ANALYSIS OF ORANGE ACTIVE QUEUE MANAGEMENT ALGORITHM TO FIND ITS OPTIMUM OPERATING PARAMETERS

Gökhan ÇATALKAYA[1], M. Kemal ŞİŞ[2]

[1]Department of Elt.&Eltr. Engineering, Dokuz Eylul University, Buca, Izmir, Turkey
[2]Department of Computer Science, DokuzEylul University, Buca, Izmir, Turkey
gcatalkaya@gmail.com

Abstract:*Due to the fast growth of the demand for the use of internet during the last decade, congestion control mechanisms to keep the throughput high and the queuing delays low get of vital importance. The purpose of this paper is to present a new approach, which is called as "Orange" in IP level congestion control as an active queue management mechanism and to compare its performance with that of the other mechanisms. Within the framework of this paper, the best operating point of Orange algorithm is evaluated by using the empirical formulas we derive. It is investigated that when the best operating point parameters are applied, Orange gives the best performance against other active queue management algorithms.*

Keywords:*Congestion control, IP level routing strategies, threshold, computer simulation, active queue management.*

1. Introduction

Communication networks have evolved significantly in the past few decades. Internet doubles its traffic every few months and more and more traffic involves increasing number of various flows. With this explosive growth over the past few years, network congestion phenomenon getsof vital importance. It is important to allocate the available resources effectively and fairly among a collection of competing users. Most networks provide a congestion control mechanism to deal with such a situation.

The basic goal of congestion control is to maximize the throughput of the link and minimize the average delay of packets in the network. In addition, it should also consider fair allocation of the resources among all the users.

TCP congestion control algorithm is the most widely used algorithm for congestion control. It detects congestion only after a packet has been dropped along the path. Increasing the queue capacities does not solve the congestion problem causing higher queuing delays. Queues should be generally kept as short as it is possible. Therefore, it is important to have mechanisms that keep throughput high but average queue sizes low.

2. Active Queue Management Schemes

Active Queue Management (AQM) schemes are IP level (gateway based) congestion control schemeswhere gateways notify the sources of incipient congestion. Active queue management schemes use a single FIFO (First In First Out) queue for all flows flowing through the router. The input to the control is the arrival rate and queue size for a particular outgoing link, and the output is a decision on how to mark or drop packets. It uses a certain algorithm to manage the length of the packet queue by dropping packets when necessary or appropriate. The responsive sources detect packet loss as a congestion indicator and react to these signals and adjust their sending rates. Unless the packets are dropped because of high queue sizes at the gateways, sources will keep increasing the sending rate causing longer delays in the network, which is not desirable. This kind of approach requires no state information and scales well.

The aim of AQM systems is to keep the average queue sizes at the gateways low. Keeping the queue sizes low has some advantages including,

- Provide queue space to absorb bursts of packet arrivals,
- Avoid lock-out and bias effects, from a few flows dominating queue space,
- Provide lower delays for interactive applications.
- Reduced packet loss rate.
- Reduced queuing delay and jitter.
- Improved throughput.

All AQM schemes detect impending queue buildup and notify the sources before the queues at the gateways overflows. AQM algorithms differ in the mechanism used to detect congestion and in the type of control method used to achieve a stable operating point for the queue size. Trying to keep the queue size stable at a desired level causes a tradeoff between link utilization and queuing delay.

Received On:17.12.2010
Accepted On:25.04.2011

A short queue reduces latency at the router but setting the target queue size too small may reduce link utilization by limiting the router's ability to buffer short bursts of arriving packets.

The way in which the congestion notification is delivered to the sources is the other important property of AQM schemes, which affects the performance. Two different alternatives are available to be used to notify the sources, namely Early Congestion Notification (ECN), and Random Early Detection (RED).

2.1.Early Congestion Notification (ECN)

Early Congestion Notification (ECN) [1] features end-to-end notification of network congestion without dropping packets. This feature is optional and only used when both of the endpoints signal that they want to use it. Unlike dropping packets to signal congestion in the traditional way in TCP/IP networks,after ECN is negotiated, itadds an explicit signaling mechanism by allocating bits in the IP and TCP headers of the packets flowing through the router in order to signal the beginning of congestion. The receiver side echoes back the congestion indication to the sender side and it reacts as if a packet drop were detected.In turn, the destination will transmit such information to the source piggybacking it into the acknowledgement message. Another way of speaking, gateways signal congestion to the sources by "marking" a packet (setting a bit in the header).

2.2.Random Early Detection (RED)

Second method is to drop the packets randomly with a probability when the queue sizes grow up in order to notify the sources about the incipient congestion. Floyd and Jacobson propose a mechanism called Random Early Detection (RED) [2]. RED makes a decision to drop a packet randomly when the queue average length ranges between a minimum and a maximum threshold. The probability of packet dropping is obtained from the average queue length accordingly to a linear law.

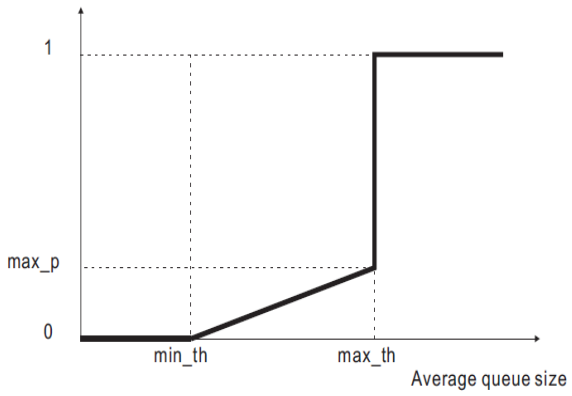

Figure 1.RED Algorithm**.**

For each packet arrival;
calculate the average queue size avg

The basic idea of RED algorithm is to keep the average queue size low (and hence end-to-end delay) while allowing occasional bursts of packets in the queue. Packet dropping probability is proportional to that connection's share of the throughput through the router. RED performs better than the drop tail algorithm because it has higher throughput and lower delays. It avoids global synchronization and has the ability to accommodate short bursts. It is easy to implement. It controls the average queue size even in the absence of non-adaptive sources. Because of its various advantages, in 1998, RED has been recommended as the standard of congestion avoidance mechanism in gateways.

RED algorithm (see Figure 1) calculates the average queue size by assigning different weights (the exponential weight factor, a user-configurable value) to old value and current measure. This means the adoption of a low pass filter to reduce the high frequency variation of the instantaneous queue. For high values of n, the previous average becomes more important. A large factor filters occasional bursts and keeps the queue length low. The average queue size is unlikely to change very quickly. RED algorithm will be slow to start dropping packets, but it may continue dropping packets for a time after the actual queue size has fallen below the minimum threshold. The slow moving average will accommodate temporary bursts in traffic. If the value of n gets too high, RED will not react to congestion. Packets will not be dropped by the RED algorithm. This would mean higher queuing delays.

if$\text{min}_{th} \leq \text{avg} < \text{max}_{th}$
calculate probability Pa
with probability Pa:
mark the arriving packet
else if$\text{max}_{th} \leq \text{avg}$
mark the arriving packet.

On the other hand, if the maximum threshold is set to a low value, the average queue size is easily affected from the current queue size. The resulting average may fluctuate with changes in the traffic levels. In this case, the RED process responds quickly to long queues. Once the queue falls below the minimum threshold, the process will stop dropping packets. If the value of n gets too low, RED will overreact to temporary traffic bursts and drop traffic unnecessarily. This would mean a bad usage of the link because of severe buffer oscillations. From these considerations, it is very difficult to find out the right trade-off, and it is hard

to tune RED to achieve both high link utilization and low delay and packet losses.

Although RED is a big success in internet congestion control, it still suffers from some problems. Dropping packets from flows in proportion to their bandwidth does not always lead to fair bandwidth sharing. For example, if two TCP connections unevenly share one link, dropping one packet periodically from the low speed flow will almost certainly prevent it from claiming its fair share, even if the faster flow experiences more packet drops. RED is designed to work with adaptive flows. Non-adaptive flows can take over the link's bandwidth. A non-adaptive connection can force RED to drop packets at a high rate from all connections. RED heavily penalizes TCP flows and awards non-TCP flows.

2.3.Other Active Queue Management Algorithms

In the last years, the active queue management policies have been object of a large interest in networking. Several proposals [3-4-5-6] have been presented to find more effective control policies than RED. REM and PI [7] are proposed to solve the problems, which RED faces. Their solution is very similar to each other. REM aims to achieve a high utilization of link capacity, scalability, negligible loss and delay. As an improvement to RED, REM algorithm differentiates between the congestion measure of each router and the dropping probability. REM algorithm maintains a so-called variable price, which eliminates the dependence of the dropping probability from the current value of the queue size. The REM algorithm uses the current queue size and the difference from a desired value to calculate the dropping probability accordingly to an exponential law. A source calculates the price of the whole path using the knowledge of the total number of packets dropped on the path. The main disadvantages of REM algorithm is that it gives no incentive to cooperative sources and a properly calculated and fixed value of price variable must be known globally.

In summary, internet routers should implement active queue management mechanisms to reduce average delay, to manage average queue length, to reduce packet dropping, and to avoid global synchronization. It is obvious that, current active queue management mechanisms have their own advantages as well as they have their own drawbacks.

3. Introducing Orange

By using the threshold type policy and the use of virtual drop server, we propose a new approach to drop or mark packets when the congestion will likely occur. We intend to use an IP level congestion control proposal, called Orange. Orange replaces RED as an active queue management algorithm to decide which packets are to be marked to indicate a congestion condition. The idea behind Orange is similar to RED which also uses "early dropping" concept to regulate the flows before congestion occurs. Here, "early" refers the fact that actually as long as there is space in the queue buffer to place the incoming packet; we still chose to drop them to warn TCP friendly sources (responsive or adaptive) against that possible congestion situation.

In a threshold queuing discipline, packets are preferably routed to the faster server. Packets are allowed to queue up while the slower server remains idle until the queue size reaches a certain "threshold" value, at which a point a packet is removed from the queue and sent to the slower server for service. The threshold value becomes critical control parameter affecting system's overall performance, and facilitating optimal system control. The primary performance parameter is the mean number of customers in the system, and accordingly the average waiting time per packet.

Optimization of the two heterogeneous servers problem is considered over an infinite time horizon with an average cost criterion. Although linear holding and service costs are considered, it is generally assumed that there is no additional cost incurred to turn on or to turn off a server.

Orange is based on the idea of dropping packets, randomly whenever some conditions are met, that is equivalent of using an alternate virtual server to the default link of that outgoing interface. Orange waits for a random amount of time after a dropping occurs before another one may be considered. This is the time equivalent of a service time sample of the "drop server". Orange proposal's main idea relies on a single queue, two server M/M/2 model analyses. In this model, first server is the link transmission element, and the second one is the unpreferred alternative link. The second one is used only when queue size exceeds a threshold.

Orange allows the incoming packet go to the queue for transmission if the queue size is below the threshold (Orange Limit). It drops the incoming packet and sets the timer if the timer is idle and the queue size is in between the Orange limit and the queue limit (maximum queue size). While the timer continues to be busy, Orange does not drop any incoming packet. Orange drops all incoming packets if the queue is full. One can refer to the Figure 2 for pseudo code of Orange algorithm.

```
If (queue limit) then
        Drop(packet)
Else
        If (Orange limit) then
                If (timer is idle) then
                        Drop(packet)
                        Begintimer()
                Else
                        Enque(packet)
                End if
        Else
                Enque(packet)
        End if
End if
```

Figure 2. Pseudo code of Orange algorithm.

The queuing model behind the Orange algorithm is mainly based on the M/M/2 queues. The M/M/2 case shown in Figure 3 is the simplest non-trivial case of a local model for a node in a network. In this type of network, for the traffic at the concerned node, there is only one final destination, but there are two different links by which the traffic can be carried toward the destination node. There may be several incoming links to the node; however, since all the traffic is destined to the same destination node it can all be stored in one queue. Arrivals to this queue are modeled as a Poisson arrival process (mean rate λ). Time spent on a link is modeled as exponentially distributed so links can be thought of as servers with exponentially distributed service times (mean rate μ). Therefore, the birth rate is always equal to λ, whereas the death rate depends on the state.

M/M/2 queue with a threshold is studied by [8], [9]. In their work, First passage time to an idle period (FPTIP) is studied. FPTIP value is derived as a function of μ_1, μ_2, λ. Here, we want to evaluate a formula for the average queue size and waiting time for the same system.

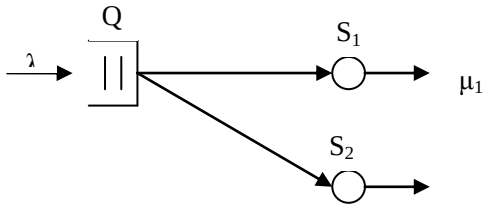

Figure 3.M/M/2 queue model.

The exact solution becomes cumbersome and is not efficient and necessary for M/M/2 system with a threshold. Moreover, the main contribution of this work is to find a direct relationship between the threshold value and the service time pair that give the minimum waiting delays per packet instead of finding an exact solution. For this aim, we consider to use the equations in Morrison's study [10] for derivation of our empirical formulas.

In his work, Morrison [10] finds an efficient solution of a threshold based queuing system with two heterogeneous servers and one queue. For the sake of simplicity, he considers a birth-death queuing system with two exponential servers with mean rates "μ", and Poisson arrivals with mean rate is "λ < 2μ", first in first out queuing discipline, unlimited buffer size of the bottleneck queue. Both servers are in use when the number of the customer in the system is more than a threshold level "c". Only one server is in use when the number of the customers in the system is less than "c + 1". Thus, the service rate of both servers is equal to each other; it is not important which server becomes idle. This system reduces to the generic M/M/2 case when "c" is one. So it is necessary to study the cases when" c > 1" for a non-trivial generalization.

The equilibrium probabilities of the number in the system are known by [11] and the mean waiting and sojourn times may be obtained from these by Little's formula. The system can be summarized as a single server system where the mean service rate is "μ" when there are less than "c + 1" customers in the system, and "2μ" when there are more than "c" customers are in the system. The difference between our preferred model and the Morrison's model is that in Morrison's model, second server with the same service rate is used when the number of customer in the system reaches to a certain threshold level whereas in our Orange's preferred model, second server with a service rate lower than the first server is used when the queue size reaches to a certain threshold level as long as the second alternate server is idle.

From Morrison's study, we easily state that, the equilibrium probability P_0 that there is no customer in the system is

(1)

The equilibrium probability P_i that there are "i" customers in the system is

$$P_i = \begin{cases} P_0 \left(\lambda/\mu\right)^i & for\ 0 \le i \le c \\ P_0 \left(\lambda/\mu\right)^c \left(\lambda/2\mu\right)^{i-c} & for\ i \ge c \end{cases} \qquad (2)$$

The mean waiting and sojourn times are given by

$$\langle W \rangle = P_0 \left(\frac{\lambda}{\mu}\right)^c \left\{ \begin{array}{l} \mu \frac{\square}{(\mu-\lambda)^2}\left[\left(\mu\frac{\square}{\lambda}\right)^{c-1} - 1\right] + \\ (c-1)\left[\frac{1}{(2\mu-\lambda)} - \frac{1}{(\mu-\lambda)}\right] + \frac{\lambda}{(2\mu-\lambda)^2} \end{array} \right\} \quad for\ \lambda \neq \mu \tag{3}$$

$$\langle W \rangle = \frac{P_0}{2\lambda} c(c+1) \qquad for\ \lambda = \mu \tag{4}$$

$$\langle T \rangle = P_0 \left(\frac{\lambda}{\mu}\right)^c \left\{ \begin{array}{l} \mu \frac{\square}{(\mu-\lambda)^2}\left[\left(\mu\frac{\square}{\lambda}\right)^{c} - 1\right] + \\ (c)\left[\frac{1}{(2\mu-\lambda)} - \frac{1}{(\mu-\lambda)}\right] + \frac{2\mu}{(2\mu-\lambda)^2} \end{array} \right\} \quad for\ \lambda \neq \mu \tag{5}$$

$$\langle T \rangle = \frac{P_0}{2\lambda(c^2+3c+4)} \qquad for\ \lambda = \mu \tag{6}$$

We can adapt the Morrison's results into our case by substituting "$2\mu = \mu_1 + \mu_2$", "$\mu = \mu_1$", and and "$c = K + 1$" in the above equations. Therefore, the mean waiting and sojourn times of M/M/2 with a threshold K case are found as,

$$\langle W \rangle = P_0 \left(\frac{\lambda}{\mu_1}\right)^{K+1} \left\{ \frac{\mu_1}{(\mu_1-\lambda)^2}\left[\left(\mu_1\frac{\square}{\lambda}\right)^{K} - 1\right] + (K)\left[\frac{1}{((\mu_1+\mu_2)-\lambda)} - \frac{1}{(\mu_1-\lambda)}\right] + \frac{\lambda}{((\mu_1+\mu_2)-\lambda)^2} \right\} \quad for\ \lambda \neq \mu_1 \tag{7}$$

$$\langle W \rangle = \frac{P_0}{2\lambda(K+1)(K+2)} \qquad for\ \lambda = \mu_1 \tag{8}$$

$$\langle T \rangle = P_0 \left(\frac{\lambda}{\mu_1}\right)^{K+1} \left\{ \frac{\mu_1}{(\mu_1-\lambda)^2}\left[\left(\mu_1\frac{\square}{\lambda}\right)^{K+1} - 1\right] + (K+1)\left[\frac{1}{((\mu_1+\mu_2)-\lambda)} - \frac{1}{(\mu_1-\lambda)}\right] + \frac{(\mu_1+\mu_2)}{((\mu_1+\mu_2)-\lambda)^2} \right\} \quad for\ \lambda \neq \mu_1 \tag{9}$$

$$\langle T \rangle = \frac{P_0}{2\lambda\left((K+1)^2+3(K+1)+4\right)} \qquad for\ \lambda = \mu_1 \tag{10}$$

Where (11)

Those derived formulas are used to justify the simulation results. The equations are valid in a system where the Poisson arrivals and exponentially distributed service rates are applied. However, most of the flows in today's networking world consist of responsive flows like TCP, which adjust their sending rate according to the congestion indications from the network. Thereforememoryless arrival process cannot be a realistic assumption.

In systems, which have the memoryless property, the time distribution until the next event is the same regardless of how much time has passed since the last event, and the average time until the next event is the same as the average interevent time. This property is also a direct consequence of the complete randomness of the Poisson process; what happens in the current interval is independent of what has happened in the previous interval.

The main goal of active queue management algorithms is to warn TCP friendly sources about the incoming congestion situation so that they will be able to reduce their sending rate to prevent the network to get in congestion collapse. The main objection of our proposed algorithm is to provide better conditions (high throughput and low per packet delays) for the networks where not only the constant bit rate sources but also the responsive sources are available. Therefore, it is meaningful to provide practically an empirical formula to determine the best operating point for Orange having its system parameters (threshold and service time) tuned for such conditions.

4. Testing M/M/2 Equations With Simulation

In order to test our algorithm's performance, we simulate the topology in Figure 4 for different queue types (DropTail, Orange, and RED) using the NS-2 network simulator. When Orange is applied, the queue model can be considered as an "M/M/2" queue with a threshold. It means that the faster server, which is the primary server at the output link of the queue, remains the same whereas the virtual drop server appears when the number of the packets at the buffer of the queue exceeds a threshold level.

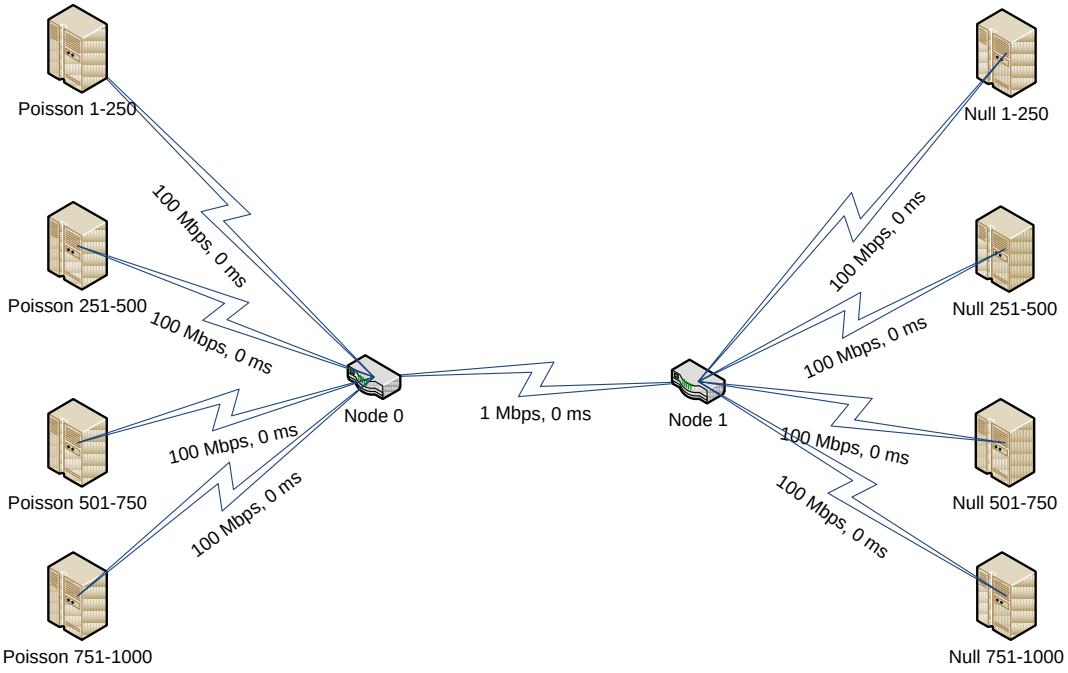

Figure 4. Simulation topology for M/M/2 queue.

Poisson sources in NS generate packets of constant size that we have to set in the beginning of the simulation. In order to overcome this lack of NS, we make a large number of Poisson sources involve in the simulation by setting the packet size of each traffic source is different and determined by random number generator that generates exponentially distributed packet sizes with an average value. Aggregating Poisson sources in this way generates a traffic source with exponentially distributed sending rates, and packet sizes with a mean value, which is assumed to be in the mathematical analysis. Note that; we have four different nodes in simulation topology. Because, in NS, the total number of agents we can connect to a node is 256, so we have to connect the 1000 sources and destinations to 4 different nodes where each node has 250 different sources.

In this set of experiments, our aim is to compare the simulation results with the calculations from equations which we have already derived in previous section.Remember that, in NS application, the maximum threshold value of the RED algorithm is three times of its minimum threshold parameter unless specifically specified. The other parameters for RED are kept the same as NS's default parameters. Different values of the minimum threshold of both RED and Orange can be applied upon our request. Orange timer (the service time of the unpreferred alternate link) of the bottleneck queue is given in milliseconds and this value is directly proportional of the capacity of the link at the output of the queue of virtual drop server.

In Orange, while the packet, which takes service from virtual drop server, is being dropped, the virtual drop server will not consider to drop another packet. For example, 8 ms service time corresponds to 1 Mbps bandwidth. It means that, if the link is fully utilized, 125 packets will take service per second. In other words, the service time of the virtual drop server is 8 ms/packet.

We want to aggregate traffic to generate a total arrival rate "λ = 1200 packets/s", and average packet size equal to 100 bytes by aggregating 1000 Poisson sources. The packet size of each source is set by sampling an exponential random variable of average 100 bytes. Each link capacity between the source and the bottleneck node is 100 Mbps with zero link delays. Bottleneck link capacity is Mbps and zero delays. Thus, the average service rate of the link can be computed for 100 bytes packets as 1250 packets/s. Minimum threshold value of Orange is 10 packets and service time of the virtual drop server is 5 ms, which corresponds to a service

rate of 200 packets per second. It is assumed that there would be a virtual link at the output buffer of the server with a capacity of 200 Kbps.

Using the above parameters, we can calculate the results from the Morrison's equations. Remember that Morrison finds the mean waiting time "W" as;

$$(W) = P_0 \left(\frac{\lambda}{\mu_1}\right)^c \left\{ \begin{array}{l} \dfrac{\mu_1}{(\mu_1 - \lambda)^2}\left[\left(\dfrac{\mu_1}{\lambda}\right)^{c-1} - 1\right] + \\ (c-1)\left[\dfrac{1}{((\mu_1 + \mu_2) - \lambda)} - \dfrac{1}{(\mu_1 - \lambda)}\right] + \dfrac{\lambda}{((\mu_1 + \mu_2) - \lambda)^2} \end{array} \right\} \; for\ \lambda \neq \mu_1 \quad (12)$$

This waiting time can be compared with the average queue size in our simulations with a calculation by using the Little's formula. Average queue size (q) is then computed by "W = q / λ ". Using the equation (12), waiting time is computed as 0.006989, and correspondingly using the Little's formula, the average queue size is computed as 8.38 packets.

Simulation results for 1200 packets per second of arrival rate, minimum threshold value of 10 packets of minimum threshold and 5 ms of service time of the virtual drop server can be found in Table 1. Average queue size is obtained by simulation as 9.04 packets, which matches to the computed value of average queue size 8.38 packets. The difference between the simulation results and the computed values are negligible so that we are able to state that our derived formulas give the correct results and match to the simulation results.

Table 1. Simulation results of M/M/2 queue with a threshold.

Queue Type	MinTh	Orange Timer (ms)	Sent Packets	Arrival to Router	Arrival to Destination	Average Delay ms	Average Queue Size
Orange	14	5	13178	12680	11728	8.49	9.04

5. Orange's Performance Analysis

To appreciate Active Queue Management application, we must consider congestion, and responsive flows like TCP. Otherwise effect of using AQM algorithms like RED or Orange over Drop Tail may not be recognized. "For offered loads up to 80% of bottleneck link capacity, no AQM scheme provides better response times than simple drop-tail FIFO queue management"[12].In the practical cases, most of the traffic is formed by the responsive sources of large amounts. Those are the flows of surfing a web site, or downloading a file from the internet. It is more complicated to control those flows.

In order to test our algorithm's performance in a more realistic environment, we use a sample topology consisting of heterogeneous TCP flows whose link delays are varying. This topology has also been studied by Kinicki and Zheng[13]. They use this topology to test their own algorithm's performance with that of RED algorithm. They claim that the chosen RED parameters in their work give the best result when RED algorithm is applied. To test our algorithm with other IP level congestion control methods, this topology which has many heterogeneous TCP Reno flows is best suited for our performance comparisons of our proposed algorithm. We run a series of simulation experiments using the NS simulator to compare the performance of Orange with RED and its variants with heterogeneous TCP Reno sources.

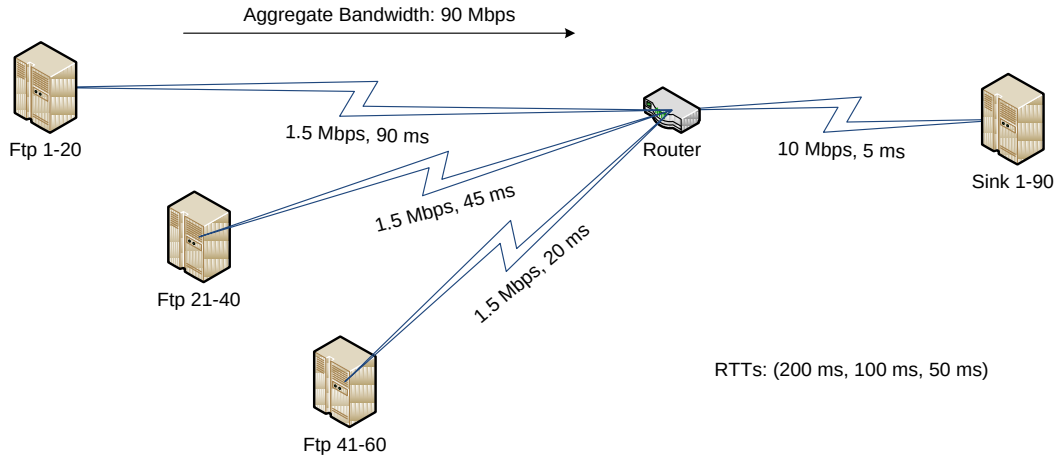

Figure 5. Simulation topology for a more realistic environment.

The simulated network topology in Figure 5 consists of one router, one sink and a number of simulated FTP sources. All flows are divided into three flow groups (fragile, average, and robust) based on the instantaneous round trip time of each flow. The mentioned Orange router maintains a single flow queue for each flow group, which is a FIFO queue that stores a pointer to a packet in the router queue for each packet. The aim of this topology is to establish a real network environment, which has many flows with too many RTT's.

Each FTP source feeds 1000-byte packets into a single congested link attached to the router. The TCP ACK packets are 40 bytes long and each source has a window size of 64 packets. The capacity of the botleneck link is 10 Mbps with a 5 ms delay to the sink. When the demand is kept constant, the number of the flows that generates the demand has a negative effect on performance. We choose one-way link delays for the fragile, average and robust sources of 95 ms, 45 ms and 20 ms respectively. Thus, the fragile, average and robust flows have round trip times of 200 ms, 100 ms and 50 ms when there is no queuing delay at the router. The router queue size was fixed at 120 packets based on published rules of thumb for accommodating the network bandwidth delay product. All simulations for this study run for 100 simulated seconds and include an equal number of fragile, average and robust TCP flows. Half of the flows in each flow group start at time zero the second half start at time 2 seconds. For example, for a 60-flow simulation, 10 fragile, 10 average and 10 robust flows start at time 0, and the remaining 30 flows start at 2 seconds. The first 20 seconds of simulated time are not considered to reduce the startup and transient effects. The sum of the capacities of all the incoming flows is held constant at 90 Mbps for all simulations in this study regardless of the number of flows. Thus when the number of flows are increased the individual link capacities are proportionally decreased. Unless specifically specified the values for RED parameters of minth and maxth are set in such a way that maxth is three times of the minth.

Our aim by using Orange algorithm is to keep the aggregate throughput high but the average packet delay and the average queue sizes low. We have four sets of simulation; each has a different minimum threshold value (10, 15, 20, and 25). In RED, when the average queue size exceeds the minimum threshold value, queue starts to drop the incoming packets according with a dropping probability value based on the calculation of the maximum dropping probability and the value of the average queue size. In Orange, when the current queue size exceeds the minimum threshold value, queue starts to drop the incoming packets according the busy – idle status of the alternate drop server. RED has a maximum threshold value parameter to drop all the incoming packets when the average queue size exceeds it. Maximum threshold of the RED is three times of its minimum threshold parameter and unless it is specifically specified.

On the other hand, Orange has no maximum threshold parameter, but it has the parameter, which is the service time of the alternate server. It is the busy period between the time that the Orange drops a packets and the time that the Orange queue will consider another packet to drop (busy time for dropping a packet). This time value (Orange Timer) is not constant, it is exponentially distributed about a mean average value, which is parameter of Orange queue type. We have simulated our sample topology with the values of the service time of the alternate server from values of 1 second to 10000 seconds in order to test the effect of the service time to the Orange's overall performance. As the Orange's service time goes to infinity, its operating behavior approaches the drop tail. With a big service time, Orange drops a packet and after this time, it never drops any packets because its alternate drop server is busy during the simulation time. The results of the experiments are given in Table2. Detailed simulation results can be found in [14].

The performance parameters that we have compared RED and Orange are the average throughput (Kbps), average delay (ms), average queue size. It is obvious that in most of the regions, Orange has better performance compared to RED and Drop Tail especially when the service time of the alternate server is around from 4 ms to 7 ms. Orange provides better performance for smaller timer values as the minimum threshold value increases.

In Table 2, when the threshold is 10, RED's throughput is measured as 9610 Kbps, average delay, and average queue are measured as 79.18, and 23.82, respectively. In this set of experiments, Orange's minimum threshold value is fixed at 10 packets. It means that Orange starts to drop the incoming packets when the queue size exceeds 10 packets. Orange's service time is adjusted from low values to the high values. When it gets higher, Orange approaches to work like a DropTail queue. Orange drops the packet, and it never gets idle because the service time for that packet is too high to consider another packet to drop or not. While keeping the threshold at a fixed level which is 10 for this set of simulations, total throughput increases, as the service time increases whereas average delay, and average queue size decrease. Orange gives better results than RED when Orange's timer is adjusted around 6.5, 7 ms.This is the point where Orange provides higher throughput values and lower delay values than that of RED. It is obvious that average delay is directly proportional to average queue size. It increases as the average queue size increases.

When the threshold is 15, RED's throughput is measured as 9615 Kbps, average delay, and average queue are measured as 88.26, and 33.83, respectively when RED's minimum threshold value, and maximum threshold value are fixed at 15, 45 respectively. RED's throughput, average delay, and average queue are measured as 9606, 80.65, 25.26, respectively when RED's minimum threshold value, and maximum threshold value are fixed at 15, 30 respectively. In this set of experiments, Orange's minimum threshold value is fixed at 15 packets. As we know, RED starts to consider dropping packets when the average queue size exceeds its minimum threshold value. Therefore, as we expect, the average delay and average queue size are more than the previous results. Orange gives better results -higher throughput and lower delay- than RED when Orange's timer is adjusted around 5, 6 ms.

When the threshold is 20, RED's throughput is measured as 9615 Kbps, average delay, and average queue are measured as 96.67, and 43.31, respectively when RED's minimum threshold value, and maximum threshold value are fixed at 20, 60 respectively. RED's throughput, average delay, and average queue are measured as 9591, 82.40, 26.98, respectively when RED's minimum threshold value, and maximum threshold value are fixed at 20, 30 respectively. Orange's minimum threshold value is fixed at 20 packets. Orange gives better results -higher throughput and lower delay- than RED when Orange's timer is adjusted around 4.5 ms.

Table 2. Simulation results

Orange Timer (ms)	When Min_{Th}= 10			When Min_{Th} = 15			When Min_{Th} = 20			When Min_{Th} = 25		
	Avg. T.Put	Avg. Delay	Avg. Q.Size	Avg. T.Put	Avg. Delay	Avg. Q.Size	Avg. T.Put	Avg. Delay	Avg. Q.Size	Avg. T.Put	Avg. Delay	Avg. Q.Size
RED-75	-	-	-	-	-	-	-	-	-	9615.54	103.78	52.56
RED-60	-	-	-	-	-	-	9615.53	96.67	43.32	-	-	-
RED-45	-	-	-	9615.48	88.27	33.83	-	-	-	-	-	-
RED-30	9610.84	79.18	23.82	9606.00	80.66	25.26	9591.64	82.41	26.99	9482.20	83.27	28.80
1	9493.04	62.30	7.00	9546.53	65.65	11.23	9560.86	69.41	15.02	9583.09	72.12	19.23
2	9516.54	65.09	7.99	9552.46	70.38	11.97	9561.77	74.39	16.12	9592.58	77.26	20.23
3	9522.86	68.87	9.35	9661.41	73.51	13.56	9585.87	78.04	17.99	9599.21	81.59	22.24
3.5	-	-	-	-	-	-	-	-	-	9615.48	78.84	24.36
4	9591.63	70.94	11.19	9603.82	74.87	16.43	9611.11	78.83	20.83	9612.87	82.87	25.60
4.5	-	-	-	-	-	-	9607.86	81.51	21.81	9610.27	84.07	25.62
5	9597.22	69.56	14.59	9610.26	73.04	18.98	9611.19	77.17	23.95	9615.27	81.30	28.38
5.5	-	-	-	-	-	-	9612.82	82.74	23.57	9614.22	86.47	27.58
6	9606.77	74.24	17.06	9613.37	77.51	21.00	9615.28	81.98	26.22	9615.44	85.95	31.01
6.5	9612.53	77.86	18.71	9613.41	81.04	22.37	9614.90	85.78	27.43	9615.54	89.07	31.65
7	9614.52	84.02	23.24	9615.63	88.56	28.26	9615.35	89.93	30.75	9615.44	93.13	34.45

8	9615.43	119.81	63.99	9615.53	119.08	93.88	9615.53	118.80	62.90	9615.63	121.48	66.15
9	9615.44	97.18	39.18	9615.64	97.96	40.56	9615.54	99.91	42.76	9615.54	99.75	42.92
10	9615.53	105.42	54.74	9615.44	105.41	54.89	9615.53	105.76	54.73	9615.53	105.02	54.56
50	9615.53	146.67	110.95	9615.43	146.65	110.71	9615.73	146.35	110.72	9615.53	146.35	110.79
100	9615.52	146.87	112.44	9615.53	145.73	111.26	9615.53	146.56	112.02	9615.45	146.81	112.51

When the threshold is 25, RED's throughput is measured as 9615 Kbps, average delay, and average queue are measured as 103.77, and 52.56, respectively when RED's minimum threshold value, and maximum threshold value are fixed at 25, 75 respectively. RED's throughput, average delay, and average queue are measured as 9482, 83.27, 28.80, respectively when RED's minimum threshold value, and maximum threshold value are fixed at 25, 30 respectively. Orange's minimum threshold value is fixed at 25 packets. Orange gives better results -higher throughput and lower delay- than RED when Orange's timer is adjusted around 4 ms.

When we try to track the change the change in Orange's timer optimum value as compared to the change in the set threshold value, we can fit an inverse proportional relation to the square root of threshold (K). For instance, if we compare the simulation results where the threshold value is 10 with the results where the threshold value is 25, service time of the alternate server should be multiplied by $\sqrt{\frac{10}{25}} = \sqrt{0.4} \cong 0.632$.

Thus, to get the optimum value of the alternate server's average service time, if we multiply best service time value where the threshold is 10 with this coefficient, we can easily see that the result fits very well with the result where the threshold is 25. (6.5 ms * 0.632 = 4.10 ms). This last value is the best service time value of the alternate server where the threshold is 25.

Consequently, empirically fitting relationship can be formulated as

$$Service\ Time \propto \frac{1}{\sqrt{K}} \tag{13}$$

where "K" is Orange's threshold value for the best performance of our simulation.

Hence, we can state that, from the analysis of the simulation, empirical results suggests with our used simulation parameters are

$$Service\ Time = \frac{20}{\sqrt{K}} \tag{14}$$

inmiliseconds. A comparison between the simulation results and this empirical formula is given in Table 3.We can easily see that the results fit well.

Table 3.A comparison between simulation results and empirical formula.

Threshold Applied	Orange's Timer in ms	
	Best Result from Simulation	Calculation from Empirical Formula - Equation (14)
10	6.50	6.32
15	6.00	5.16
20	4.50	4.47
25	4.00	4.00

6. Interpretation Of Empirical Formula For Determining Orange's System Parameters

We have made our experiments for different threshold values and different service times for slower server in order to find the best operating point of our algorithm in a congested network environment, which includes responsive flows. Our aim is to find a relation between the values of the threshold and the service time of the slower server at the operating point from the experiments and the mathematical analysis. Şiş [8] studied the optimum threshold value of an M/M/2 queue where Poisson arrivals, and exponentially distributed service times are of interest (when the service rates of both servers are predetermined). He proved that the first order approximate value of optimum threshold, is the largest non-negative integer which satisfies (if there is no such non-negative integer, it is zero)

$$\tilde{K} \le \frac{\mu_1 - \lambda}{\mu_2} - 1 \tag{15}$$

This approximate value for the optimum value of the threshold gives satisfactory result under the assumption that "μ_1" is considerably greater than

"μ_2" and "$\mu_1 >> \lambda$". Here, the results are approached as there is a continuous flow of traffic arriving to the queue with the average rate of "λ" units/time and similarly μ_i units/time is the continuous average out-flow through link i. Therefore, the results are valid in the systems where memoryless sources like Possion sources are applied. If "μ_1" is not considerably larger than "μ_2", it is clear that threshold is nearly zero. When "μ_1" is considerably larger than "μ_2", if "$\mu_1 >> \lambda$", for optimum threshold we can use the approximate value in Equation 15. The only remaining case is the case where "$\mu_1 >> \mu_2$", but "$(\mu_1-\lambda) \approx 0$". There, actually a non-zero threshold value occurs which is not anticipated in our approximation. Although this afore mentioned analysis can be made to find the expected delay value to relate it with the threshold value, this would be restricted to the case where Poisson arrivals and exponentially distributed service times are involved.

In order to test our algorithm's performance in a network where the responsive flows are dominant, we use the responsive sources (ftp sources) in our simulation. Responsive sources probe the available bandwidth in the network, and they adjust their sending rate as long as there is no packet loss. Arrival rate will be almost the same as the service rate of the server. We can easily say that, in our experiment "$(\mu_1-\lambda) \approx 0$". We need to find an equation for this case in terms of μ_1, μ_2, λ, and K under these circumstances where responsive flows are involved.

Padhye and his friends [15] develop a simple analytic characterization of the steady state throughput of a bulk transfer TCP flow (i.e., a flow with a large amount of data to send, such as FTP transfers) as a function of loss rate and round trip time. Their model captures not only the behavior of TCP's fast retransmit mechanism but also the effect of TCP's timeout mechanism on throughput.

In their work, N_t represents the number of packets transmitted in the interval [0,t] and "B_t (N_t/t)" represents the throughput on that interval. Thus, B_t represents the throughput of the connection, rather than its goodput. They define the long-term steady-state TCP throughput B to

$$B = \lim_{t\to\infty} B_t = \lim_{t\to\infty} \frac{N_t}{t} \qquad (16)$$

They have assumed that if a packet is lost in a round, all remaining packets transmitted until the end of the round are also lost. Therefore they define *p* to be the probability that a packet is lost, given that either it is the first packet in its round or the preceding packet in its round is not lost. They are interested in establishing a relationship B(*p*) between the throughput of the TCP connection and the loss probability (*p*).

In their work, when timeout occurrences are ignored, B(*p*) is derived to be;

$$B(p) \approx \frac{1}{RTT}\sqrt{\frac{3}{2bp}} \qquad (17)$$

where "*b*" is the number of packets acknowledged by a received ACK. In many TCP implementations, "*b* = 2". When timeouts are taken into account, they derive the B(*p*) as;

$$B(p) \approx \frac{1}{RTT\sqrt{\frac{2bp}{3}} + T_0 min\left(1, 3\sqrt{\frac{3bp}{8}}\right) p(1+32p^2)} \qquad (18)$$

By this formula, we can easily observe that TCP favors the flows with short RTT. It means that when downloading a file from a closer server, the download performances will be better. We can observe that the relationship between loss rate *p* and throughput is not linear but an inverse square root relation! It means when *p* is increased 4 times, throughput drops to half.

As we have already shown, while the service time of the drop server increases, the optimum value of the threshold decreases in order to achieve the best operating point. If the service time of the drop server were too low, the threshold would be high enough to prevent unnecessary packet drops. If the threshold were too low, we need high values of the service time of the drop server to make the drop server idle after dropping a packet. To use the drop server for enough times, it must work faster. Therefore, we can easily say that the optimum value of the threshold is inversely proportional to the service time of the drop server. We have found empirically a relation like;

$$Service\ Time = \frac{1}{\mu_2} \propto \frac{1}{\sqrt{K}} \qquad (19)$$

On the other hand, according to [15], we can state that TCP's throughput is inversely proportional with the square root of dropping probability (P). We can also intuitively claim that the dropping probability is inversely proportional with the threshold (K):

$$p \propto \frac{1}{K} \qquad (20)$$

If we think alternate server as a real server, departures from it contributes to the total throughput. Therefore, the service rate of the alternate server and the throughput can be assumed that they are directly proportional.

(21)

If we use the last two formulas in Padhye's simple throughput equation, then we get

$$B(p) \propto \frac{1}{RTT}\sqrt{\frac{3}{2bp}} \approx \frac{1}{RTT}\sqrt{\frac{3K}{2b}} \quad (22)$$

In general TCP implementations, b value is fixed as 2 and we can assumed that the RTT is constant during simulation so without specifying proportionality constants, we can end up with

(23)

or, alternate server's service time is inversely proportional with square root of threshold (K):

$$Orange's Timer = \frac{1}{\mu_2} \propto \frac{1}{\sqrt{K}} \quad (24)$$

We can just conclude that, empirically the best value of the service time is inversely proportional with the square root of the threshold value applied as we have already stated in Equation 13.

Furthermore, if we can estimate as "a rule of thumb", the best value of μ_2 departing from the capacity of the main (bottleneck) link, it can be estimated in the order of "1/10" to "1/5" of the capacity of the main link. By using our empirical formula, we can try to find an optimal threshold (K) value. This finishes selecting Orange parameters that gives the best operating point. This parameter estimation procedure is much simpler, effective, and more meaningful than the tuning the complex RED parameters.

The other way around, if we take the threshold value of Orange departing from the minimum threshold value proposed for RED implementers, we can easily calculate the best optimum value of Orange's timer for the best performance of active queue management.

Explanation of this relates the drop server to behave like a TCP friendly source. The implication of this can be very meaningful. Mentioning TCP friendliness in general means reacting to congestion in the same way as TCP, considering only Triple Duplicate (TD) packet loss occurrences that result in TD, this would mean to be conformant with the throughput equation (16). We have demonstrated that our empirical result is in accordance with the equation (16), therefore suggesting the TCP friendliness for the best operating conditions. However, keeping in mind that alternate server's output is, in return as retransmission, a load for the original sender (TCP source), they will be part of the offered load, hence throughput is in relation with alternate server's link capacity or service time.

7. Conclusions

The main contribution of this work is to present an IP level congestion control mechanism to control the performance of a traffic network at the node level. In this work, a new active queue management algorithm called Orange is designed and evaluated. The main idea behind Orange comes from the analysis of two heterogeneous servers and one queue with a threshold-based queuing system in order to achieve both higher throughput and lower queuing delays. By using the threshold type policy and the use of virtual drop server, we have proposed a new approach to drop or mark packets when the congestion will likely occur. The primary performance parameter is the mean number of customers in the system, and accordingly the average waiting time per packet as well as the throughput of the network.

In addition, we consider finding out an empirical relationship between the system parameters of our algorithm using the mathematical analysis. Simulation results are used to tune up the empirical formulation. By achieving this aim, we consider to use a virtual drop server to drop the incoming packets when the actual queue size exceeds a threshold level. The only adjustable parameter based on the changing conditions of the network is the service time of the virtual drop server. Since for many applications, this service time is not usable, we consider it an important and distinguishing characteristic of our work.

Moreover, we provide an efficient solution of a threshold based queuing system with two heterogeneous servers and one queue.

This study confirms that generally Orange performs better than RED due to the fact from simulations that it results in higher throughput values and lower queuing delays (thus the lower mean waiting times per packet) for the networks with heterogeneous flows. Orange simulations indicate that Orange requires less parameter settings than RED.

We can propose that Orange replaces RED as an active queue management algorithm to decide which packets are to be marked to indicate a congestion condition for the current Internet routers. We still chose to drop them to warn TCP or TCP friendly sources (responsive or adaptive) against that possible congestion situation. While doing so, we tune Orange parameters such that Orange's drop server acts like a TCP friendly source as depicted in (24). Since dropped TCP packets by the virtual drop server will be re-sent as offered load to the system. Drop server can be considered as a virtual source to the network.

8. References

[1] K. Ramakrishnan, S. Floyd, "A Proposal to add Explicit Congestion Notification (ECN) to IP", RFC 2481, Proposed Standard, 1999. Retrived May 2009 from http://www.rfc-editor.org/info/rfc248.
[2] S.Floyd, V.Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, vol: l, no: 4, pp. 397-413, 1993.
[3] W.Feng, D. Kandlur, D.Saha, K. Shin, "A Self-Configuring RED Gateway", *Proceedings IEEE INFOCOM '99*, vol: 3, pp. 1320-1328, 1999.
[4] T. J.Ott, T. V.Lakshman, L. H. Wong, "SRED: Stabilized RED", *Proceedings IEEE INFOCOM'99*, vol: 3, pp. 1346-1355, 1999.
[5] D. D.Clark, W.Fang, "Explicit Allocation of Best Effort Packet Delivery Service", *IEEE/ACM Transactions on Networking-TON*,vol: *6*, no: 4, pp. 362-373, 1998.
[6] R.Pan, B. Prabhakar, K.Psounis, "CHOK, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation", *Proceedings IEEE INFOCOM 2000*, vol: 2, pp. 942-951, 2000.
[7] C.V.Hollot, V. Misra, D. Towsley, W. Gong, "On designing improved controllers for AQM routers supporting TCP flows", *Proceedings IEEE INFOCOM 2001*,vol: 3, pp. 1726-1734, 2001.
[8] M. K. Şiş, A Dynamic Local Congestion Reducing Strategy Based on a Mini-Max Criterion. USA; Polytechnic Institute of New York University, Ph.D. Thesis, 1994.
[9] G.Çatalkaya, Simulation of a Local Congestion Reducing Routing Strategy for Multidestination Networks. Izmir; DokuzEylul University, Engineering Faculty, Msc. Thesis. 2003.
[10] J. Morrison, "Two Server Queue with One Server Idle below a Threshold", *Queueing Systems: Theory and Applications*, vol: 7, no: 3-4, pp. 325-336, 1990.
[11] L.Kleinrock,Queueuing Systems. vol I. Wiley,1975.
[12] L. Le, J. Aikat, K.Jeffay, F. D.Smith, "The Effects of Active Queue Management on Web Performance", *ACM SIGCOMM2003*, pp. 265-276, 2003.
[13] Z. Zheng, R. E. Kinicki, "Adaptive Explicit Congestion Notification Techniques for Heteregeneous TCP Flows", *Computer Science Technical Report Series*, 2001.
[14] G. Çatalkaya A New Approach to IP Level Congestion Control. Izmir; DokuzEylul University, Engineering Faculty, Ph.D. Thesis, 2011.
[15] J. Padhye, V. Firoiu, D. Towsley, J. Kurose, "Modeling TCP Thrughput: A Simple Model and its Empirical Validation", *ACM SIGCOMM 1998*,vol: 28, no: 4, pp. 303-314. 1998.

Note:

Malik Kemal ŞİŞ has received his BS and MSc degrees in Electrical Engineering in 1978 and in 1980, respectively, both from Istanbul Technical University. In 1994, he has received his Ph.D. in EE from Polytechnic University in New York. After gradution he has been employed as Networking design consultant for several private companies. Since 2001 he has been working for Dokuz Eylül University in Computer Engineering department as a member of faculty. His major research interests are Netwok control, Performance analysis, Digital Audio Processing and Quantum Computation.

Gökhan ÇATALKAYA has received his BS, MSc and Ph.D. degrees in Electrical & Electronics Engineering in 1999, 2003, 2011 respectively from Dokuz Eylul University. He has been working in a private company as an engineer since 1999. His main interests are networking, congestion control, network performence, and processor sharing.