Greedy Flow Control by FRED Active Queue Management Mechanism

Alireza Gharegozi
Faculty Member of Computer Group
Islamic Azad University – Shahin dez Branch
Shahin dez, Iran
Gharegozi@yahoo.com

Abstract. The emergence of multimedia- and real-time applications lead to the throughput of the TCP flows converges to zero while the throughput of the CBR flows converges to the link-bandwidth. Furthermore, unresponsive flows cause congestion collapse. This paper presents evaluation of several active queue management algorithms with respect to their abilities of maintaining high resource utilization, identifying and restricting unresponsive flows. The performance of FRED is compared based on simulation results. Simulation is done by using NS2 and the graphs are drawn using x-graph. The simulation results show that FRED is often fairer than RED, but is not able to restrict greedest flow.

Keywords: unresponsive flows; congestion control; random early detection; FRED; active queue management;

1. Introduction

Congestion control for IP networks has been a recurring problem for many years. The problem of congestion collapse encountered by early TCP/IP protocols has prompted the study of end-to-end congestion control algorithms in the late 80’s and proposals such as [1], which forms the basis for the TCP congestion control in current implementations. The essence of this congestion control scheme is that a TCP sender adjusts its sending rate according to the rate (probability) of packets being dropped in the network (which is considered a measure of network congestion).

It is still an accepted assumption that Internet traffic is dominated by TCP. However, the rise of applications such as voice, audio and broadcast services increases the use of UDP as a transport protocol. Internet traffic is a mixture of various kinds. Some sources use congestion control mechanisms such as TCP; others, such as constant-bit-rate (CBR) video, do not react to congestion. Since UDP lacks any functionality to adapt to network traffic congestion, a substantial increase in UDP usage raises serious concerns about fairness and stability in the Internet [2], [3].

An extreme formulation of the fairness problem due to heterogeneous congestion control mechanisms is the problem of “unresponsive flows”. Unresponsive flows have no end-to-end congestion control implemented hence they do not back off at all in response to packet loss. As a result, Internet users have an incentive to be misbehaving, to utilize non-conservative congestion control mechanisms and thereby generate flows having a low level of friendliness.

TCP’s end-to-end congestion control [4] adapts the volume of data a source transmits into the net to the actual load situation by varying the congestion window as a function of the packet loss rate. Queue
management is the decision when to start dropping packets and which packets to drop at a congested router output port. Common Internet routers employ Drop Tail [5] queue management, discarding arriving packets if the buffer of the output port overflows. Drop Tail gateways often distribute losses among connections arbitrarily. Small differences in the round trip times of competing TCP connections, for instance, can cause large differences in the number of packets a Drop Tail gateway discards from the connections. Drop Tail gateways also tend to penalize bursty connections.

Contrary to Drop Tail, active queue management (AQM) [6] mechanisms start dropping packets earlier in order to be able to notify traffic sources about the incipient stages of congestion. The IETF is advocating the use of active queue management such as Random Early Detection (RED) [7] and Flow Random Early Drop (FRED) [8] to prevent packet loss. RED has been proposed in order to alleviate the problems of simple drop-tail queue management. The basic idea behind RED queue management is to detect incipient congestion early and to convey congestion notification to the end-hosts, allowing them to reduce their transmission rates before queues in the network overflow and packets are dropped. FRED, a modification to RED that improves fairness when different traffic types share a gateway.

The idea behind active queue management is to detect incipient congestion early and to convey congestion notification to the end hosts, in order to allow them to reduce their transmission rates before queue overflow and packet loss occur [7]. This paper presents the evaluation of RED and FRED active queue management mechanisms. The performance of RED and FRED is compared based on simulation results. Simulation is done by using NS2 [9] and the graphs are drawn using X-graph.

2. Related Work

2.1. TCP Congestion Control

During slow start, a TCP increments congestion window (cwnd) by one segment for every ACK received. The slow start ends when cwnd exceeds the slow start threshold (ssthresh), and congestion avoidance takes over. During congestion avoidance, cwnd is increased linearly, i.e. by 1 segment per round-trip time (RTT). When a TCP sender detects segment loss using retransmission timer, the value of ssthresh is halved, and the cwnd is reset to 1 segment. The lost segment is retransmitted [4]. The TCP algorithm is given in Fig. 1.

Over the years a lot of research concerning TCP has been carried out and many modifications and extensions such as Reno [10], New Reno [11], SACK [12] and Vegas [13] have been proposed.

2.2. RED : Random Early Detection

The Random Early Detection (RED) algorithm has been proposed in order to alleviate the problems of simple drop-tail queue management. RED keeps the average queue size low, allows occasional packet bursts, and prevents global synchronization of source windows due to its randomness in marking or dropping packets at a congested node. However, it has been proven through simulations that an unresponsive bandwidth greedy connection gets a larger than fair share of the bandwidth at a bottleneck link when competing with responsive connections at a RED gateway. RED monitors the average queue size (avg) of a shared FIFO queue at the router output port. Based on comparison of the average queue size against a minimum (minth) and a maximum threshold (maxth) a packet dropping probability (pa) is computed. The RED algorithm is given in Fig. 2. A detailed explanation of the RED algorithm can be found in [7].

The RED based AQMs such as PD-RED [14], Auto RED [15], MRED [16], DS-RED [17] tried to solve the various problems existing with RED. Some of these AQMs tried to get rid of the parameter tuning problem in RED. While some of them tried to improve the performance compared to RED. The problem of unfairness was attempted by some of the AQMs.
2.3. FRED: RED with per active Flow Accounting

Flow Random Early Drop (FRED), a modification to RED that improves fairness when different traffic types share a gateway. FRED is more effective in isolating ill-behaved flows, provides better protection for bursty and low-speed flows, and is as fair as RED in handling identical robust flows such as large bulk-data transfers. FRED provides these benefits by keeping state for just those flows that have packets buffered in the gateway. The cost of this per-active-flow accounting is proportional to the buffer size and independent of the total number of flows, except to the extent that buffer use depends on the number of active flows. FRED keeps track of the number of bytes a flow has stored in the queue (the “instantaneous per-flow queue size”) and computes the packet dropping probability as a function of the average queue size and the ratio of the instantaneous per-flow queue size to a “fair per-flow-buffer allocation”. The fair per-flow buffer allocation is given by the average queue size divided by the number of active flows. The FRED algorithm is given in Fig. 3. A detailed explanation of the FRED algorithm can be found in [8].

For each packet arrival:
  Calculate the average queue size
  If $q > 0$
    \[ \text{avg} = (1 - wq) \text{avg} + wq \text{q} \]
  else
    \[ \text{avg} = (1 - wq/m) \text{avg} \]
    \[ \text{If minth} < = \text{avg} < \text{maxth} \]
      Calculate packet dropping probability
      \[ \text{pb} = \maxx((\text{avg} - \text{minth})/(\text{maxth} - \text{minth})) \]
      \[ \text{pa} = \text{pb} / (1 - \text{count pb}) \]
      With probability $\text{pa}$ Drop the arriving packet
      else if $\text{maxth} < = \text{avg}$
        Drop the arriving packet.

3. Simulations

The following simulation, executed with the NS2[9] simulator illustrates the problems of unfairness and congestive collapse due to unresponsive flows. 4 bulk data TCP flows and 1 CBR flows send from the left to the right over the network shown in Fig. 4. The CBR flow (flow5) send with rate of 1.5Mbps. Flows 1, 2, 3, 4, 5 traverse the bottleneck link between the R2 and R3. All packet size = 1000 bytes and buffer size = 25 packets.
3.1. **RED Queueing**

RED parameters are set as follows: minth = 6, maxth = 18. Fig. 4 shows that the low-pass filtering mechanism of RED causes the average queue size to change slowly in comparison to the actual queue size. Initially as the responsive sources open their windows, the actual queue size remains low. When the average queue size crosses maxth, 18 in this case, the RED algorithm is triggered (A and B points in Fig. 5.). Fig. 6 shows that the 1 flow5 allocates nearly 100 percent of the link-bandwidth. 4 TCP flows are shut out at the congested link between the R2 and the R3 routers. Table 1 and 2 report the R2’s output port packet queue statistics for all flows. Thus, flow5 should be limited.

![Network Topology](image)

![Average and Instant Queue sizes in R2(RED queueing)](image)

![Queue occupancy by flows(RED queueing)](image)
### TABLE I. The R2’s Output Port Packet Queue Statistics

<table>
<thead>
<tr>
<th>Flow</th>
<th>Number of Packets at Time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Flow1</td>
<td></td>
</tr>
<tr>
<td>Flow2</td>
<td></td>
</tr>
<tr>
<td>Flow3</td>
<td></td>
</tr>
<tr>
<td>Flow4</td>
<td></td>
</tr>
<tr>
<td>Flow5</td>
<td></td>
</tr>
</tbody>
</table>

### TABLE II. The R2’s Output Port Packet Queue Statistics

<table>
<thead>
<tr>
<th>Time</th>
<th>Queue Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Active Connections</td>
</tr>
<tr>
<td>A</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>3</td>
</tr>
</tbody>
</table>

### 3.2. FRED Queueing

Fig. 7 shows that the all flows allocate nearly equal percent of the link-bandwidth in FRED queue management mechanism. FRED is often fairer than RED, also protects adaptive flows from non-adaptive flows by enforcing dynamic per-flow queuing limits.

![Figure 7. Queue occupancy by flows (FRED queueing)](image)

FRED queuing has some benefits. But, FRED suffers from several shortcomings. In case of many active flows the fair per-flow buffer allocation is low. Hence, at the arrival of a traffic burst the instantaneous per-flow-queue-size is likely to exceed the fair per-flow buffer allocation enabling a bias against bursty flows. FRED is not able to restrict greediest flow.

### 4. Conclusions and Future Work

This paper, presented implementation and evaluation of the RED and FRED congestion control mechanisms on the NS2 testbed. RED reduces the time period that a router is overloaded by causing the endpoints to infer that the router is congested before congestion at the router has become critical. But, the basic problem of a lag between congestion’s occurrence and its remedy remains.
FRED queuing has some benefits. But, FRED suffers from several shortcomings. In case of many active flows the fair per-flow buffer allocation is low. Hence, at the arrival of a traffic burst the instantaneous per-flow-queue-size is likely to exceed the fair per-flow buffer allocation enabling a bias against bursty flows.

This paper demonstrated that discarding packets in proportion to the bandwidth used by a flow does not provide fair bandwidth sharing at a gateway. A selective discard mechanism is needed to protect flows that are using less than their fair share, and to prevent aggressive flows from monopolizing buffer space and bandwidth.

Current end-to-end feedback congestion control systems detect and relieve congestion only at end points. But, there is an increasing interest in solving the congestion problem for Internet with, Active networks [18,19,20,21,22] with the idea of reprogramming routers with data packets. This paper will extend to evaluation of using active network based mechanism to solve the unresponsive connections problem.

5. References


