medium then *srv2Status* is *Yes*
R6: If n_d is high then *srv2Status* is *Yes*

3.3 The Rule-base for Fuzzy Scheduling Policy

To fulfill the fuzzy sets theory, the previous six rules (R1 to R6) can be combined within a 2-dimensional rule-base to control scheduling policy adaptively as presented in Table 1. For example, according to Table 1 the first rule is:

IF q_c is *Low* AND n_d is *Low* THEN *srv2Status* is *No*

Table 1: Fuzzy scheduling policy rules for queuing systems

		n_d		
		Low	Medium	High
q_c	Low	No	No	No
	Medium	No	Yes	Yes
	High	Yes	Yes	Yes

3.4 Membership Functions for the Fuzzy Variables

After defining the fuzzy linguistic 'if-then' rules, the membership function (MF) corresponding to each element in the linguistic set should be defined. For example, if the queue size is 5 k bytes and q_c equal to 2 k bytes, using conventional concept, it implies q_c is either 'low' or 'medium' but not both. In fuzzy logic, however, the concept of MFs allows us to say the q_c is 'low' with 80% membership degree and 'medium' with 20% membership degree.

The MFs we propose to use for the fuzzy inputs (q_c, n_d) and the fuzzy output (*srv2Status*) are illustrated in Fig. 2. These MFs are used due to their economic value of the parametric and functional descriptions. In these MFs, the designer needs only to define one parameter; *midpoint*. These MFs mainly contain the *triangular* shaped MF. This function is specified by three parameters (a, b, c) as follows:

$$triangle(x;a,b,c)=\begin{cases}(x-a)/(b-a) & for\ a\le x\le b\\ (c-x)/(c-b) & for\ b\le x\le c\\ 0 & elsewhere\end{cases} \quad (4)$$

where a = *midpoint*/2, b = *midpoint*, c = 3 × *midpoint*/2 and x is the input to the fuzzy system. The remaining MFs are as follows: Z-shaped membership to represent the whole set of low values and S-shaped membership to represent the whole set of high values.

Maxpoint is the maximum queue size in q_c−MF (Table 1), and it is the number of the network's nodes in n_d −MF.

Midpoint of q_c−MF is a threshold that indicates whether the queue is going to be full soon. The threshold is simply set to 60% of the queue size. The optimal value for this variable depends in part on the maximum average delay that can be allowed by the nodes.

Midpoint of n_d−MF is a threshold that indicates whether the congestion will happen. The threshold is simply set to 60% of the expected flow's rate from the neighbor nodes.

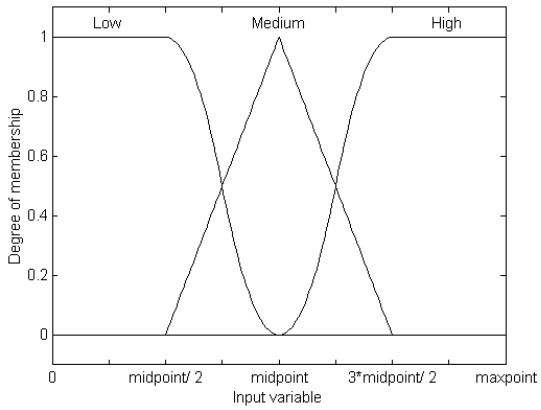

Fig. 2 Membership functions used for the fuzzy variables.

3.5 Fuzzification, Inference and Defuzzification

The fundamental diagram of the fuzzy system is presented in Fig. 3. Fuzzification is a process where crisp input values are transformed into membership values of the fuzzy sets (as described in the previous section). After the process of fuzzification, the inference engine calculates the fuzzy output using the fuzzy rules described in Table 1. Defuzzification is a mathematical process used to convert the fuzzy output to a crisp value; that is, *srv2Status* value in this case.

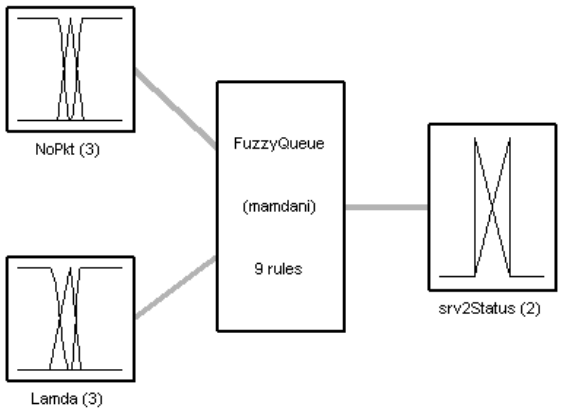

Fig. 3 Block-diagram for the basic elements of the proposed fuzzy scheduling policy.

There are various choices in the fuzzy inference engine and the defuzzification method. Based on these choices, several fuzzy systems can be constructed. In this study, the most commonly used fuzzy system, *Mamdani* method, is

selected; for further details on this system see [15].

Formally, the rule-base (Table 1) of the fuzzy scheduling policy can be rewritten in the following format:

IF q_c is A_{i1} AND n_d is A_{i2} THEN *srv2Status* is B_i (5)

where A_{i1}, A_{i2}, and B_i are the linguistic labels *Low*, *Medium*, *High*, *Yes* and *No* of the i^{th} rule.

Mamdani method is used as the fuzzy inference engine, where Min ($\wedge$) operator is chosen as AND connective between the antecedents of the rules as follows:

$$\tau_i = A_{i1}(x_1) \wedge A_{i2}(x_2) \quad (6)$$

where τ_i is called the *degree of firing* of the i^{th} rule for the input values: $x_1 = q_c$ and $x_2 = n_d$. The next step is the determination of the individual rule output F_i (fuzzy set) which is obtained by:

$$F_i(y) = \tau_i \wedge B_i(y) \quad (7)$$

The third step is the aggregation of rules outputs to obtain the overall system output F (fuzzy set), where Max ($\vee$) operator is chosen as OR connective between the individual rules:

$$F(y) = \vee_i F_i(y) = \vee_i (\tau_i \wedge B_i(y)) \quad (8)$$

To use this algorithm in the queuing systems environments, a fourth step needs to be added to get a crisp single value for *srv2Status*. This process is called *defuzzification*. Center of area (COA) [15] is chosen as the defuzzification method as follows:

$$srv2Status = \frac{\sum_{j=1}^{m} F(y_j) \times y_j}{\sum_{j=1}^{m} F(y_j)} \quad (9)$$

here y_j is a sampling point in the discrete universe output F, and $F(y_j)$ is its membership degree in the MF.

3.6 Implementation Complexity of the Fuzzy Algorithm

Using fuzzy logic algorithm with scheduling policy of queuing systems we may achieve comparable or better run-time computation than purely conventional methods. This can be achieved using one of the following methods:

1. *Lockup table:* The input-output relationship of the fuzzy reasoning engine for the fuzzy scheduling policy can be stored as a lookup table which will result in a very fast execution.
2. *Fuzzy logic interpreter:* Instead of implementing the fuzzy system using a high level language with its local interpreter and compiler, an interactive computing environment based on a fuzzy logic interpreter can be used to minimize the calculation overhead [16].
3. *Dedicated fuzzy hardware:* Fuzzy systems based on dedicated hardware can deliver much higher performance than those based on general-purpose computing machines [17].

4. Performance Analysis of the Proposed Fuzzy Scheduling Policy

4.1 Simulation Experiment

To simulate the heterogeneous environment, we simulate a buffer served by two servers with *different* mean service rates μ_1 and μ_2 as shown in Fig. 4.

To simulate the conventional *threshold* scheduling policy with optimal threshold value, when server 2 is available, a queuing customer is allocated to it only if the queue size reaches or exceeds the threshold value (Thr). For comparison purpose, we run the simulation with two different threshold values: Thr = 5 and Thr = 6.

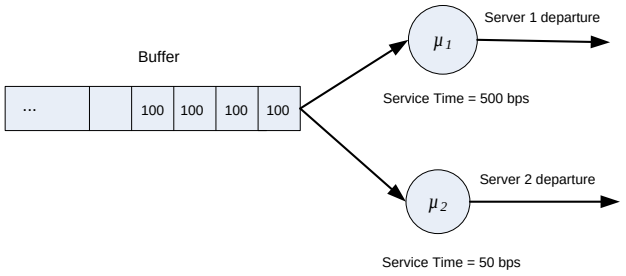

Fig. 4 The arrangement of the simulation environment.

In our simulation we used 200 sources for denoting flow's rate from the neighbor nodes.

Simulation of the heterogeneous environment, the conventional *threshold* scheduling policy, and the proposed fuzzy scheduling policy was done using C++ programming language.

Fig. 5 and Fig. 6 shows the arrival functions of threshold and fuzzy scheduling policies, while Fig. 7 and Fig. 8 shows the departure functions.

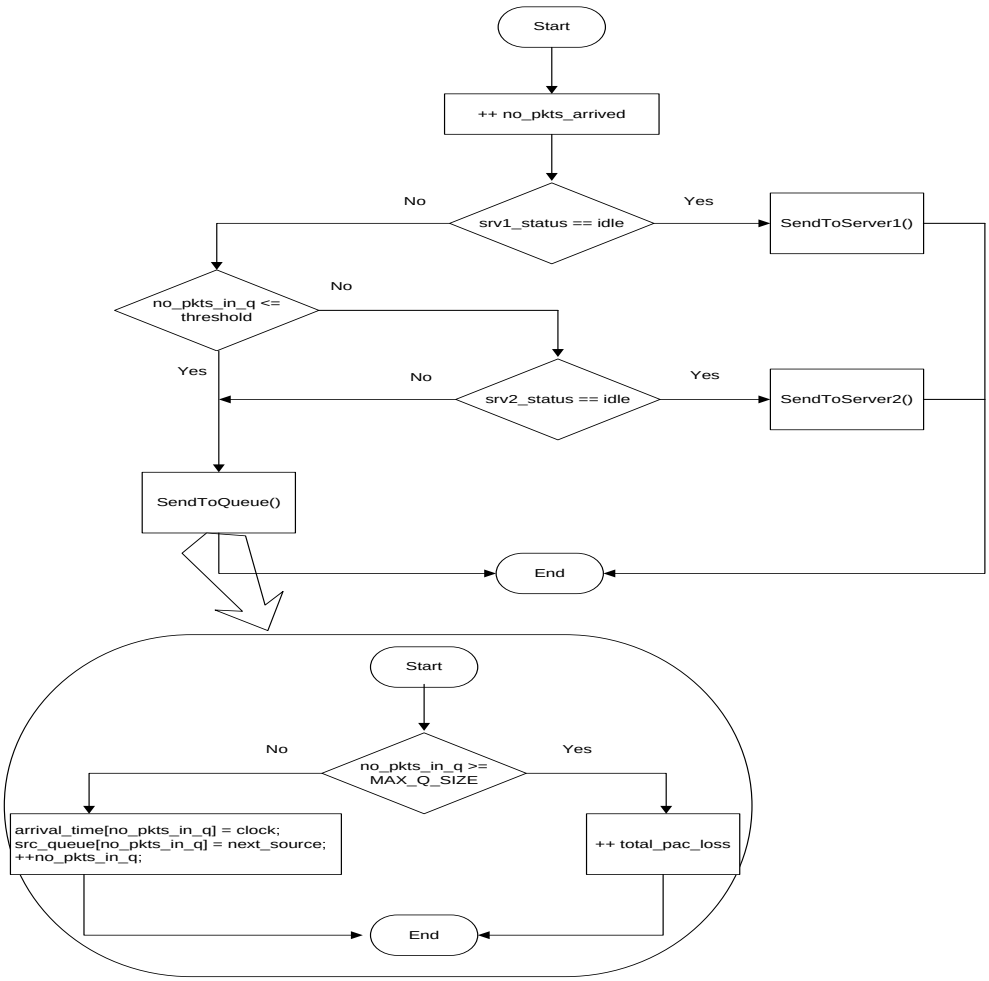

Fig. 5. Arrival function of threshold scheduling policy.

```
void srv2_departure (void)
{     ++no_pkts_departed;
      ++sentToSrv2;
      if (no_pkts_in_q <= threshold)
              {    srv2_status = 0; // free -- idel
                   event_time[next_source][next_event] = infinite;
              }
              else if (no_pkts_in_q > threshold
              {    --no_pkts_in_q;
                   delay = clock - arrival_time[0];
                   total_delay += delay;
                   tx_src = src_queue[0];
                   if (src_queue[0] != next_source)
                   event_time[next_source][next_event] = infinite;

                   swap();
                   // schedule the next departure
       event_time[tx_src][next_event] = clock + service_time2;
                  }
} // end-departure
```

Fig. 7 Departure function of threshold scheduling policy.

```
void srv2_departure (void)
{   ++no_pkts_departed;
    ++sentToSrv2;
     srv2Status = fuzzify (qc, nd);
     if (srv2Status >=0.5)
        srv2_Decision = 1;          // you can use srv2 if its free
     else srv2_Decision = 0;        // no need for slower srv2
     if (srv2_Decision == 0)
     {     srv2_status = 0;      // free -- idel
           event_time[next_source][next_event] = infinite;    }
     else if (srv2_Decision == 1)
     {       --no_pkts_in_q;
             delay = clock - arrival_time[0];
             total_delay += delay;
             tx_src = src_queue[0];
             if (src_queue[0] != next_source)
                event_time[next_source][next_event] = infinite;
             swap();
      event_time[tx_src][next_event] = clock + service_time2;
     }
} // end-departure
```

Fig. 8 Departure function of fuzzy scheduling policy.

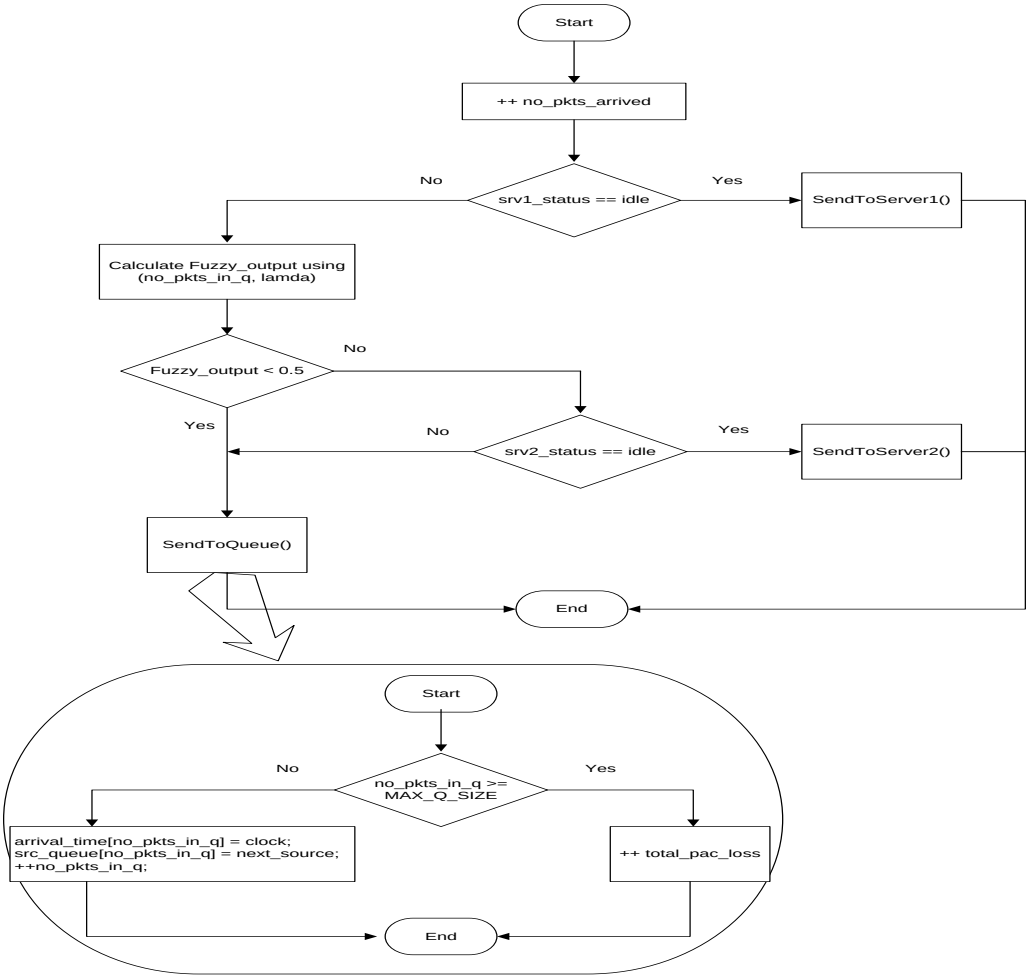

Fig. 6. Arrival function of fuzzy scheduling policy.

4.2 Simulation Results and Evaluations

Figures from 9 to 13 provide comparison between threshold and fuzzy scheduling policies.

From the Fig. 9, Fig. 10 and Fig. 11, it's noticed that as the load (rate of data transmitted) through the system increases, the average packets delay, loss ratio and the average buffer utilization increases linearly. However, as the load reaches the network's capacity, the buffers begin to fill. This increases number of packets sent to server 2, as shown in Fig. 12 and Fig. 13. Once the buffers begin to overflow, packet loss occurs. Increases in load beyond this point increase the probability of packet loss as shown in Fig. 10.

Taking the threshold policy as a base system, Table 2 shows the percentage of improvement of fuzzy method over the original threshold method.

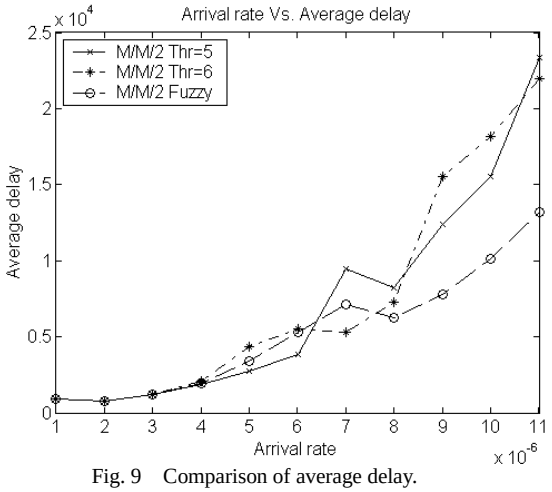

Fig. 9 Comparison of average delay.

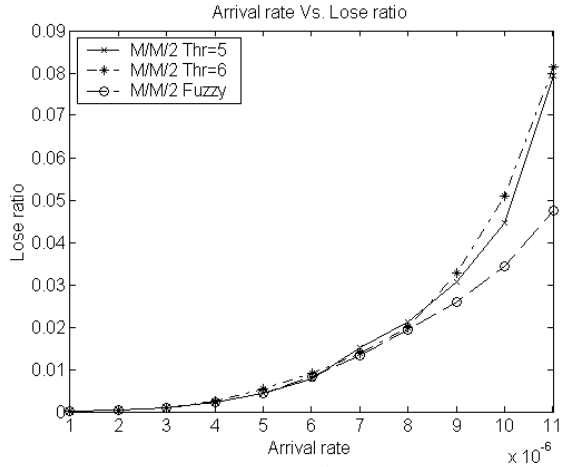

Fig. 10 Comparison of loss ratio.

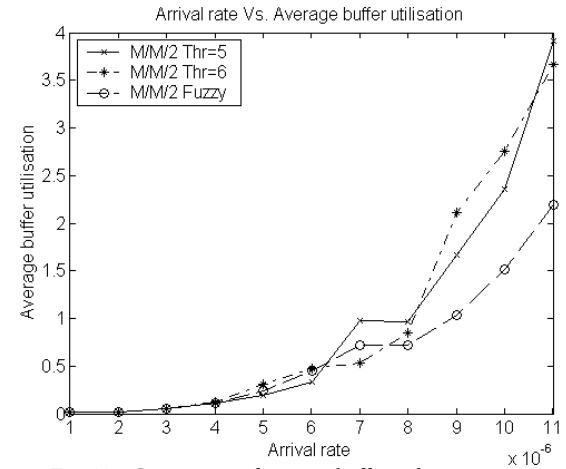

Fig. 11 Comparison of average buffer utilization.

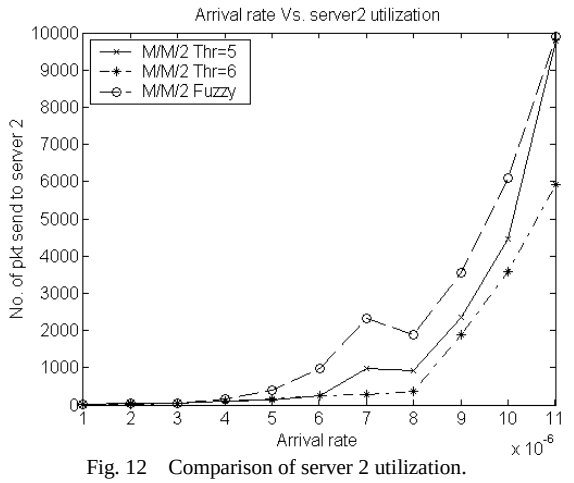

Fig. 12 Comparison of server 2 utilization.

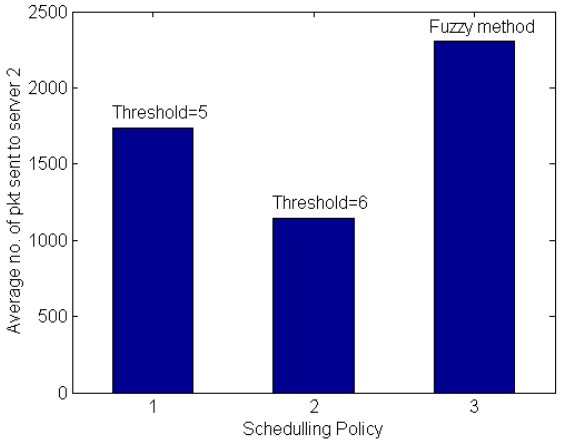

Fig. 13 Comparison of average server 2 utilization.

Table 2: Percentage of improvement of fuzzy method over threshold method

	Average delay	Loss ratio	Buffer utilization	Server 2 utilization
M/M/2 with thr = 5	27.88 %	24.13 %	33.10 %	32.75 %
M/M/2 with thr = 6	30.25 %	27.88 %	34.98 %	99.95 %

5. Conclusion

In this study, the problem of optimal allocation of customers in a two server queue with heterogeneous service rates is addressed. A novel scheduling policy of queuing systems based on fuzzy reasoning is suggested. The fuzzy reasoning have capabilities of adapting to high variability and uncertainty in such heterogeneous environments. The proposed fuzzy scheduling policy is contrasted with a well-known classical scheduling policy. From the simulation results, the efficiency of the proposed fuzzy scheduling policy in terms of average delay, loss ratio, buffer utilization and servers utilization are pronounced than classical scheduling policy.

Acknowledgment

The authors would like to express their appreciation to Deanship of Scientific Research at King Faisal University for supporting this research.

References

[1] R. Zhang and Y. A. Phillis, "Fuzzy routing of queueing systems with heterogeneous servers," in *Proc. Int. Cong.*

Robot. Automat. Albuquerque, NM, vol. 3, pp. 2340–2345, Apr. 1997.

[2] R. Zhang and Y. A. Phillis, "Fuzzy control of queueing systems with heterogeneous servers", *IEEE Trans. Fuzzy Systems*, vol.7, no.1, pp. 17-26, Feb. 1999.

[3] R. D. Nobel and H. C. Tijms, "Optimal control of a queueing system with heterogeneous servers and setup costs," *IEEE Trans. on Automatic Control*, vol. 45, Issue: 4, pp. 780-784, Apr. 2000. DOI: 10.1109/9.847122.

[4] I. Viniotis and A. Ephremidis, "Extension of the optimality of the threshold policy in heterogeneous multiserver queueing systems," *IEEE Trans. Automat. Contr.*, vol. 33. pp. 104–109, Jan. 1988. DOI : 10.1109/9.371.

[5] X. Liu, "An approach to reduce the Erlang C probability of the M/M/2 system", Proc. of IEEE International Conference on Communications, vol. 2, pp. 994-998, 2005. DOI: 10.1109/ICC.2005.1494498.

[6] X. Liu, "Optimality of the Second Order Metrics of the M/M/2 System with Heterogeneous Service Rates", Proc. of IEEE International Conference on Communications, pp. 6350-6355, Glasgow, 2007. DOI : 10.1109/ICC.2007.1051.

[7] S. Varma, "Optimal allocation of customers in a two server queue with resequencing", IEEE Trans. Automat. Contr., vol. 36 , Issue:11, pp. 1288-1293, 1991. DOI: 10.1109/9.100940.

[8] Y. Xia and D. N. C. Tse, "On the Large Deviations of Resequencing Queue Size: 2-M/M/1 Case", IEEE Trans. on Information Theory, vol. 54, Issue:9, pp. 4107-4118, 2008. DOI: 10.1109/TIT.2008.928234.

[9] S. Stidham Jr., "Computing optimal control policies for queueing systems," 24th IEEE Conference on Decision and Control, vol. 24 , Part: 1, pp. 1810-1814, 1985. DOI: 10.1109/CDC.1985.268874.

[10] K. Dragicevic and D. Bauer, "A survey of concurrent priority queue algorithms", Proc. of IEEE International Symposium on Parallel and Distributed Processing, pp. 1-6, 2008. DOI: 10.1109/IPDPS.2008.4536331.

[11] L. Kleinrock, Queueing Systems—Volume I: Theory, Wiley-Interscience, New York, 1 edition, 1975. ISBN-13: 978-0471491101.

[12] W. Lin and P. R. Kumar, "Optimal control of a queueing system with two heterogeneous servers," IEEE Trans. Automat. Contr., vol. AC-29, pp. 696–703, Aug. 1984.

[13] J. Walrand, "A note on 'optimal control of a queueing system with two heterogeneous servers'," *Syst. Contr. Lett.*, vol. 4, pp. 131–134, 1984.

[14] E. Plasser, T. Ziegler and P. Reichl, "On the Non-Linearity of the RED Drop Function", *Pro. of the 15th international conference on Computer communication*, vol. 1, pp. 515-534, India, Aug. 2002.

[15] R. R. Yager and D. P. Filev, "*Essentials of Fuzzy Modeling and Control*", Ch. 4, pp. 109-153, John Wiley & Sons, 1994.

[16] P. P. Bonissone, "A compiler for fuzzy logic controllers," in *Fuzzy Eng. Toward Human Friendly Sys. IFES'91* IOS Press, 1992.

[17] [17] D. L. Hung, "Dedicated Digital Fuzzy Hardware", *IEEE Micro*, vol. 15, Issue 4, pp. 31-39, Aug. 1995.

Essam Natsheh obtained his PhD in Communications and Networks Engineering from University Putra Malaysia in 2006. Currently, he is an Assistant Professor at the Computer Information Systems Department, College of Applied Studies and Community Services, King Faisal University (Saudi Arabia). Essam has more than ten years of teaching and research experiences in Malaysia and Saudi Arabia. Also, he has more than 15 publications in refereed journals at international level. His research interest is mobile ad-hoc networks, in particular, the development of a new routing algorithm for this type of networking.

Khalid Buragga received the B.Sc. in Computer Information Systems (CIS) from King Faisal University, Saudi Arabia, and the M.Sc. in CIS from University of Miami, USA, and the Ph.D. in Information Technology from George Mason University, USA. He is currently an assistant professor at college of Computer Sc. & IT, King Faisal University. His general research interests span the areas of software design, development, quality, and reliability; business process re-engineering, integrating systems; communications, networking and signal processing.