USING PRIORITY SCHEDULING IN PROVIDING THE DIFFERENTIATED SERVICE ROUTER WITH PREMIUM SERVIC

استخدام الجدوله ذات الأولويه لتزويد موجهات الخدمات المميزه بخدمات ذات افضليه

Soha Eld*, Tarek Kamel* And Samir Shahine**

خلاصة :

يتناول البحث دراسة إمكانية استخدام الجدوله ذات الأولويه لتزويد موجهات الخدمات المميزه بخدمـــه ذات افضليه. ويعتمد البحث على زمن التأخير فى الخدمات المميزه كمقياس لجودة الخدمه. يقدم البحـــث تحليـــلا للخدمات المميزه زمن التأخير فى الخدمه وتأثير حركه الاشارات الاخرى الغير مميزه على نفس العقـــده. وقـــد تم التحليل عن طرق المحاكاه. وقد تم مقارنة بين نتائج المحاكاه ونتائج التحليل النظرى مع تفسير ســـبب الاختـــلاف بينهما.

Abstract

In the present work, the possibility of using the priority scheduling mechanism for providing differentiated service node with premium service was investigated. The metric that was focused on was the average delay experienced by the packets of the premium service. The effect of the existence of another traffic transiting the same node was also investigated. The investigation was carried out using the sim++ simulator that enabled to study the relation between the average delay of the premium service with respect to its mean arrival rate. A comparison between the obtained simulation results and the theoretical calculations is presented and discussed.

* Electronics Research Institute, Cairo, Egypt.

** Faculty of Engineering, Cairo University, Cairo, Egypt.

Accepted October 22, 2000

scheduling, the router maintains separate FIFO queues for each class and assigns priorities in serving these queues. Both of the differentiated service mechanisms use bandwidth as the resource that is being requested and allocated.

Clarck and Wrolawski defined an assured service, [Clark, 97]. The assurance that the user of such service receives, such traffic that is unlikely to be dropped as long as it stays within the expected capacity profile. An assured service traffic flow may exceed its profile, but the excess traffic is not given the same assurance level. The general approach of this mechanism is to define a service profile for eaeh user, and to design a meehanism in the routers that favors traffic which is within those service profiles, [Heinanen, 99]. The core of the idea is very simple. It is to monitor the traffic of each user as it enters the network and tags packets as being "in" or "out" of their service profile. Then, it preferentially drops traffic that is tagged as being "out", at each router, if congestion occurs.

On the other hand, premium service levels, which are proposed by Jacobson, (1997) is provisioned according to a peak capacity profile that are strictly not oversubscribed and that is given its own high priority queue in routers. Premium service can be realized as the capacity whieh the customer expects to be there whenever that service is needed, though it might be idle a great deal of time. On the other hand, whenever this capacity is not being used, it is available to best effort traffic, [Nichols, 99]. The premium service is a paid for service with hard time requirements which should not be affected by other traffic transiting the node.

The present work is devoted to investigate how the internal router can provide a premium service. In other words, how the internal router should treat the packet that has its premium bit set. In fact, no particular scheduling mechanism exists, that provides the application with premium service, in the router. Most of these scheduling mechanisms are categorized as a priority scheduling. For this reason, the priority scheduling has been chosen as the routing mechanism to be studied in the present work. The study will be focused on the behavior of the average delay experienced by the packets of Premium service transiting a single node. Moreover, the effect of the existence of another traffic that is transiting the same node will be studied. This traffic may be considered as a best effort service.

This paper is divided as follows: Section 2, describes our algorithm of the priority scheduling mechanism. Section 3, provides a description of the implementation of this algorithm using Sim++. Section 4 contains our results and the discussion where Section 5 will conclude the paper.

2. Description of the proposed Algorithm

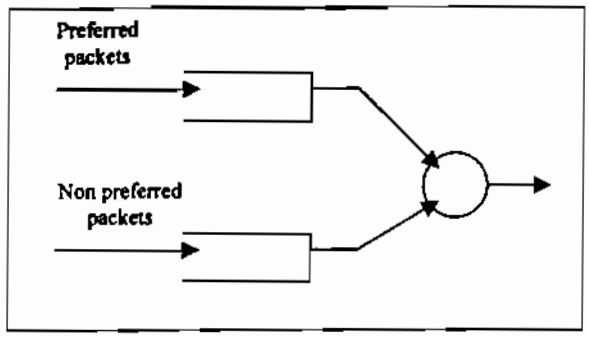

Fig (1): Priority Scheduling

The priority scheduling has been theoretically investigated [Sahu, 99]. The most important conclusion that has been reached is that the average delay of the premium service is dependent on the mean arrival rate of the premium service. Meanwhile, the average arrival rate of the best effort service has no effect on the average delay of the premium service. The average delay of the premium service is naturally increased as the mean arrival rate of the same service is increased. While increasing the mean arrival rate of the best effort service has no effect on the average delay of the premium service. This is because it has been assumed that the best effort service would never be served as long as the premium service exists.

In priority scheduling mechanism, a router maintains separate FIFO queues for each class and assigns priorities for serving these queues. Let us consider the case of a single internal router with two traffic classes where one class is given preference over the other as shown in Fig (1). That means, the router mechanism distinguishes between two classes of packets. These two classes are preferred and non-preferred. Naturally, the premium service is represented by the preferred service with the higher preference, while the non- preferred service will represent the best effort service.

The queues are modeled by using Markov queuing model with Poisson process represents the traffic model where the service model depends on the type of Markov queue that is being used. It is well known that Poisson arrivals have exponential inter arrival time which means that, the time between two arrivals is exponentially distributed. An exponential distribution with mean u , which is the average time between two packet arrivals, can be obtained from the uniformly distributed random variable U in [0,1] by the transformation:

$$f(u) = - u \log U \qquad \text{Eq. (1)}$$

So, if a packet arrives at time t, then the next packet can be scheduled at time t + f(u). It is very important to keep track of the time at which a packet of each class enters its corresponding queue. This will help the average delay computations. Moreover, gives a good understanding of the behavior of the system. It is also important to keep track of the time at which a packet leaves its queue for the average delay computation and the time spent by a packet in a queue should be added to the simulation time.

In the present work, the constructed algorithm is composed of the following modules:

The first module of the present algorithm is to generate two services with the parameters and variables associated with their packets. Each service has a service id, upon which the two services are differentiated, where service (1) is the preferred service and service (2) is the non-preferred service. Each service has its own buffer where each buffer has a max-buffer size that was allocated to a very large buffer capacity to be approximated as infinite. An indicator to the present buffer capacity, that is the buffer-size, is also provided to each buffer. A packet is admitted into its corresponding buffer, only if the buffer-size is less than the max-buffer size. According to the priority scheduling principle, non-preferred service can never be served as long as the buffer-size of the preferred service is higher than 0 i.e., the non preferred packets are not served as long as, there is even a single preferred packet to be served. Each service has its own departure indicator that is incremented every time its packet is served out of its queue. These departures are associated in the calculation of the average delay. Each service has its average (or mean) arrival rate and thus an

average inter arrival time. The inter arrival time at any moment is given by equation (1). The arrival time is given by the accumulation of those inter arrivals. For pure arbitrary conditions, the starting time for each service is given by a random variable, the starting time of the simulation will be determined as the arrival time of the first packet.

Each packet has its own arrival time attached to it, which is calculated as previously mentioned. This arrival time is used in the calculation of the delay of this packet where the total delay for a certain service is the accumulation of the delays for all packets leaving the queue of this service.

The mean service rate and hence, the mean time taken to complete the service of a packet, is the same for both services. The actual inter service time is dependent of the type of Markov queue. Two cases are considered. The first case is a special case of Markov queuing system with Poisson arrivals and constant inter service time. This case is called M/D/1 queue where D represents the deterministic behavior of servicing packets. The second case is the general one, M/M/1 queue, in which the inter service time is exponentially distributed and is given in equation (1), where the mean service time equals to the reciprocal of mean service rate.

A variable called Real-Time has been introduced to keep track of the actual time through the algorithm.

In the second module, the Real-Time variable is initiated at the beginning of the algorithm. Its initial value is the value of the arrival time of the first packet arrived, either preferred or non-preferred, to the system. The Real-Time is updated each time a packet leaves its queue by the value of the inter service time added to the present Real-Time.

In the third module, the first packet that has arrived to the system is scheduled in its corresponding queue then served out of the queue. The Real-Time is then updated. This is done after each time a packet leaves the queue.

In the forth module, which is done after each time the Real-Time is updated, is to generate packets whose arrival time is less than or equal to the recently updated Real-Time. These packets are then queued, each according to its serviee id.

The fifth module is to eheck the queue of the preferred service, if it is filled, even with a single paeket, it will be served first. If the preferred queue is empty, the non-preferred queue is served. If both queues are empty, then the algorithm will check to find out whieh service will have a packet to arrive first, and the Real-Time will be updated to this packet's arrival time. This will initiate the algorithm again as in the second step.

All steps, from the third step till the fifth, are repeated until the Real-Time reaches a predefined value that is the max-simulation-time, which has been determined before the simulation begins. If this is the case the simulation will be terminated.

Each time a packet leaves its queue, its delay is calculated as the difference between the Real-Time and its arrival time. The total delay of each service is updated step by step at the departure of each packet belonging to its queue. When the simulation is terminated, the average delay, mainly of the preferred service, will be obtained.

3. Implementation

This algorithm has been implemented by using Visual C++ and the Sim++ simulator. Computer simulation is highly interdisciplinary and simulation users and researchers can be found, for instance, in physics, industrial engineering, operations research and computer design. In spite of this diversity and global coverage, there are definite generic aspects of simulation that forms the core of simulation design. The core of the simulation design is composed of three areas: model design, model execution, and input and output analysis. The SimPack, which is an earlier version of Sim++, was constructed to permit experiments in the model design area.

The Simpack is a collection of C and C++ libraries supporting the creation of executable models for the computer design simulations. In this collection, several different simulation algorithms are supported including discrete event simulation which caters for performing the models of queuing systems.

SimPack was described by Dr.Paul Fishwick [Cubert,95], the prineiple auther, as follows:
The SimPack tool kit supports experimentation in computer simulation with variety of model types. The purpose of SimPack is to provide researchers and educators with starting point for simulating a system. The intention is that people will view what exists as a template "or a seed" and then grow a simulation program.

This section is devoted to the deseription of the implementation of our algorithm using some important classes of the Sim++. The deseription will be restricted to those classes that were used in the implementation of the algorithm.

3.1. Initializing the Model

A simulation rests on its list-processing capability. Naturally, much of what sim++ does depends on its list-processing capabilities. So it is good to have an understanding of how sim++ manages its lists. Vlist is a base class, its name means "virtual list". It has many derived classes. Presently, these derived classes are Linked, Heap, Leftist and Calender. Each derived class is a particular kind of structure. Methods of class Vlist are supported irrespective to of what particular derived class is chosen. These methods include: Insert, Remove and display. The only direct contact the application normally has with Vlist is to select a particular derived class for the future event list (FEL) at the start of simulation. Note that, nothing prevents an application from using any of the derived classes of Vlist. Something Danie Hay [Cubert,95] said in his README.2 which gives this conclusion: no matter what structure we choose for the FEL, we expect identical simulation results, with the only difference being the amount of time and the amount of space the computer needs to run the simulation. The linked structure is the type that has been selected to be used in our simulation. The FEL structure is determined when initializing the Sim++ at the

beginning of the simulation which is done by using *init_simpack(LINKED)*. It is mandatory to call *init_simpack()* before calling any other function related to the simulator. If this was not the case, an error will be generated accompanied by a message and the simulation will be terminated.

It should be noted that the elements of any Vlist are Events where an Event is a Token with some additional things wrapped around it. That means that class Events is a derived class of class base Token. A Token is the basic concept of the simulation. It can be considered as a simulation indefinable, in the same sense that length, mass, and time is indefinable in physies. In other words, a Token represents an individual unit of interest going trough the simulation. The individual airline passenger is represented in the airport simulation by a Token. Each process might be represented by a Token in a simulation of a computer system. In the queuing system simulation, a Token represents each packet going into the system that is the queue. In sim++, a Token is a collection of attributes and other things. The most important thing that comes with a Token is *id* where every Token needs an id. The API copies the information from the Token provided to it so it is possible to define only one Token structure and to use it as a buffer which the application code loads with whatever data is appropriate before calling sim++ API method.

3.2. Scheduling the Events

Scheduling means putting a future event in the future event list (FEL). The FEL is ordered on simulation time at whieh the future event will occur. So no matter in what order events are scheduled, or put in the FEL, they come out of the FEL in the ascending order of their occurrence times. Events with the same occurrence time are scheduled FIFO. Hence, there is a parameter that specifies a time interval that is the length of time into the future from now at which the event is to occur. It is very important to distinguish between a simulation time and time interval. The simulation time is the elapsed time since the start of simulation measured in arbitrary time unit. A time interval is a length of time in to the future measured with the same arbitrary time units as the simulation time. In *schedule()*, we specify the time interval. In the FEL, the time associated with each event is converted to and stored as a simulation time. The simulation time along with the time interval compose a very important part of our

implementation sense it helped to overcome a serious problem in this version of the simulator where this version was built to simulate only one queue.

3.3. The Simulation Clock

The simulation clock gives the elapsed time in arbitrary time units since the simulation has started. The clock starts at zero when the simulation is initiated and advanced as each scheduled event occurs. *Future::SimTime()* returns the present value the present simulation time in a variable that has been called checknum. By using this simulation clock, the future event list, that represents the simulated queue, has been dealt with as two different queues. The first queue has a simulation time=0 and its packets (events) are served first with FIFO order. While the second queue is scheduled so that its simulation time=1 and hence it is served when the first queue is empty. Notice that, the parameter associated with *schedule()* is the time interval and it is converted and stored in the FEL as the simulation time. Hence, before scheduling any event, the checknum is checked and the appropriate time interval is evaluated according to the service id of the packet.

3.4. Event Occurrence

Event occurrence causes the occurrence of whatever future event has the minimum simulation time, removing this event from the front of the future event list. There are three methods for event occurrence in Sim++:

- Fully automatic.
- Semi automatic.
- Manual method using *Future::NextEvent*.

The manual method has been used in the implementation of our algorithm for its simplicity .and in the same time gives us the upper hand in controlling the simulating steps. If all goes normal, the event_id in the Estatus object gets an event_id and Token info is copied into the Token in the Estatus object. A historical note: *NextEvent()* has a problem. If the FEL is empty when *NextEvent()* is called, *NextEvent()* returns to the application giving no indication of error. The above mentioned algorithm has avoided this error by introducing a variable that is called

FELsize. The FELsize is incremented each time a packet of each service enters its queue, and decremented each time a packet leaves its queue. Before the next_event() is called, the FELsize is checked. If the FELsize is zero, the algorithm generate a packet in a queue and update the algorithm before calling the *NextEvent()*.

There are two more functions that were used, one of them is *expntl(x)* which returns sample from a negative exponential distribution with mean x each time it is called. This function was used to produce equation (1). The second function is ranf() which returns a pseudo random variant with a uniform distribution between 0 and 1 and it is directly used for randomizing the starting point of the services.

4. Results and Discussion

As mentioned before, this study has two parts each part considers a different type of Markov queue. The main goal is to study the effect of the presence of non-preferred traffic on the preferred one. The metric of our concern is the average delay of the preferred service, while the resource being shared and allocated is the bandwidth that is the service rate.

In the first part of this study, the M/D/1 queue is considered. The inputs are the mean arrival rates of the preferred traffic, that are Varying from 0.1 packet /sec to 0.9 packet/sec. The inter arrival time at any moment is obtained from equation (1). The service rate is 1.8 packet/sec which gives a constant inter service time for the deterministic behavior of the M/D/1 queue. The buffer size of each service is infinite. The simulation time is 18,000 sec (5 hrs). This part has examined three different mean arrival rates of the non-preferred packets. In Fig (2), the non-preferred mean arrival rate is 0.3 packet/sec. As seen, the average delay of the preferred service increases as the preferred mean arrival rate is increased. In Fig (3), the mean arrival rate of the non-preferred service is 0.5 packet/sec, where Fig (4) shows the same relation between the preferred service mean arrival rate and its average delay, at the mean arrival rate of the non-preferred service of 0.7 packet/sec. The same conclusion has been reached, which is, the average delay of the preferred is increased as the arrival rate of the preferred service is increased. The three figures have been aggregated in

Fig (5) to show the effect of increasing the non-preferred arrival rate on the preferred service average delay.

In the second part of this study, the M/M/1 queue is considered. The inputs are Poisson's arrival traffic with mean arrival rate varying from 0.1 packet/sec to 0.9 packet/sec. The inter arrival times follow equation (1). The service model is exponential distribution with mean 1.8 packet/sec where the time taken to complete servicing one packet follow the equation (1). The buffer size of each service is infinite. The simulation time is 18,000 sec (5 hrs). This part has also examined three different mean arrival rates of the non-preferred packets. As shown in Fig (6), for non-preferred mean arrival rate of 0.3 packet/sec, the average delay of the preferred service increases as the preferred mean arrival rate is increased. In Fig (7), the mean arrival rate of the non-preferred service is 0.5 packet/sec, where Fig (8) shows the same relation between the preferred service mean arrival rate and its average delay, at the mean arrival rate of the non-preferred service is 0.7 packet/sec. The same conclusion has been reached, which is the average delay of the preferred increases as the arrival rate of the preferred service is increased. The last three figures have been aggregated Fig (9) to show the effect of increasing the non-preferred arrival rate on the preferred service average delay.

As indicated before, The theoretical result states that the average delay of the preferred service is not affected by the existence of any other service transiting the same node. From Fig (5) and (9), it is shown that, on the contrary to this theoretical result, the arrival rate of the non-preferred service does affect the average delay of the preferred service. As the arrival rate of the non-preferred service is increased, the average delay of the preferred service is also increased. This can be explained by observing the arrival times of the preferred and non-preferred service as well as their departures. It is true that, the non-preferred service is never been served as long as there is even a single packet in the preferred queue. However, if the preferred queue is empty, the queue of the non-preferred service is served. Meanwhile, if a packet of the preferred service has arrived, it will be queued until the non-preferred packet completes its service and this causes an extra delay on the preferred packet. As the

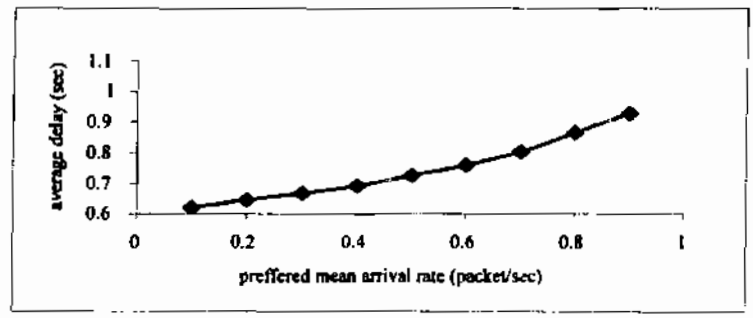

Fig (2): Relation between the average delay of preferred packets with varying the mean arrival rate, at non-preferred mean arrival rate of 0.3 packet/sec.

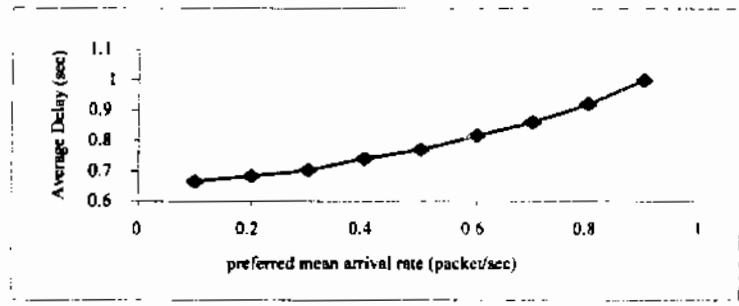

Fig (3): Relation between the average delay of preferred packets with varying the mean arrival rate, at non-preferred mean arrival rate of 0.5 packet/sec.

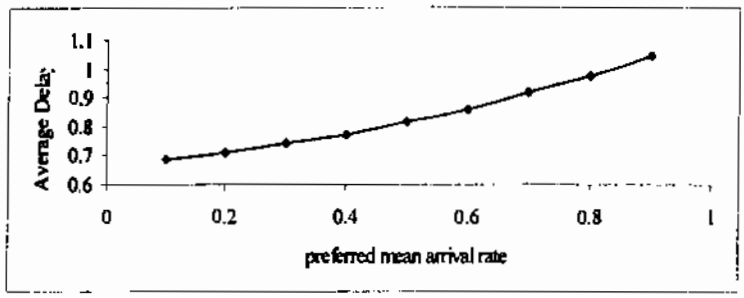

Fig (4): The relation between the average delay of preferred packets varying the mean arrival rate, at non-preferred mean arrival rate of 0.7 packet/sec.

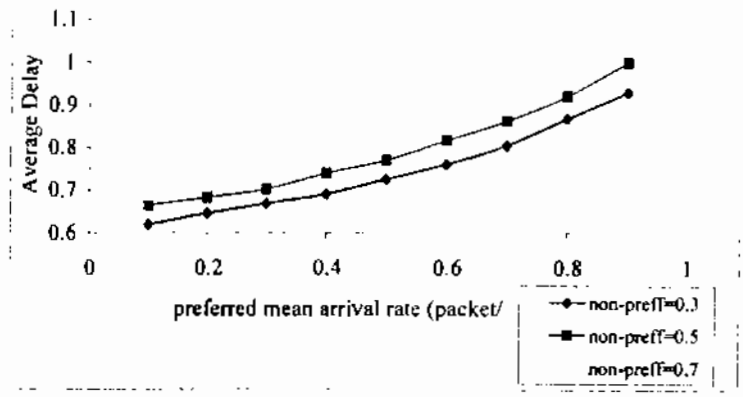

Fig (5): the relation between the average delay of the preferred packets due to the change in the preferred service mean arrival rate, at the non preferred mean arrival rate of 0.3, 0.5, 0.7 packet/sec

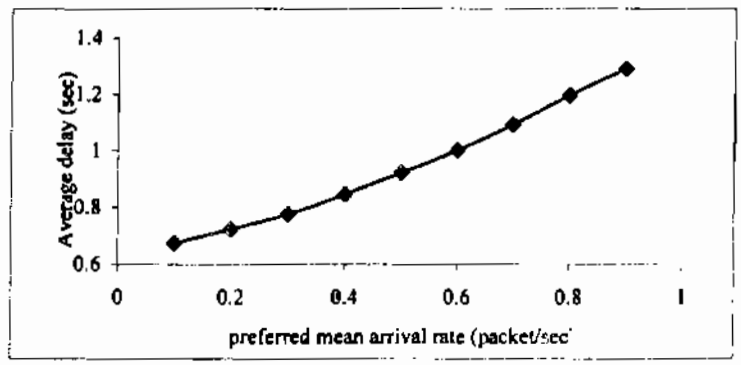

Fig (6): The relation between the average delay of preferred packets with varying the mean arrival rate, at non-preferred mean arrival rate=0.3 packet/sec

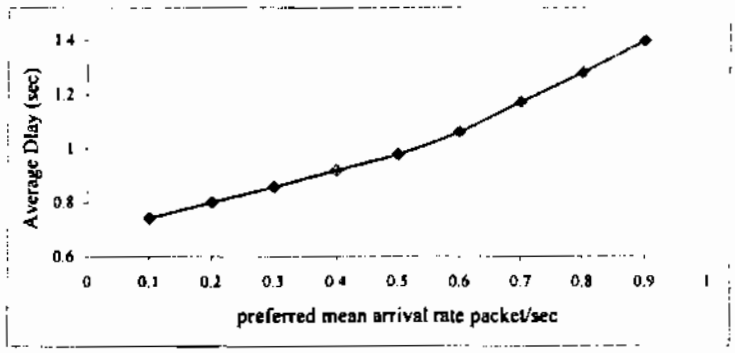

Fig (7): The relation between the average delay of preferred packets with varying the mean arrival rate at non preferred mean arrival rate=0.5 packet/sec

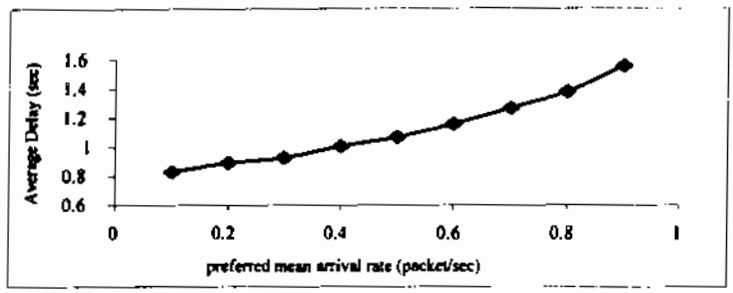

Fig (8): The relation between the average delay of preferred packets with varying the mean arrival rate, at non-preferred mean arrival rate of 0.7 packet/sec

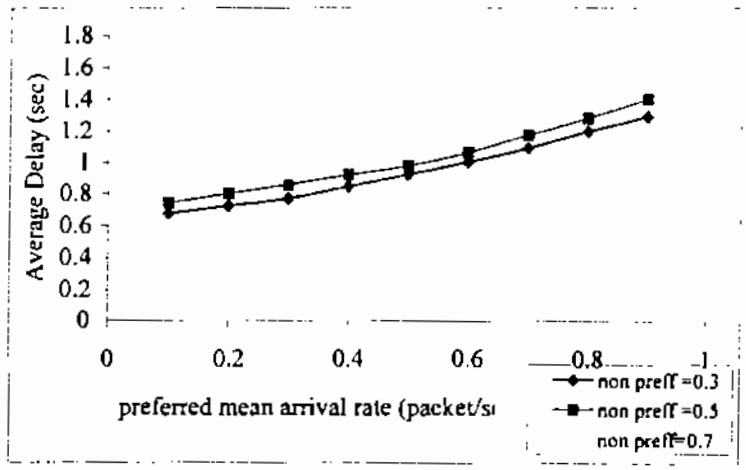

Fig (9): the relation between the average delay of the preferred packets due to the change in the preferred service mean arrival rate, at the non preferred mean arrival rate of 0.3, 0.5, 0.7 packet/sec

mean arrival rate of the non-preferred packet is increased, this case is repeated more often which leads to the increase of the average delay of the non-preferred packets.

It worth noting that, when the simulation time was increased up to 86,400 sec (24 hrs) the same results were obtained. Note that, when the service rate is decreased the average delay is increased, which is a natural result.

5. Conclusion

In this work, Sim++ simulator helped to investigate the performance of the premium service in a single node, using the priority scheduling mechanism. The obtained simulation results show that the average delay of the premium serviee increases as its mean arrival rate is increased. This result is in consistent with the theoretical result calculations. On the other hand, it was found that the average delay of the premium service is affected by the existence of another service transiting the same node. In the obtained simulation results, it was found that the average delay of the premium service increases as the mean arrival rate of the other service is increased. This finding is in contradiction with the theoretical calculations.

6. References

[Clark, 97] Clark, D., Wroclawski, J., An Approach to Service Allocation in the Internet, Internet Draft, June 97.

[Cubert,95] Cubert, R. M., Fishwick, P., SIM++ Version 1.0, July 95.

[Heinanen, 99] Heinanen, J., Baker, F., Weiss, W., Wroclawski, J., Assured Forwarding PHB Group, RFC 2597, June 99.

[Jacobson, 97] Jacobson, V., Differentiated Services Architecture, talk in the Int-Serv WG at Munich IETF, August 97

[Jacobson, 99] Jacobson, V., Nichols, K., Poduri, K., An Expedited Forwarding PHB, Internet Draft, June 99.

[Nichols, 99] Nichols, K., Jacobson, V., A Two-bit Differentiated Services Architecture for the Internet, Internet Draft, April 99.

[Sahu,99] Sahu, S., Towsley, D., Kurose, J., A Quantitative Study of Differentiated Services for the Internet, Proceedings of IEEE Global Internet'99.