An Efficient Queue Management for Adaptive and Non-adaptive Traffics

(Received November 30, 2001)

by Rabin SHAKYA
Noriyasu YOKOO
Hiroshi KOIDE
Yasuhiro SHIGEMATSU

Abstract

This paper considers the major negative impacts of deploying DropTail FIFO, the traditional queueing algorithm in almost all routers. The unfairness to the TCP traffics due to the lack of efficient queue management and the unavoided congestion collapse of the Internet due to unresponsive flows are discussed. To overcome the several defects of the conventional queueing algorithm, we propose an algorithm, which is fair to the different traffics even in the presence of heterogeneous traffics. The proposed algorithm classifies the incoming packets into a particular traffic type and an appropriate queueing algorithm is deployed to each traffic type, contradict to DropTail FIFO, which has a single queueing algorithm for any kinds of traffics. Our method does not require any time consuming schedulers. Our simulations show the TCP sharing a small portion of the bandwidth even in the time of congestion. We conclude with a comparison of the proposed algorithm to some of the conventional algorithms.

1. Introduction

Recently, there has been a tremendous increase of traffics in the Internet. The multimedia traffics, the real time applications, on-demand video and telephony services have been dominating the Internet and these traffics are predicted to be more increased in the coming years. These traffics, if classified accordingly to ISO/OSI (International Standard Organization/Open System Interconnection) are non-adaptive traffics like UDP (user datagram protocol) in the transport layer. The wide spread use of UDP is due to its simplicity and the transfer rate, better than other protocols like TCP (Transport Control Protocol). On the other hand, UDP traffics has a great influence on other adaptive traffics like TCP. When the adaptive and non-adaptive traffics compete for the bandwidth, it is always the non-adaptive which dominates the adaptive traffics[5,9]. Consiquently, the adaptive traffics either become scarce in the bandwidth or do not get even a small portion of the bandwidth in the worst cases. During the time of congestion, the Internet behaves as it is dedicated to non-adaptive traffics only delivering the non-adaptive pre-dominantly and dropping the adaptive ones. The Internet, which so often called the best-effort, must not be fair to UDP only, unless there is some sophisticated pricing system or some other means to give the priorities to the certain traffics. The stereotype that TCP is reliable is not always true. Like in the cases above TCP does not get the share of the bandwidth till the UDP stops and the transfer of the packets can never be guranteed.

One of reason for the above mentioned problem is in the protocols of TCP and UDP. TCP supports flow control where UDP does not, thus TCP slows its transfer rate on the detection of packet drop or the congestion, while UDP never decreases its transfer rate. Thus, there is always bandwitdth available for UDP, in comparision to TCP. On the other hand, the use of the end-to-end congestion mechanisms of TCP has been a major factor in the robustness of the Internet today. However, the Internet today, is no longer a small and closely knit user community, and it is no longer possible to rely on all end-nodes to use end-to-end congestion control. The intermediate nodes, whose architecture has been so simple, should also play an essential role in the congestion mechanisms. Moreover, the most effective detection of congestion can occur in the gateway itself. The gateway can reliably distinguish between propagation delay and persistent queueing delay. Gateway is shared by many active connections with a wide range of roundtrip times, tolerances of delay, throughput requirements, etc, decisions about the duration and magnitude of transient congestion to be allowed at the gateway are best made by the gateway itself.[5]

The second reason is the router, where the queueing algorithm is still the conventional Drop Tail FIFO (First In First Out) algorithm, which is being the standard Internet gateway queueing algorithm. Drop Tail has been deployed for its simplicity. It drops the incoming packets after the buffer is exhausted. The traditional technique for managing router queue lengths is to set a maximum length in terms of packets for each queue, accept packets for the queue until the maximum length is reached. Then, drops the subsequent incoming packets until the queue decreases because a packet from the queue has been transmitted. However, its drawbacks like Lock Out, Global Synchronization and Full Queues can result in severe service degradation or Internet meltdown [2]. Some of the drawbacks are discussed below:

LockOut

In most of the cases of DropTail allows a single connection or a few flows to monopolize the queue, thus other connections do not get a share of the queue. The packets from the faster connections have the least probability to be dropped. This causes the unfairness of the whole network.

Full Queues

The DropTail allows queues to maintain a full queue state for a long period of time and signals congestion only when the queue has become full. Thus, the feedback of the congestion is not sent to the sender at the appropriate time. If the queue is full or alomost full, an arriving burst will cause multiple packets to be dropped.

Global Synchronization

When the TCP packets are dropped the sender throttles down its speed. In some cases when the TCP packets are dropped, the window size in every TCP are decreased at the same time and when there is excess bandwidth, the window size is increased at the same time resulting a phenomenon of global synchronization, a cause of Internet bandwidth degredation.

The Danger of Congestion Collapse

The congestion collapse of the network occurs when it is at the full or near full utilization but little useful work is being is accomplished. In 1980, Nagle[7] was the first person to report the congestion collapse due to the unnecessary retransmission of the packets of TCP that were either in transit or had already been received at the receiver. Another kind of congestion collapse occurs when the bandwidth is wasted by delivering packets through the network that are dropped before reaching their ultimate destination.

Morever, DropTail algorithm does not decode the incoming packets except the port number and the IP address. While dropping the packets, the DropTail drops all the incoming packets, no matter whether the packets are important or not. The rejection of important packets like ACK (acknoledgement), ICMP (Internet Control Message Protocol) packets results severe service degradation.

2. Typical Queueing Algorithms

When the network gets congested, the excess packets are kept in a queue. Queueing algorithm is the mechanism to control the packets in the queue. There are two aspects of scheduling mechanisms; namely fairness and providing incentives for end-to end congestion control, the aspects that DropTail does not support. DropTail is well known for its simplicity, but the Internet now is being more complicated; the number of the packets flowing and their characteristics has been dramatically changed. Several algorithms have been implemented to overcome the defects of the Drop Tail FIFO. Some algorithms are discussed below:

Drop Front Algorithm

Drop Front algorithm[13], a simple algorithm, which drops the packets from the front when the queue is full and a packet arrives. The algorithm is very simple and the sender is allowed to detect the packet drop earlier and the time to retransmit can be decided at the earliest. But the time required to drop the front packets is large, it has to walk O (N) to drop a packet and dropping the front packets do not always solve the above problems. Moreover, some reasearchers argue that dropping packets that has been in the queue for the longer period is the waste of bandwidth and the time-out may occurs before the dropping the packets that has been moved from the tail to head,

Random Drop Algorithm

Random Drop algorithm, which drops the packets randomly from the queue to overcome the global synchronization and the full-queue problem, so that end nodes can respond to congestion before buffers overflow. By dropping packets before the queue limit, active queue management allows routers to control when and how many packets to drop. But the certain studies show that in Random

Drop gateways, if the queue length exceeds a certain drop level, then the gateway drops each packet arriving at the gateway with a fixed drop probability[5].

Random Early Detection, well-known as RED which is discussed later on this paper, also works well in comparision to DropTail but is not effective in the presence of heterogeneous traffics. And several other algorithms like BRED (Balanced RED)[4], Fair Queueing etc. have been proposed but neither has been the substitutional attempt to the Drop Tail. Thus, the above-mentioned drawbacks have never been solved. In most of the algorithms, all the traffics, no matter whether it is adaptive or non-adaptive, are treated with the same queue management algorithm as in DropHead and Random Drop. Thus the algorithm may be fair to a particular traffice but does not support heterogeneous traffics Fig.-3.

Weighted Round Robin

In some algorithms like Weighted Round Robin, a scheduler like Round Robin is used for delevering the packets from the queue after giving them an appropriate weight or priority. This algorithm provides a certain gurantee of the bandwidth and latency in some special cases but consumes a longer time to decide when to deliver packets and the algorithm itself is complicated to deploy. In such kinds of algorithms, either a number of queues have to be maintained or the scheduler has to process all the packets, which is time consuming process. Other similar kinds of algorithms are discussed in [1] and [10].

The traffic in the Internet has different characteristics; thus, each type of traffic should be queued in different queues to avoid the influence of other traffics and should be treated with the appropriate queue algorithms or schedulers. The quality of service (QoS) required by the data traffics like mail or Web and the real time datas like voice or videos are not the same. The former one is not sensitive on the latency of few seconds or the jitter but in the second one, the quality of voice and the image is affected. Here an algorithm that treats the traffic accordingly to their types is proposed.

3. Proposed Algorithm

In the Internet, congestion control is achieved mainly through end-host algorithms. However, starting with Nagle[7], many researchers observed that such end-to-end congestion control solutions are greatly improved when routers have mechanisms that allocate bandwidth in a fair manner. Fair bandwidth allocation protects well-behaved flows from ill-behaved ones, and allows a diverse set of end-to-end congestion control policies to co-exist in the network. In the presence of adaptive flows and non-adaptive flows, it is always non-adaptive flows, which get the higher portion of the bandwidth, and the adaptive flows suddenly get scarce of bandwidth. In most of the cases, every adaptive packet is dropped, before they enter the gateway, consiquently the sender stops sending packets. To overcome these drawbacks, here both of the flows are buffered in a separate queue virtually i.e. to every packets there is a threshold after which the packets are discarded, thus the influence of greedy non-adaptive flows do not affect the adaptive flows. It consists only one FIFO queue that has a maximum limit of queue with several thresholds to different traffics. As the packets are queued virtually accordingly to their traffic types, the proposed algorithm does not need any time consuming schedulers like Round Robin. The prompt delivery of the important packets is done to increase the efficiciency of transfer of the packets, so the important packets whose drop affects the efficiency of the connections and the small packets have the highest priority. Here, the traffics are classified according to their traffic types based on their characteristics. The proposed algorithm consists of two parts namely: Classifier and Queue Management.

3.1 Classifier

The classifier behaves as the initial point for each and every incoming packet. The grouping of the packets that has similar characteristics are done by the classifier. The classifier classifies the incoming traffics into adaptive, non-adaptive, small packets and others. For grouping of the traffics with similar kind in Transport level is done by the IP header. The protocol field of each incoming packet header is decoded and is classified.

In Fig-1, an IPV4 header is shown. The 8 bit protocol field in the IPV4 header identifies the next upper level protocol i.e. transport control protocol or others. Most of the recent algorithms are studying to utilize the 8-bit TOS (Type Of Service) field, which has not been used to classify the packets or to provide priorities. Several studies and workgroups are discussing the details of the TOS field, which may needs more time to discuss for its standard definition. But here, our effort is done to classify the packets by using the existing IP header. The protocol field with clear grouping of packets already exists; thus new definitions are not required. Some of the assigned Internet Protocol numbers in the Protocol field are listed below:

4-bit Version	4-bit header length	8-bit type of service (TOS)	16-bit total length in bytes
16-bit identification	3-bit flags	13-bit fragment offset	
8-bit time to live (TTL)	8-bit protocol	16-bit header checksum	
32-bit source IP address			
32-bit destination IP address			
Option Headers			Padding
DATA			

Fig-1 IPV4 Header.

Table 1. Internet Protocol numbers.

Decimal	Keyword	Protocol
1	ICMP	Internet Control Message
2	IGMP	Internet Group Management
4	IP	IP in IP (encapsulation)
6	TCP	Tramsmission Control
8	EGP	Exterior Gateway Protocol
9	IGP	Interior Gateway
17	UDP	User Datagram
92	MTP	Multicast Transport Protocol

From the Table-1 above the incoming packets can be classified easily from the protocol field. Some of the protocols listed in the above table may have further sub-groups like in TCP; the packets may be ACK, FIN or etc, which is not possible to identify from the table only. At the present the classifier classifies the various packets as follows:

Non-adaptive

Traffics, like UDP take as much bandwidth as they require and do not slow down under congestion. Applications like audio, video and telephony, which require faster transmission and the drop of few packets do not effect on the quality, are non-adaptive type. The sender should always keep in mind that some packets may be dropped while transmission and the dropped packets are never retransmitted. These types of traffics do not support flow control thus, congestion avoidance algorithms do not exist. If no action is taken, such unresponsive flows could lead to a new congestion collapse. These kinds of traffics have a particular sending rate and try to get the share of the bandwidth proportional to the seding rate, while these traffics do not increase the sending rate even in the abundant avability of the bandwidth and the full utilization of the bandwidth is not done. These kinds of traffics are with very simple header and are fast in transmission but non-reliable, the delivery of the packets is never guranteed. The protocol number for UDP in the IPV4 header is given 17.

Adaptive

These connections slow down when they detect the congestion or packet dropped and sends more packets if allowed as long as no packets are lost. As the network becomes congested and the buffer at the gateways fills up packet loss occurs. In response to that adaptive traffics decrease the sending rate. Thus the sending rate of these applications is changed according to the level of congestion perceived in the network. Packet marking or packet dropping serves the congestion indicator of the network. Adaptive traffics are window-based congestion control; the window size is the number of packets that can be transmitted at a time. The window-size decreases in the detection of packet drop and increases in the avability of bandwidth. The dropped or lost packets are always retransmitted; therefore these traffics are very reliable. TCP is an adaptive traffic the sending rate of the packets is proportional to the RTT (Round Trip Time), MSS (Maximum Segment Size and MTU (Maximum Transfer Unit) which varies in the time of congestion or packet drop. Thus adaptive traffics share the available bandwidth with each other and the full utilization of the available bandwith is done. The major transport protocol in use over the Internet is the TCP that provides end-to-end congestion control. These kinds of traffics are reliable and fast in transmission in the availability of excessive bandwidth. Applications like FTP and Web that are not latency conscious but require reliability use adaptive protocols. The protocol number for TCP in the IPV4 header is given 6.

Small Packets

Some packets that are very small in size but play very important roles in the Internet. Packets like ACK, ICMP packets and Telnet packets, which are very small in size but their loss or the drop of these packets slow down the Internet. The small packets, can be adaptive or non-adaptive. From the above list the small packets may be ICMP, IGMP, EGP or IGP. The small packets can be differentiated into two types. The first one is the one that has its protocol number in the protcol field of IPV4 header, which plays a vital role in the Internet and their drop or the retransmission affects the efficiency of the Internet. The second ones are those that are not listed in the protocol field. These packets are small only because of their size. The telnet packets can be considered as small packets, most of the time the telnet packets are very small in size. For e.g. when we send a 1 byte data through telnet, first the TCP or UDP headers of 20 bytes are added then the IP header of 20 bytes and ethernet header of 14 bytes are added. Thus the maximum overhead occurs in the above cases to transfer 1 byte data; a packet at least not less than 67 bytes has to be transferred.

Generally, here the incoming packets are classified on the basis of ISO/OSI transport layer i.e. TCP or UDP. The small packets do not have direct connection with the transport layer; thus measuring the packet size is the only way of identification of these packets. The method of identification of the important packets like ACK, FIN packet, which can be verified by the Transport Level only, is yet to be done.

4. Proposed Queue Management

Queueing management is the core of network that is responsible for the packet forwarding and packet discarding. Therefore, an efficient queue management must be deployed to supress latency and should be fair to every incoming packets but with a very few processing time. Here, the incoming packets after the protocol has been decoded are kept in a single queue. The queue works as a FIFO until the detection of the congestion, the FIFO works well in the absence of congestion. Obviously, the queue has a maximum queue limit and the packets arrived after the maximum limit is dropped. When the packets in the queue cross the threshold, the proposed algorithm is activated and the incoming packets after they are classified by the classifier. Then they are treated with an appropriate queueing algorithm. Here, each classes is treated with different algorithm as follows shown in the Table-2.

Table-2 Deployed Algorithms.

Traffic Type	Deployed Algorithm
Adaptive Traffic	Random Early Detection (RED)
Non-adaptive Traffic	DropTail
Small Traffic	Priority Queueing
Others	DropTail

The characteristics of the three various traffics types are very different from each other, thus they should be treated with an appriopriate algorithms. The non-adaptive traffics tend to get the share of the bandwidth proportional to the sending rate; thus the maximum effort should be given to fulfil the demand of these traffics. The adaptive traffics share the bandwidth with each other, thus even in the case of scarce bandwidth, each connections gets the fair share of the bandwidth. The small packets as mentioned above are the most important packets and should be given the highest priority. The adaptive traffics that send feed back to the sender in the time of congestion are treated with RED as it is originally developed for adaptive traffics. RED has been strongly recommended by [2] as the next generation queueing algorithm. But the disability of RED to support non-adaptive has been pointed out in several researches[3,4,12]. RED solves the global synchronization of the adaptive packets and drops the packets from the queue earlier before the queue becomes full. The feedback of the congestion and the packet drops are sended to the sender for the retransmission or the decrement of the sending rate. RED works very well with adaptive traffics, hence in this algorithm for adaptive traffics, it is deployed. The adaptive traffics support flow control, whereas other traffics do not support it. The adaptive traffics correspondly respond to the packet drop and traffic like TCP is non-speed oriented, rather it provides the gurantee of the packets transmission only. On the other hand, other traffics do not respond to the packet drop, thus deployement of RED with other traffics is meaningless. As in the case of deploying RED with UDP traffics, RED drops several packets randomly for the early notification of the congestion, but as UDP do not support retransmission, the feedback is of no use. Thus RED should be deployed only in the homogenous traffics of adaptive traffic only.

As non-adaptive traffics do not slow down even in the detection of congestion and packet drop. Hence here, the conventional DropTail FIFO is deployed for the non-adaptive traffics, thus the connections with higher speed gets more bandwidth than the slower ones. The DropTail FIFO delivers packets from the queue depending on the arrival of the packets. Consiquently the faster connections get more bandwidth. The non-adaptive traffics do not alter their transfer rate and the retransmission of the packets are not done, thus the need of random drop or the earlier drop of the packets is needless for non-adaptive traffics. Here, the question of the connection with the lower speeds does not get the share of the bandwidth may arise. The transfer rate of the connection is proportional to the charge payed, thus the higher connections get more bandwidth than the others do. The non-adaptive traffics like UDP are speed-oriented traffics and the DropTail FIFO algorithm works well with it.

Small packets play a vital role in the Internet. There is a lot of ovehead in the transmission of small packets, when the retransmission of these small packets has to be done due to their drop, the efficiency of the traffic decreases. The maximum effort is done here to avoid the retransmission of the small packets and other important packets like ACK, FIN and others. The incoming small packets are alwayes buffered and are never dropped. The small packets are also treated with Drop Tail, but the highest priority goes to the small packets as the discardment of these packets can cause degradation in the throughput. The packets that do not lie on the above mentioned traffic types are treated with the DropTail FIFO algorithm.

4.1 Dropping Behaviour

In conventional queueing algorithms like DropTail FIFO, the packets are dropped after a certain limit of the queue. In RED, there are two thresholds, after which the packets are dropped with a probability. Here, each traffic types have different thresholds, after which the packets are dropped with a probability. For UDP there is only one threshold and the packets are dropped after crossing the limit. With TCP, the dropping behaviour is the same to RED, where the packets are dropped with a certain probability if the queue length lies between the minimum and the maximum threshold. Here, the calculation of the size of average queue length (avg) is also done as in RED using a low-pass filter. Thus, the short-term increases in the queue size that result from bursty traffic or from transient congestion do not result in a significant increase in the average queue size. Bursty traffic may be from FTP connection with a long delay-bandwidth product with a small window; a window of traffic will be sent and then there will be a delay until the ack packets return and another window of data can be sent. Variable-bit-rate video traffic and some interactive traffic are bursty traffics. The low-pass filter is an exponential weighted moving average (EWMA)

$$avg = (1-w)\ avg + w*q$$

The weight (w) determines the time constant of the low-pass filter and q is the queue length at that instance. The size and duration of bursts in queue size that are allowed at the gateway determine the queue weight. The use of the average queue size is essential in RED to give a tolerance for transient congestion so that a packet is not dropped from a transient queue caused by a TCP slow-start to eliminate global synchronization, and to eliminate biases against bursty traffic. Here, weight is kept 0.002 as suggested by RED.

For the small packets the maximum queue length is the threshold to drop. Here, for easiness the maximum threshold of TCP and small packets or the maximum queue length is set equal. Another difference of the dropping behaviour is the packet recognition, in the conventional algorithms, while dropping packets the router never know what kind of packets it dropped, neglecting the importance and the effect of dropping such packets. Here, each packet are decoded and the importance of the packets is studied and are dropped, thus the dropping mechanism of the proposed algorithm is serving a better and efficient method to increase the efficiency of the Internet. Thus the proposed algorithm is equipped with various threshold to drop the packets and may also be used as an admission control.

4.2 Random Early Detection (RED)

RED is an active queue management for congestion avoidance in packet-switched networks that has been recommended by IETF (Internet Engineering Task Force) as the next generation queueing algorithm. RED use randomization to ensure that all connections encounters the same loss rate. It also prevents congestion rather than just reacting to it, by dropping packets before the gateway's buffers are completely full or by marking the packets. RED does not require per-flow state and thus easy to add to the existing IP gateway and have little impact on packet forwarding efficiency.

The RED algorithm consists of two main parts: estimation of the average queue size and the decision of whether or not to drop an incoming packet. The packets are dropped with a certain probability when the average queue size lies in between the minimum threshold and the maximum threshold and all the packets are dropped after the average queue size cross the maximum threshold.

RED is well-known for its several advantages. Deploying it the global synchronization is prevented and the feedback of the congestion of the queue is sended to the sender. The early drop of the packets before it gets full, solves the problem of lock-out and the sender can know the congestion before-hand.

4.3 Algorithm

A very simplified form of the proposed algorithm is shown below:

```
/*************************************
**************************************
 Max = Max packets allowed in the queue
 Max Udp = Max UDP packets allowed
 Max Tcp = Max Tcp packets allowed
 Min Tcp = Min Tcp packets allowed
**************************************
*************************************/

If   flow > Max
        drop;

else  switch "flow"
   case "UDP" :
         if queue length > Max Udp
                drop;
         else enqueue;

   case "TCP" :  /* RED algorithm */
         if queue length > Max Tcp
             drop;
         else if Min Tcp <  queue length <
Max Tcp
             Drop random;
         else enqueue;

   case "Small Packet" :
         enqueue;

   default : FIFO algorithm;
```

Fig-2 A Simplified form of the algorithm.

The queue has a maximum queue limit (Max), after which every incoming packet is dropped. The queue length is calculated after a packet is enqueued or dequeued. For every traffic types, after the packets are classified, a particular queue limit is setted, after which the packets are dropped. In the case of UDP, it holds only one threshold (Max Udp) but for TCP there are two thresholds as RED algorithm recommendation. RED serves byte mode and packets mode, here packet mode is deployed. The packets that enter the queue after the initial threshold are dropped by a certain probability. The maximum threshold of TCP is set equal to the maximum queue length. The value of the thresholds used here differs from the values recommended by RED. The values of the thresholds in the original RED is used and studied for the heterogeneous traffics, here RED is used for the homogeneous of adaptive traffics only, thus the values used here are deployed after experiments and studies. But the small packets do not have any threshold, so they are always enqueued if the queue length is less than the maximum limit. If the classifier is not able to classify the packet then the default FIFO algorithm is deployed

5. Simulation

Several simulations, with different parameters and conditions were done with the proposed algorithm, the DropTail FIFO algorithm and RED algorithm. For all of the simulations presented, the NS[11] simulator, which has been the major simualtor for the research of Internet and Networkings, is used. However, here the small packets are not classified for the easiness of the simulation.

Fig-3 shows one of the simulations with sources consisting

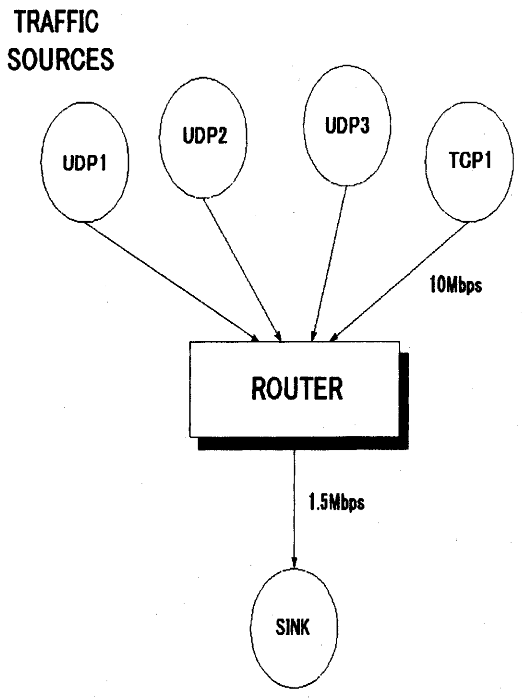

Fig-3 A simple network with a bottle-neck gateway.

of a TCP and three UDP with the connection of 10Mbps to a router.

In the above simulation, the maximum size of queue is set to 15 packets, after which the incoming packets are dropped The packets from the sources are sended to the sink. The sink is set up with the connection of 1.5 Mbps for a bottle necked gateway. The packet size of TCP is 1000 bytes and UDP with the size of 300, 400, 500 bytes and the transmission transfer rate of UDP is .006, .005, and .004 secs respectively. The transfer rate of the TCP varies and is dependent on the state of traffic, thus is not shown in the figure. The above simulation is constructed to analyze the behaviour of the traffics with multiple UDPs and limited number of TCP showing the present Internet dominating by UDP. In every simulations, the soureces are sended to the sink, the sender trensmits another packets after receiving the ack packets for the adaptive traffics. The simulations were continued for 5 seconds and the result is analyzed.

5.1 Result

The result of the above simulation is shown in the Fig-4, Fig-5 and Fig-6. It is the bandwidth governed by the TCP and UDP sources in Mbps and time in secs.

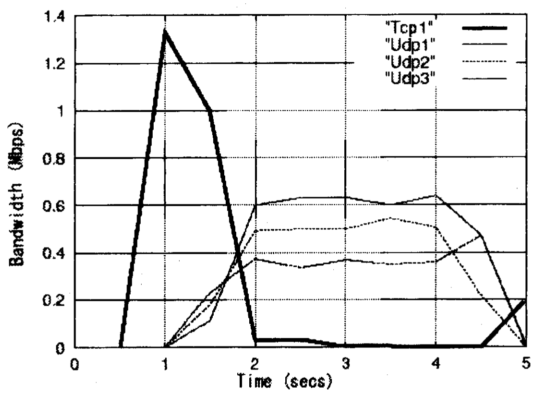

Fig-4 Simulations with DropTail

In Fig-4, TCP starts from 0.5 secs and risese gradually, and then it gets the whole share of bandwidth in the absence of other traffics in the first second. The bandwidth shared to TCP goes on decreasing as the greedy traffics like UDP enter from the first second, which clearly shows that UDP steals the bandwidth and hinders the throughput of TCP. The sender decreases its transfer rate to send packets and the packets sended also are dropped, resulting the shortage of bandwidth. The UDP traffics that start from the first second get their desired bandwidth and continue the fixed share of the bandwidth. There is no flow control in UDP, thus even in the congestion period also; the sender keeps on sending the packets. After three seconds, the maximum bandwidth is shared to UDP only and consiquently TCP is blocked. The next TCP is delivered only after UDP stops, if UDP is sent continuously for a longer period as in video traffics, the TCP has to wait for the same longer period, thus the Drop Tail algorithm does not distribute the bandwidth to TCP in the time of congestion. This shows the unfairness of DropTail algorithm to TCP and UDP packets act as they are given the higher priorities.

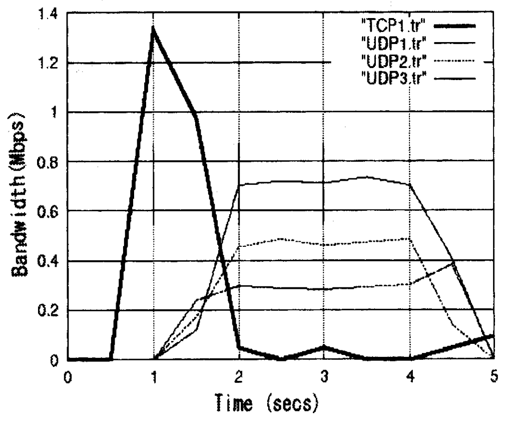

Fig-5 Simulations with RED

In Fig-5, with RED algorithm, the result is more or less the same to Fig-4; the problems are not solved as with the DropTail. In RED the packets are randomly dropped after the queue cross the threshold. Dropping TCP sends incentives to the sender that responds by decreasing the speed but dropping UDP does not send any feed-back to the sender. Thus some of the UDP packets or TCP packets may be dropped consiquently some of the TCP packets get a very tiny share of the bandwidth (3secs). But the incoming UDP packets, which do not resppond in packet drop enter the queue, the TCP again becomes scarce in the bandwidth.

In Fig-6, TCP packets get a small portion of bandwidth even in the time of congestion. In the previous results, TCP did not get a single share of bandwidth after 3 secs but here TCP stops at 3 secs but soon it gets a small share of bandwidth. In the proposed algorithm the influence of UDP over TCP packets is deminished. As the influence of UDP packets to TCP packets is decreased. The TCP packets get a very small portion of the bandwidth even in the time of congestion. Comparing with the above figures Fig-4 and Fig-5, the bandwidth of UDP are almost the same. Thus, deploying the proposed algorithm, the fairness of the gateway can be maintained even in the presence of heteregenous

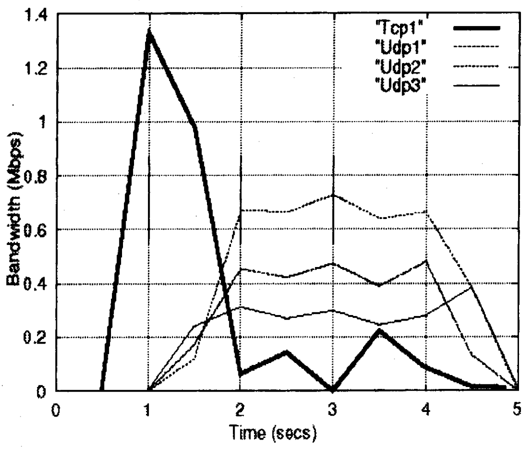

Fig-6 Simulations with the proposed algorithm

traffics.

Several other simulations were done to show that the proposed algorithm works well with not only the above simulation but with different environments and the result was found satisfactory. In the above simulation, there is only one TCP with multiple UDPs, another simulation with a UDP and multiple TCP were performed. A similar kind of result was obtained, TCP sharing the bandwidth and not constraining the UDP. Therefore, the proposed algorithm can be applied in any kind of traffics and environments and can be the substitutional algorithm to the DropTail.

5.2 Evaluation and Conclusion

Here an algorithm is proposed that classifies and treats the traffic accordingly to their types and the efficiency of the Internet is increased by setting the highest priority to the small packets. The proposed algorithm does not require any schedulers as separate queues are deployed virtually.

The major advantage of the proposed algorithm is the influence of the greedy traffics over the other traffics is minimized. The unfairness of the adaptive traffics due to the monopoly of the non-adaptive traffics is diminished. By classifying the packets in particular traffic types, sophisticated functions can be supported. Like in the case of ECN (Explicit Congestion Notification), which notifies the next routers about the congestion. When ECN is used in heterogeneous traffics, the ECN bit flag in the UDP is of no use, where as the TCP respondes to the congestion. The defects of DropTail are solved using the proposed algorithm.

The proposed algorithm is very simple to implement and can be easily subsitute the exsiting algorithms. The classification of the packets is very simple and can be implemented easily from the existing Protocol field of IP header. It is a parameter sensitive algorithm; the threshold of the traffics can be changed accordingly to personal needs for the better results.

The classifier used here may have some influences from the DiffServ (Differentiated Services). In DiffServ the packets are classified by identifying the TOS field, an 8-bit field in the IP header which has not been used for a longer period. DiffServ use the 6bit of the TOS field and has three classes namely AF (Assured Forwarding), EF (Expedited Forwarding) and BE (Best-Effort Forwarding). These classes are further sub divided but the details are yet to be defined. Here the 8-bit Protocol field which is used for grouping the traffics. The advantage of classifier used here over DiffServ is that the Protocol field has been defined clearily and is already exist in the IP header, that allow us for the prompt deployment of the algorithm.

6. Future Research

In the proposed algorithms the small packets are not used, the simulations must be done with these small packets for the efficient queue management of the heterogeneous traffics. The proposed algorithm is a parameter sensitive; thus the simulations with a varities of parameters in different environments should be done. Contradiction to the real practice, the size of the packets in the simulation is kept fixed for the easiness. The simulation with variable packet size and the bursty packets should be performed. As the traffics are classified accordingly to their traffic type, the proposed algorithm may be used in the DiffServ. The algorithm at present behaves merely as the best-effort, but the priorities given in the DiffServ can be further classified to the traffic types explained here, and may be used for the precise gurantee of the bandwidth and latency. Moreover, the experiments should be done based on the above simulations to study the real characteristics of the algorithm.

The defects of the proposed algorithm can be the time consuming while identifying each incoming packet and the definition of the small packets is yet to be done. Further studies must be done on small packets for the more effecitveness of the algorithm or the algorithm may be extended to byte mode.

References:

[1] Archan Misra (Doctor dissertation), "Dynamics of TCP congestion avoidance with Random Drop and Random marking queues", University of Maryland, College Park, p.194, 2000.

[2] B.raden, D.Clark, et al.,"Recommendations on Queue Management and Congestion Avoidance in the Internet", RFC 2309, April 1998.

[3] Dong Lin, Robert Morris, "Dynamics of Random Early Detection", Proceedings of SIGCOMM 97, 1997.

[4] Farooq M.Anjum, Leandros Tassiulas, "Balanced=RED: An algorithm to Achieve Fairness in the Internet", Center for Satellite and Hybrid communication Networks, p.29, March 1999.

[5] Floyd, S., Jacobson "Random Early Detection gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, Vol.1, No.4, pp.397-413, Aug. 1993.

[6] Mikkel Christiansen, Kevin Jeffay, David Ott, F.Donelson Smith "Tuning RED for Web Traffic", ACM Digital Library, pp.246-249, Sep. 2001.
http://www.cs.unc.edu/Research/dirt

[7] Nagle, J. "Congestion Control in IP/TCP", RFC 896, Jan. 1984.

[8] Rabin Shakya, Noriyasu Yokoo, Yasuhiro Shigematsu, "An Efficient Queue Management for Adaptive and Non-adaptive Traffics", Record of 2001 Joint Conference of Electrical and Electronics Engineers in Kyushu, pp.465-465, 2001.

[9] Sally Floyd, Kevin Fall, "Router Mechanisms to Support End-to-End Congestion Control", Network Research Group, Lawrence Berkeley National Laboratory, Berkeley CA, p.19, Feb. 1997.

[10] Sally Floyd, V.Jacobson, "Link-sharing and Resource management Models for Packet Networks", IEEE/ACM Transactions on Networking Vol.3, No.4, pp.365-386, 1995.

[11] S.Keshav "REAL network simulator", May 1990.
http://www.mash.cs.berkeley.edu/ns/ns.html

[12] Stefaan De Cnodder, Omar Elloumi, Kenny Pauwels, "Effect of different packet sizes on RED performance", Traffic and Routing Technologies project, Alcatel Corporate Research Center, Belgium, p.9, 2000.

[13] T.V. Lakshman, Arnie Neidhardt, Teunis Ott, "The Drop From Front Strategy in TCP over ATM and its Interworking with other control Features", Infocom 96 MA28.1, 1996.