ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2, Issue 4 (Oct. - Dec. 2014)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

Analyzing the performance of RED, RARED and CoDel in Active Queue Management

[I]Anita Yadav, [II]Pratap Singh Patwal
[I]Student, [II]Assistant Professor
[I,II]Dept. of Information Technology, IET Group of Institutions

Abstract
Active queue management (AQM) denotes to a family of packet dropping mechanisms for router queues that has been suggested to maintenance end-to-end congestion control mechanisms in the Internet. The suggested AQM algorithm through the IETF is Random Early Detection (RED). The RED algorithm permits network operators concurrently to reach at high throughput and low average delay. But, the resulting average queue length is comparatively sensitive to the level of congestion. The Refined Adaptive RED (RARED), use for decreasing the sensitivity to parameters that affect RED performance. CoDel Algorithm is proposed for controlling the congestion in routers. We compare these three algorithms on the basis of simulations results. CoDel is parameterless and it treats good queue and bed queue differently. So we see that the CoDel scheme improves almost performance of the network.

Keywords
RED, RARED, Bufferbloat, CoDel, AQM, tail-drop.

I. Introduction

With the quick advancement of communication and networking technologies in the last three decades, the Internet has renewed the biggest artificial system in the world. It is one of the most important and fastest media for carrying information today. One may bring together and direct varioustypes of information such as data, speech and movie, in the form of E-mail, web-pages, online conference and online TV/Radio. The way of life of individuals has been considerablydeveloped by this range of information distributed and related information processing and utilization. But, the present usage of the Internet is distant beyond its original design envelope and made reasons of various operating and performance difficulties. In this paper, we are taking consideration on traffic modelling and active queue management strategies to improve the performance of the Internet.

The most common type of computer network is a packet-switched network, where nodes send data in the form of packets to each other. The most common strategy used to transfer data is store-and-forward. Every node waits till it has received entire packet before accelerating it at a later time to the suitable output link. The Internet is an example of a network that is prominently packet-switched. The data route from a source to destination is computed by different methods by routers. When we talk about congestion control we essentially talk about control of data packets in these routers. Congestion take place in a router when the total bandwidth of arriving packets, destined for a certain output link, go beyond the link'sbandwidth. Next is illusion of congestion taking place at a router is show in Fig 1.

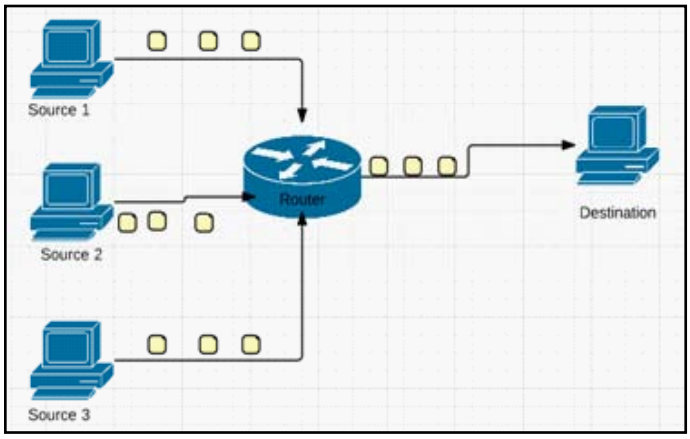

Fig 1: Small congested network.

There are two kinds of congestion: transient and persistent [1]. Transient congestion can be managed with a queue of buffer at the router. All through to the congestion time the queue will increases and hold the excess packets. When the congestion period finished, the buffered data is sent to the suitable output link. On the other hand, persistent congestion is supposed to happen when the data excesses the buffer. Whereas transient congestiononly leads a delay in data transmission, persistent congestion's results in data loss. These problems are tackled in two ways. Either the router detects the queue build up and informs the sources to decrease their transmission rate. Such a strategy is called Congestion avoidance. The other method is to use end-to-end strategies where the routers do not get directly involved but the hosts use indirect methods to detect congestion. Such a mechanism is known as Congestion Control. In TCP/IP, the most commonly used protocol in the Internet, both methods are used to tackle problem related to congestion.

In, it is pointed out that the very detection of occurrence of congestion in a network may not be very easy. The immediate side effects of congestion are data loss or unacceptable delay in transmission. However, these may be results of faulty routers or corrupt packets. Hence these do not necessarily form good indices for congestion detection. Also, delay or loss of performance is very much a qualitative measure. Delay may be acceptable for one type of user and not to another. However, proper congestion

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

control mechanisms do result in better network performance under heavy load conditions.

Completed the previous few years, there has been a remarkable growth in the Internet usage and the variety of its applications. Transmission Control Protocol (TCP) has been the main protocol of the Internet always since its beginning. The achievement of the Internet, in fact, can be fairly accredited to the congestion control apparatuses implemented in TCP. Still, incredible progress in the variety of bandwidth, growth in Bit-Error Rates (BER) and bigger diversity in the applications of the next-age group networks have faced the congestion control mechanisms of TCP.

The performance of TCP centered applications, spaced out from the congestion control mechanisms, censoriously depends on the selection of queue management scheme employed in the routers [2]. Queue management schemes control the length of the queues via proactively dropping packets when required. Passive Queue Management (PQM) (e.g., tail-drop) is the best commonly deployed queue management mechanism in Internet routers. PQM does not occupy any proactive packet drop beforehand the router queues become full and hereafter, is simply set up. Though, due to intrinsic problems of PQM such as global synchronization and lock-out, IETF praises the deployment of Active Queue Management (AQM) for the next-age group Internet routers. Furthermore, one more drawback of PQM named persistently full buffer difficulty (in recent times open as a portion of Bufferbloat) has proved the need of wide spread placement of AQM.

As a outcome, AQM mechanisms have been widely considered in the current past to observe and limit the increasing queues at router. These mechanisms escape congestion by proactively telling the TCP source about congestion, either by dropping or by marking a packet. Random Early Detection (RED) is the most broadly deployed AQM mechanism in the routers. On the other hand, it has been presented that the usefulness of RED mainly depends on properly setting at least four parameters, that are: minimum threshold (min_th), maximum threshold (max_th), queue weight factor (Wq) for exponential weighted moving average and maximum drop probability (P_max). Ideal values for these parameters vary for dissimilar situations and later, setting suitable values for these parameters has been a thoughtful problem ever since the beginning of RED. While a lot of RED variations have been suggested in the literature, there is still a lot of averseness in the extensive acceptance of RED because these variations added difficulties its mechanism.

In recent times, a different AQM mechanism called Controlled Delay (CoDel) has been suggested to reduce the limitations of PQM and RED. Unlike RED, CoDel is parameterless AQM mechanism that adjust to change link rates and can be easily set up. Furthermore, unlike RED and its variants that use average queue sizeas a predictor of congestion, CoDel uses packet sojourn timeto predict congestion. In this project, the fair study of RED, RARED and CoDel is made as a prequel.

II. Random Early Detection (RED)

RED has been intended to remove several of the Drop Tail problems. In particular, RED active queue management scheme is capable to accomplish high throughput and low average delay by spreading randomly packet's drops in the middle of flows [1]. RED identifies congestion using an exponentially weighted moving average (EWMA) of the queue size, and probability drops or marks packets to control the congestion at router buffers.

RED algorithm manages the queue in a more active way by randomly dropping with increasing probability as the average queue size increases [1]; the packet drop rate rises linearly from zero; when the average queue size is at the RED parameter Minimum Threshold (min_th) to a drop rate of P_max when the average queue size reaches Maximum Threshold (max_th).

Random Early Detection recovers network performance by fixing the parameters of RED. Performance of RED is sensitive to rapid modifications in network traffic and packet sizes [3]. RED provides fair network resources to all escaping the global synchronization. Following are the RED parameters:

1. Minimum Threshold min_th
2. Maximum Threshold max_th
3. Queue Weight Factor (W_q)
4. Maximum Drop Probability P_max
5. Packet Drop Probability P_d

These parameter values depend on number of flows passing through router and packet size.

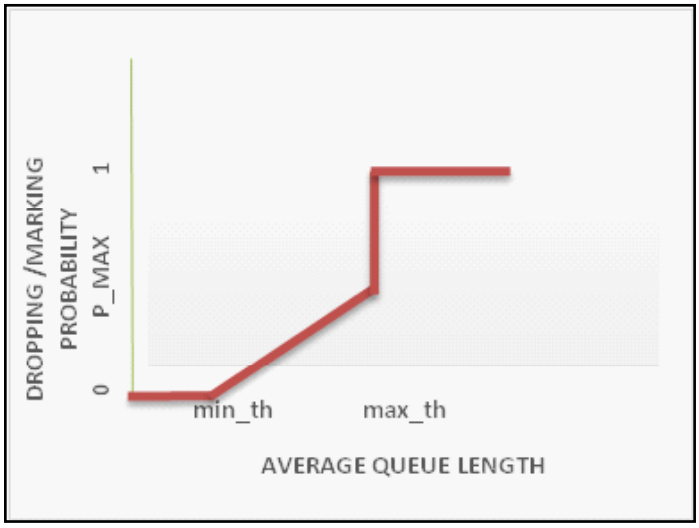

Fig. 2: The marking/dropping behavior of RED [6]

Algorithm 1: RED Algorithm

On the arrival of each packet
Calculate average queue size (avg) using exponential weighted average:
avg=((1- Wq) * oldavg)+ Wq * curq
Here,
oldavg :Average Queue Size during pervious packet arrival
curq : Current Queue Size
If avg < min_th then
Enqueue the packet
Pd = 0
If avg> max_th then
Drop the packet
Pd = 1
If min_th< avg>max_th then
Drop the packet with certain probability
Pd=(avg-min_th)/(max_th-min_th)*P_max

Packet drop probability is a linear function of avg. When avg > max_th , Pq increases suddenly to 1 resultant is high packet drops.

Though RED is perhaps the most commonly used AQM scheme for congestion avoidance and control but it has been saw from numerous studies [5] [6] [7]that the performance of RED is

extremely dependent upon the situation where it is used along with the manner its parameters are adjusted. Therefore, the performance benefits of RED as claimed in [2] and others are not commonly correct.

- Throughput: Depends upon traffic intensity and the way its parameters are tune.
- Mean Queue length: Packet drop probability rises with rises in mean queue length.
- Packet Loss Probability: When avg< max_th, then the packets jump to drop.
- Link Utilization: Link utilization is well-organized in instance of small queue/buffer size.
- End-to-End Delay or Latency: Delay may rises in case of large queue size.

III. REFINED ADAPTIVE RED

We present the Refined Adaptive RED algorithm (RARED) that aims at improving the performance of ARED. The RARED algorithm is to maintain the average queue size near the specified target queue size. RARED adapt P_max to keep the average queue size near the specified target queue size, it is reflects the desired average queuing delay[4]. Our aim is to bring the average queue size closer to its target value more quickly. For this purpose, we compare the current average queue size with specified target queue size. The proposed algorithm packet marking/dropping policy is shown in Figure 3.

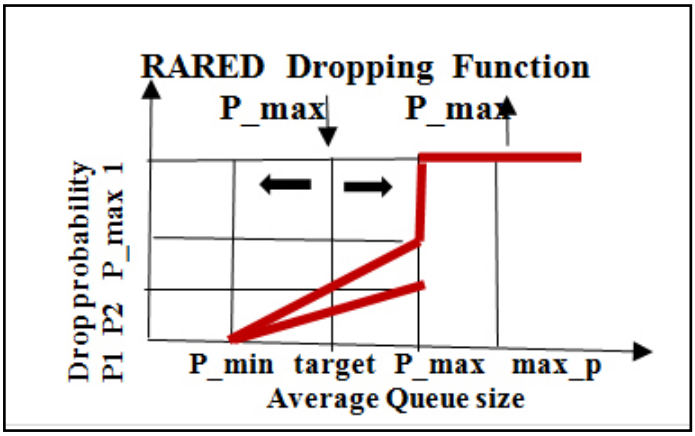

Fig.3: The evolution of the RARED dropping function[4]

The overall guidelines for RARED as implemented here are the same as those for the ARED. The overall RARED, which was implemented, has the following, features:

- P_max is adapted to keep the average queue size near specified target queue size.
- P_max is adapted slowly, over time scales greater than a typical round-trip time and in small steps.
- P_max is constrained to remain with the range of [0.01, 0.5].
- Instead of multiplicatively increasing and decreasing P_max we use an additive-increase multiplicative-decrease (AIMD) policy.

Algorithm 2: RARED Algorithm

For each interval seconds:
If (avg > target and P_max < _0.5) then
α = 0.25 * (avg - target) / target * P_max
P_max= P_max+α
Else if (avg < target and P_max >_0.01) then
β=1-(0.17*(target-avg)/(target-min_th))

P_max =P_max * β

Variable:
avg: Current average queue size

Fixed Parameters:
target: Time; 0.5 seconds
target: Target for average queue size;
[min_th + 0.48 *(max_th-min_th],
min_th + 0.52 * (max_th –min_th)]
α: increment P_max
β: decrement P_max

The guideline of adapting max_p slowly and infrequently allows the dynamics of ARED – of adapting the packet dropping probability in response to changes in the average queue size - to dominate on smaller time scales[4]. The adaptation of max_p is invoked only as needed over longer time scales.

IV. CoDel ALGORITHM

In order to be in effectcounter to bufferbloat, a answer in form of an Active queue management (AQM) algorithm must be capable to identify an happening of bufferbloat in a queue and respond with deploying effective countermeasures[7][8].

Some important characteristics of CoDel are numbered below:

- It is parameterless--no knobs/handles are required for operators, users, or implementers to adjust.
- It handle good queue and bad queue in a different way - that is, it keeps the delays smallbutallowing bursts of traffic.
- It manage delay, whereasunaffected to round-trip delays, link rates, and traffic loads.

It get used to dynamically varying link rates with no harmfuleffect on utilization. Authors while working with CoDel differentiate between two types of queues used in the concept of this algorithm. The queues are discussed in the next section.
CoDel relies on the packet sojourn time i.e. the actual queue delay experienced by a packet as a metric to predict congestion in the network. If the packet sojourn time is above the targetvalue for a specified interval of time, CoDel starts proactively dropping/ marking the packets to control the queue length. However, CoDel avoids the underutilization of outgoing link by not dropping/ marking the packets proactively in case if current queue size is less than one MTU.

Algorithm 3: CoDel Algorithm

On arrival of every packet:
If $current_{queue\_size}$<$queue_{limit}$then
Enqueue the packet
Attach a timestamp in packet header
End
Else
Drop the packets
End
On the departure of every packets
$enqueue_{time}$= timestamp for enqueue time
$dequeue_{time}$=timestamp for dequeue time
$sorojan_{time}$=dequeue_time-enqueue_time

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

Ifsorojan$_{time}$<target
or current$_{queue_size}$<MTU then
do not drop packets
Come out of dropping state
End

Else
Whiledequeue$_{time}$ >_next$_{drop_time}$do
Drop the packets
count= count+1
next$_{drop_time}$ + =interval / √ count
End
End
End
Else if outside droping state and first packets is being dropped then
Enter the dropping start
End

The algorithm works in two parts: (i) at the time of enqueuing the packet and (ii) at the time of dequeuing the packet[7]. On arrival of each packet, the current queue size is tested. If it is less than the queue limit, the packet is enqueued and the timestamp is attached in the header. This timestamp specifies enqueue time. On departure of each packet, the timestamp is removed from the header and is deducted from the current time to obtain the packet sojourn time.

The CoDel algorithm remains either in the dropping state or not in the dropping state. If the packet sojourn time above the target for a listed interval of time, CoDel go into the dropping state and starts proactively dropping/marking the packets. Note that the packets are proactively dropped while dequeuing rather than during enqueuing. The time interval between the two proactive packet drops is calculated.The count showsthe total number of packets dropped since the dropping state is entered.

While the algorithm is in dropping state, if the packet sojourn time come to be lesser than target or if queue does not have enough packets to fill the outgoing link, the algorithm leaves the dropping state.

There are two supreme important CoDel parameters to be set to reachfinestresults [7]: target and interval. These are stable parameters and their values are selected based on the explanations from numerousresearch.Succeeding are the values for target and interval[7]:

- target = acceptable standing queue delay (constant 5ms)
- interval = time on the order of worst case RTT through the bottleneck (constant between 10ms to 1sec)

V. Conclusion

The RED algorithm is simple and effective as an active queue management mechanism. Yet, the resulting average queue length is relatively sensitive to thelevel of congestion and the RED parameter settings, and is hence not predicable in advance. We present a variant Refined Adaptive RED in traffic load. The RARED can adapt ARED drop probability built on traffic load, and it can effectively reduce packet drop rate in congestion network and preservefixed average queue size in mixed traffic.The CoDelis free of queue size, queue size averages, queue size thresholds, rate measurements, link utilization, drop rate, queue occupancy time or round trip delays [7]. It reaches high link utilization and reduces the queue occupancy in most of the simulated scenarios. The CoDel is a very common, effective, parameterless AQM methodology that can be useful to congested queues. It is a serious tool in solving bufferbloat.

References

[1] *V. Jacobson, "Congestion Avoidance and Control", in Proc. ACM SIGCOMM, pp. 314–329, August 1988.*
[2] *M. Allman and V. Paxson, "TCP Congestion Control", Internet Engineering Task Force, RFC 2581, April 1999.*
[3] *S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", IEEE/ACM Transactions on Networking, vol. 1, no. 4, Aug. 1993.*
[4] *Tae-Hoon Kim, Kee-Hyun Lee," Refined Adaptive RED in TCP/IP Networks" SICE-ICASE International Joint Conference 2006Oct. 18-2 1, 2006.*
[5] *T. Bonald, M. May, and J. Bolot. "Analytic Evaluation of RED Performance," INFOCOM'00,2000.*
[6] *Shakeel Ahmad, Adli Mustafa, Bashir Ahmad, Arjamand BanoAnd Al Sammarraie Hosam,"Comparative Study OfCongestion Control TechniquesIn High Speed Networks" International Journal of Computer Science and Information Security,Vol.6, No. 2, 2009.*
[7] *Dipesh M. Raghuvanshi, Annappa B., Mohit P. Tahiliani, "On the Effectiveness of CoDel for Active Queue Management" 2012 Third International Conference on Advanced Computing &Communication Technologies.*
[8] *K. Nichols and V. Jacobson, "Controlling Queue Delay," ACM Queue Magazine: Networks, vol. 10, no. 5, pp. 68–81, May 2012.*