Cooperative Congestion Control in P2P Networks Edge Router

Prof. Bharathi. M1, Manu GV2

1Associate Professor, Department of Computer Science and Engineering, SJCIT, Chickballapur, India
2M.Tech in Computer Science and Engineering, SJCIT, Chickballapur, India

Abstract -- Next generation data centers and cloud computing design architectures require high data rates of the order of 100 GBPS from the Internet. This paper presents Random Early Detection (RED) edge routers for congestion avoidance in P2P networks. The edge routers detect the possible congestion by computing the average queue size. The edge router could notify connections of congestion either by dropping packets arriving at the edge router or by setting a bit in packet headers. When the average queue size exceeds a preset threshold, the edge router drops or marks each arriving packet with a certain probability, where the exact probability is a function of the average queue size. RED edge routers are designed to accompany a transport-layer congestion control protocol such as TCP, UDP. In particular, a novel AQM (Active Queue Management) algorithm is adopted in the forwarding board to guarantee the UDP flow fairness. Our Simulations of a UDP network are used to illustrate the performance of RED edge routers along with the novel cooperation congestion control can achieve better fairness and data flow protection.

Keywords- congestion control, TCP, UDP, RED, Router, AQM, fairness

I. INTRODUCTION

Congestion control mechanism such as those in TCP and UDP are not enough to prevent congestion collapse in the Internet (for starters not all applications might be willing to use them) and they must be supplemented by control mechanisms inside the network. Buffers are key component in the networking which are used for the purpose of storing the data larger buffers can absorb larger bursts but they tend to build up at high load and increase queuing delays. The traditional technique for managing delay is to set a maximum length for each buffer queue accept packets in the queue until the maximum length is reached then drop subsequent incoming packets until the queue decreases below its maximum value. This buffer management scheme is referred to as Tail Drop. End-to-end control mechanisms are used in the Internet to regulate the amount of traffic in the network and match it to available capacity thereby making sure that queue lengths and loss rates remain reasonable.

The Congestion Occurrence In Intermediate Switches

Not all applications are willing to use control mechanisms; in particular at a rate independent of the state of congestion in the network and thus grab all the bandwidth of the network when competing with rate adaptive applications. Clearly the uncontrolled use of such applications again raises the possibility of Internet-wide congestion collapse. The difficulties above bring out the necessity to complement traditional control scheme mechanisms with router-based control mechanisms that extend beyond the current Tail Drop scheme. The objective is to provide a detail understanding of the Active Queue Management and the steps involved in the implementation of router based control mechanisms. This project provides the clarity and we focus on the congestion detection and the flow-based load balancing problems on a multistage circuit switching fabric known as Combined Input Output Queue architecture. Although the Combined Input Output Queue network is a non-blocking architecture in theory. The key design issue is how to allocate the links within the two-level switches and how to avoid network congestion as illustrated in Figure 1. Cell-based schedulers such as Combined Input Output Queue cannot probe packets boundaries effectively and will result in the significant performance degradation. In all architectures of Combined Input Output Queue switches the multi-level Combined Input Output Queue switches are adopted in asynchronous on-chip networks introducing much smaller overhead than crossbars.

Figure 1: Congestion Occurred in 2-level Cos switches
The Edge router includes multiple sub-router virtual nodes. In this paper each block of edge board can be considered as an edge router and forwarding board can be considered as a core router. The control panel can be seen as Open Flow controller. Based on this architecture the first non-blocking Edge router not only controls the internal node congestion but also ensures the forwarded data to select the other path automatically. The normal behavior of Edge router queues on the Internet is called tail-drop. Tail-drop works by queuing up to a certain amount then dropping all traffic that ‘spills over’. This is very unfair and also leads to retransmit synchronization. When retransmit synchronization occurs the sudden burst of drops from a router that has reached its fill will cause a delayed burst of retransmits which will over fill the congested router again.

In order to cope with transient congestion on links backbone Edge RED routers will often implement large queues. Unfortunately while these queues are good for throughput they can substantially increase latency and cause TCP connections to behave very burstily during congestion. These issues with tail-drop are becoming increasingly troublesome on the Internet because the use of network unfriendly applications is increasing. The Linux kernel facilitates us RED short for Random Early Detect also called Random Early Drop as that is how it behaves and works. RED statistically drops packets from flows before it reaches its maximum limit. This causes a congested backbone link to slow more gracefully and prevents retransmit synchronization. This also helps TCP or UDP find its 'fair' speed faster by allowing some packets to get dropped sooner keeping queue sizes low and latency under control. The probability of a packet being dropped from a particular connection is proportional to its bandwidth usage rather than the number of packets it transmits. RED is a good queue for backbones where you can't afford the complexity of per-session state tracking needed by fairness queuing.

Congestion control mechanism such as those in TCP and UDP are not enough to prevent congestion collapse in the Internet (for starters not all applications might be willing to use them) and they must be supplemented by control mechanisms inside the network. The traditional technique for managing delay is to set a maximum length for each buffer queue accept packets in the queue until the maximum length is reached then drop subsequent incoming packets until the queue decreases below its maximum value. This buffer management scheme is referred to as Tail Drop. End-to-end control mechanisms are used in the Internet to regulate the amount of traffic in the network and match it to available capacity thereby making sure that queue lengths and loss rates remain reasonable.

II. ANALYSIS AND RELATED WORKS

Next generation data centers and cloud computing design architectures require high data rates of the order of 100 GBPS from the Internet. However most of things at edge routers stand point are typically implemented as clusters of computer and line cards where it is difficult to allocate bandwidth and to guarantee flow fairness. Not all applications are willing to use control mechanisms; in particular at a rate independent of the state of congestion in the network and thus grab all the bandwidth of the network when competing with rate adaptive applications. Clearly the uncontrolled use of such applications again raises the possibility of Internet-wide congestion collapse. The difficulties above bring out the necessity to complement traditional control scheme mechanisms with router-based control mechanisms that extend beyond the current Tail Drop scheme.

Drawbacks in Existing System
- The Normal behavior of the Edge RED Router queues on the Internet is called Tail Drop
- Tail-Drop works by queuing up to a certain amount
- Then dropping all traffic that 'Spills Over'
- Complexity of Estimation and Iterative Planning
- Complexity in Tool Selection
- Congestion Occurrence.
- Tail-Drop is very unfair
- It leads to retransmit synchronization
- Stabilize the Router functionality is too tedious
- Possibility of loosing important information due to congestion.

III. PROPOSED METHOD

The objective is to provide a detail understanding of the Active Queue Management and the steps involved in the implementation of edge router based control mechanisms. This paper summarize to produce a edge router based congestion avoidance mechanism using RED algorithm implemented at the edge router to manage the queue and to avoid incipient congestion. To achieve this goal the three main procedures yet to implement in the congestion control mechanism at the Edge routers in P2P networks i.e. Edge Red Router, Red Implementation and Packet Format. It is targeted objective to implement an efficient algorithm at the edge router for congestion avoidance.
• **Congestion Control**: RED identifies the earliest stages of congestion and responds by randomly dropping packets. If the amount of congestion continues to increase RED drops packets more aggressively to provide the queue from reaching 100 percent capacity which would result in a complete denial of service. This allows RED to maintain upper bound on the average queue depth even with non cooperative transport layer protocols.

• **Global Synchronization**: Because RED doesn’t wait until a queue is 100 percent full to begin discarding packets RED allows a queue to accept burst of traffic and not discard all of the packets in burst. As a result RED is protocol friendly because it doesn’t discard clusters of packets from any single protocols session and helps avoid global synchronization.

• **Queue Control**: RED allows maintaining the amount of traffic in the queue at the moderate level. RED supports the fair discarding of packets across multiple flows without requiring the router to maintain state for the amount of traffic carried by the each flow traversing in a given queue.

There are many areas for research on RED edge routers:

- Determining the optimum average queue size for maximizing throughput and minimizing delay for various network configurations.
- This question is heavily dependent of the characterization of the network traffic as well as on the physical characteristics of the network.
- The traffic dynamics with a mix of Drop Tail and RED routers as would result from partial Deployment of RED routers in the current internet.
- The behavior of the RED router machinery with transport protocols other than TCP including open- or closed-loop rate-based protocols

The major area where significant rework is to be done is regarding the parameters of the RED algorithm:

- These parameters have to be implemented and understood well enough to perform the action smoothly if not this will give rise to the situation which is worse than the Tail Drop Queue Management.
- Intended to implement Graphical utility that shows all the parameters used in the Red Router such as buffer used no of packets received packets marked packets rerouted etc. The performance of the algorithm can be reevaluated by altering parameters like packet process delay red drop probability for packet discarding.

**IV. RED ALGORITHM ANALYSIS AT EDGE ROUTERS**

RED congestion control mechanisms monitor the average queue size for the output queue and choose connections to notify of the incoming congestion. It discards packets that arrive at the router selectively hence TCP connections after they have detected lost packets reduce their transmission rate and congestion can be prevented. Moreover RED drops packets in a probabilistic manner and such probability grows with the estimated average size of the queue.

**Algorithm:**

**Initialization**

\[
\text{avg} = 0 \\
\text{count} = -1
\]

For each packet arrival

\[
\text{avg} = (1 - wq) \times \text{avg} + wq \times q
\]

If \( \text{minth} < \text{avg} < \text{maxth} \)

Increment count

\[
\text{Pb} = [(\text{avg-minth}) / (\text{maxth-minth})] \times \text{maxp}
\]

\[
\text{Pa} = \text{Pb}/1-\text{count} \times \text{Pb}
\]

with probability \( \text{pa} \):

mark the arriving packet

\[
\text{count} = 0
\]

Else if \( \text{maxth} < \text{avg} \)

mark the arriving packet

\[
\text{count} = 0
\]

Else \( \text{count} = -1 \)

**Notations:**

- **[1] Set Of Saved Variables:**
  - \( \text{avg} \): average queue size
  - \( \text{count} \): packets since last marked packet

- **[2] Set Of Fixed Parameters:**
  - \( wq \): queue weight (weighted moving average)
  - \( \text{minth} \): minimum threshold for queue
  - \( \text{maxth} \): maximum threshold for queue
  - \( \text{maxp} \): maximum value for pb

- **[3] Set Of Other Parameters:**
  - \( \text{pa} \): current packet-marking probability
  - \( q \): current queue size
The average queue size is compared to two parameters is shown in equation 1: the minimum queue threshold \( q_{\text{min}} \) and the maximum queue threshold \( q_{\text{max}} \). If the average queues size is smaller than \( q_{\text{min}} \) the packet is admitted to the queue. If it exceeds \( q_{\text{max}} \) the packet is marked or dropped. If the average queue size is between \( q_{\text{min}} \) and \( q_{\text{max}} \) the packet is dropped with a drop probability \( P_b \) that is a function of the average queue size

\[
p_{b,i} = \begin{cases} 
0 & \text{if } q_{k+1} < q_{\text{min}} \\
1 & \text{if } q_{k+1} \geq q_{\text{max}} \\
\frac{q_{k+1} - q_{\text{min}}}{q_{\text{max}} - q_{\text{min}}} \cdot p_{\text{max}} & \text{otherwise}
\end{cases}
\]  \( (1) \)

where \( p_{\text{max}} \) is the maximum packet drop probability. The relationship between the drop probability and average queue size is shown as follows.

![Figure 2: Graph giving the relationship between \( P_a \) and \( q_{\text{avg}} \)](image)

The final drop probability \( P_a \) is given by equation 2. It is introduced in order to avoid the severe increase of the drop rate shown in Figure 2.

\[
P_a = \frac{p_b}{1 - \text{count} \cdot p_b}.
\]  \( (2) \)

In equation 2 the \( \text{count} \) is the cumulative number of the packets that are not marked or dropped since the last marked or dropped packet. It is increased by one if the incoming packet is not marked or dropped. Therefore as \( \text{count} \) increases the drop probability increases. However if the incoming packet is marked or dropped \( \text{count} \) is reset to 0.

**Evaluation of RED parameters**

Unlike Drop Tail the performance of the RED algorithm is not determined only by the buffer size. Other parameters such as queue weight (\( w_q \)) maximum drop rate (\( p_{\text{max}} \)) and the two queue thresholds (\( q_{\text{min}} \) and \( q_{\text{max}} \)) are also of great importance. The network designer should select these appropriately so that the system can achieve better performance. \( w_q \) is an exponentially weighted average factor which determines the time constant for the averaging of the queue size. If \( w_q \) is too low then the estimated average queue size is probably responding too slowly to transient congestion. If \( w_q \) is too high then the estimated average queue size will track the instantaneous queue size too closely and may result in an unstable system. In quantitative guidelines are given for setting \( w_q \) in terms of the transient burst size that the queue can accommodate without dropping any packets. The optimal setting for \( w_q \) is between 0.001 and 0.005.

The packet drop probability is calculated as a linear function of the average queue size if the average queue size is between \( q_{\text{min}} \) and \( q_{\text{max}} \). However the maximum drop probability determines the oscillation of the drop rate. If \( p_{\text{max}} \) is set too small then the network behaves similar to the Drop Tail shown in Figure 3. However if this value is set too high the packet drop probability increases thus enforcing the system to oscillate severely and experience a decrease in throughput. It has been shown that steady-state packet drop rates of 5% to 10% would not be unusual at a router. There is therefore no need to set \( p_{\text{max}} \) higher than 0.1 for real network implementations.

![Figure 3: RED drop probability with \( P_{\text{max}} = 0 \)](image)
The optimal values for $q_{min}$ and $q_{max}$ depend on the desired average queue size. If the typical traffic patterns are fairly bursty then $q_{min}$ must be correspondingly large to allow the link utilization to be maintained at an acceptably high level. This value should also be associated with the buffer size of the network. If $q_{min}$ is set too small it leads to low bandwidth utilization. Conversely if $q_{min}$ is set too high it may result in unfair competition for bandwidth among multiple links thereby cancelling out the benefits of the RED algorithm. The optimal value for $q_{max}$ depends in part on the maximum average delay that can be allowed by the router. The Edge RED router functions most effectively when the difference between $q_{max}$ and $q_{min}$ is larger than the typical increase in the calculated average queue size in one roundtrip time. A useful rule of thumb is to set $q_{max}$ to at least twice the size of $q_{min}$. If the difference between $q_{max}$ and $q_{min}$ is set too small then the computed average queue size can regularly oscillate up to $q_{max}$. This behavior is similar to the oscillations of the queue size up to the maximum queue size experienced with Drop Tail routers.

**V. FURTHER WORK AND CONCLUSIONS**

Random Early Detection routers are an effective mechanism for congestion avoidance at the router in cooperation with network transport protocols. If Edge RED routers drop packets when the average queue size exceeds the maximum threshold rather than simply setting a bit in packet headers then Edge RED routers control the calculated average queue size. This action provides an upper bound on the average delay at the router. The probability that the Edge RED router chooses a particular connection to notify during congestion is roughly proportional to that connection’s share of the bandwidth at the router. This approach avoids a bias against bursty traffic at the router. For Edge RED routers the rate at which the router marks packets depends on the level of congestion avoiding the global synchronization that results from many connections decreasing their windows at the same time.

The Edge RED router is a relatively simple router algorithm that could be implemented in current networks or in high-speed networks of the future. The Edge RED router allows conscious design decisions to be made about the average queue size and the maximum queue size allowed at the router.

This paper work covers a congestion avoidance algorithm for a higher level of congestion avoidance in Edge RED Routers. In this work the threshold is calculated based on randomly generated values. Future work can be carried on calculation of threshold by optimizing the average $q$-size for maximizing the throughput with minimal delay. The algorithm can be further enhanced with incorporation of traffic management algorithms. This paper is implemented by targeting UDP/TCP and further enhancements can be carried out to make it compatible with other protocols. Although much research effort has been focused on understanding and utilizing RED algorithm to leverage the current network, some interesting research topics are yet to be investigated in more detail in future. For example, since it is widely accepted that Poisson model is not sufficient to characterize the traffic in current Internet, it is important to understand how RED and similar Active Queue Management (AQM) algorithm act when self-similar network traffic is applied. Further studies may produce more meaningful characterization of RED performance in the real-world network.

**ACKNOWLEDGEMENT**

I would like to express my immense and deepest gratitude to Prof. Bharathi. M, Associate Professor, Dept of CSE, I am very much thankful to Dr. S N Chandrashekara, HOD, Department of Computer Science and Engineering, SJCIT Chickballapur, for being supportive, technical assistance and for continuous encouragement and insightful suggestions which helped me to successfully complete this paper.

**REFERENCES**

[1] LingYun Lu., RenPing Liu., “Cooperative Congestion Control in Multi-Level InterWeave Routers” The 2012 International Symposium on Communications and Information Technologies (ISCIIT), 978-1-4673-1157-1/12


