Managing Best Effort Traffic: RED

Christophe Deleuze

LIP6 - C.N.R.S.
Université Pierre et Marie Curie - Paris - France
Christophe.Deleuze@lip6.fr

Abstract. Best effort traffic in the Internet is managed by end to end congestion control mechanisms whose limitations are known. The continuous Internet growth leads to more and more applications that don't "play the rules" of the end-to-end congestion control scheme. Simple mechanisms inside the routers, refered as random early detection (RED), provide a more efficient congestion control. It can also be used to protect fair applications from unfair ones. RED can be deployed progressively as it does not require any modification of the applications using the classical Internet congestion control scheme.

Keywords: Internet, congestion control, random early detection, router

1 The Internet and Congestion Control

The Internet has been primarily designed and will probably be used in the forthcoming years to carry best effort traffic. To allow efficient managing of best effort traffic, congestion control mechanisms have been defined. These mechanisms apply on a end-to-end basis, and are implemented in the transport protocol, TCP. Among them is the well-known *additive increase, multiplicative decrease* [5].

First, we describe two aspects of problems that exist with the TCP congestion control scheme (its imperfection and the impact of possibly non compliant sources) and present the approaches that are proposed to solve them. We then show how the RED mechanisms can achieve a much more efficient congestion control. We discuss the way to used them (in association with simple scheduling techniques) to prevent non compliant sources to deprive compliant ones. Finally, we discuss the advantages and drawbacks of using *explicit congestion notification* instead of packet drops.

2 Problems with the TCP Congestion Control Scheme

End-to-end congestion control associated with FIFO scheduling in the routers presents some limitations, including:

- Applications using UDP instead of TCP can "shut out" TCP flows.
- Global synchronization effect, resulting in reduced aggregate throughput [6].
- Biases against bursty traffic [3].

– Phase effects resulting in unfairness [3].
– Transient and persistent congestion are not distinguished.

Moreover, in high delay-bandwidth product networks, such as high speed networks, routers are likely to have large buffers to accommodate transient congestion. Current congestion control mechanisms that apply only when a queue overflows could lead to large delays. A good congestion control scheme aims at a hight throughput and a low average queue size.

The authors of RED argue that the most effective detection of congestion can occur in the router itself, as it can reliably distinguish between propagation delay and persistent queueing delay.

Also, news problems arise due to the continuous extent of the Internet. It is no longer possible to rely on end-nodes to use end-to-end congestion control. The network itself must now participate in controlling its own resource utilization. In [2], Floyd and Fall identify three possible ways to do it:

– flow isolation in each router,
– rely on pricing mechanisms to control sharing,
– develop in the routers incentives for flows to use end-to-end congestion control.

Flow isolation solutions have a *scalability* problem. Moreover, the fairness they ensure is purely local (at each node), and may not make sense at the flow level. For example in Fig. 1, where two flows AB and CD compete, the bandwidth is shared fairly on the R1–R2 link, but as flow BD experiences congestion on the R2–D link, it could use less bandwidth on the R1–R2 link.

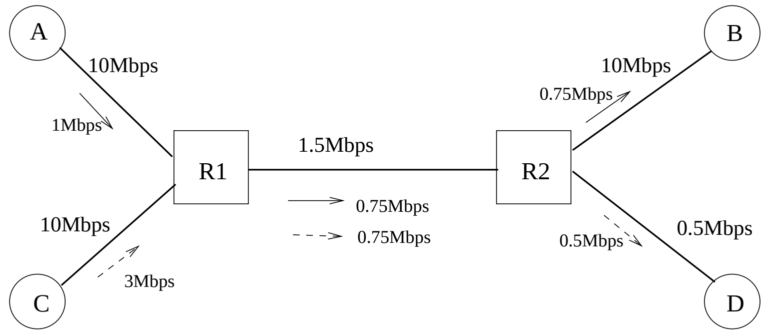

Fig. 1. Local fairness may not make sense

Rely on flow pricing means to gamble that the network providers are able to provision additional bandwidth and deploy appropriate pricing structures fast enough to keep up with the growth in unresponsive best-effort traffic in the Internet.

We describe the last solution in the following. Note that those three approaches are not necessarily mutually exclusive.

3 RED Congestion Control

RED (Random Early Detection) [4] uses the mechanisms of feedback-based transport protocols (especially TCP) for the network to actively inform applications about the bandwidth they can use. Thus, RED can be progressively deployed and does not imply other software changes that in the routers.

In a FIFO router, packets are dropped when the queue overflows, i.e. congestion is already there. RED tries to prevent congestion to occur, by dropping packets (thus implicitly asking applications to reduce their bandwidth usage) *before* congestion happens.

RED manages two thresholds for the queue, a minimum min_{th} and a maximum max_{th}. The average queue length avg is recomputed for each arriving packet. A simple version of the RED algorithm is shown below:

```
for each packet arrival
       calculate the new average queue size avg
       if min_th ≤ avg ≤ max_th
               calculate probability p_a
               with probability p_a:
                      drop/mark the arriving packet
       else if max_th < avg
               drop the arriving packet
```

RED has different methods for:

- detecting congestion, by watching out the average queue length;
- choosing which connections to notify of this congestion, by choosing randomly packets to mark.

In a classic TCP network global synchronization occurs because all applications lose packets at the same time when a queue overflows. By choosing randomly packets to discard, this is avoided. The distinction between transient and persistent congestion is achieved by the management of an *average* queue size. The maximum burst size considered as transient congestion (that triggers no packet marking/dropping) depends directly on the method used to compute the average queue length. The more "conservative" the average computation is, the large the bursts that are accepted.

As the average queue length reach min_{th}, some packets may be dropped or just marked (carry an indication of congestion for the application) with a probability increasing with avg. Normal TCP implementations only react to packet drops, but one could imagine to adapt TCP, or another transport protocol to react to such *explicit congestion notification*. This is briefly discussed in Sect. 5. We do not present the details of the calculation of the average queue size avg and the packet drop probability p_a. The reader can refer to [4].

If the average queue length goes beyond the maximum threshold, all arriving packets are dropped. This allows RED to control the queue size even in presence of non cooperative flows (i.e. that do not react on congestion indication — dropped or marked packets).

RED queue management can operate in *packet mode* or in *byte mode*. In *byte mode*, the probability to drop a packet is a function of the packet size in bytes. RED in *packet mode* will give flows an incentive to use larger packets, but this can be useful for routers limited by the number of *packets* arriving from each flow, rather than the number of *bytes*.

Concerning misbehaving users, we note that the probability that a router chooses a particular flow to drop a packet from is roughly proportional to this flow's share of the bandwidth. Thus, if a flow has a large fraction of the recent dropped packets, then it has most likely received a large fraction of the recent bandwidth. This gives an easy method for identifying high-bandwidth flows when congestion occurs.

4 Advanced RED-based Mechanisms

Floyd and Fall [2] present RED-based mechanisms to encourage best effort flows to use end-to-end congestion control. These mechanisms are able to:

- identify high-bandwidth flows,
- identify flows which should be regulated, including:
  - non "TCP-friendly" flows,
  - "unresponsive" flows,
  - other flows that use too much bandwidth.
- regulate flows that have been decided to.

4.1 Identifying High-bandwidth Flows

Identifying high-bandwidth flows can be done efficiently by using the RED packet drop history. It does not require the router to keep per-flow state, and can be achieved by a periodic pass in the background over information about the packets dropped at the router by RED.

The definition of a flow depends on the considered environment. One interesting possibility is to use source and destination IP addresses. This would not be "fooled" by an application using multiple TCP connections to increase its throughput. Thiner granularity can be obtained by using source and destination addresses and port numbers. Backbone routers might use a coarser granularity (subnetwork addresses for example).

We start by identifying two kinds of packet drops. *Forced packet drops* happen when $avg > max_{th}$, while *random packet drops* (which can be just markings) happen when $min_{th} \leq avg \leq max_{th}$.

Then we define two drop metrics for a flow:

- *packet drop metric:* ratio of the number of packets dropped from that flow over the total number of dropped packets.
- *byte drop metric:* ratio of the number of bytes dropped from that flow over the total number of bytes dropped.

The *packet drop metric* is shown to be meaningful to the *random packet drops*. In byte mode, it estimates the flow's fraction of the aggregate arrival in bytes per seconds. In packet mode, it estimates the flow's fraction of the aggregate arrival rate in packets per second.

The *byte drop metric* instead is shown to be meaningful to the *forced packet drops*, and it gives the best estimate of the flow's arrival rate.

Finally, we define the *combined drop metric* by weighting the flow's byte and packet drop metrics by the ratio of forced and random packet drops:

$$M_{forced} \times f_{forced} + M_{random} \times f_{random}$$

where M_{forced} is the flow's byte drop metric for the forced packet drops, M_{random} is the flow's packet drop metric for the random packet drops, and f_{forced} and f_{random} are the fraction of the total packet drops that are forced and random, respectively.

Non TCP-friendly Flows. Identifying flows to regulate is necessary only when congestion becomes important. The most conservative policy is to detect all the flows that are unresponsive or that are using more than what a TCP connection would under the same circumstances. The authors propose a test to find if a flow is or not "TCP-friendly". The test is based on the fact that TCP connections can be characterized by their response to congestion notification.

Unresponsive flows. The previous test is somewhat "TCP-centric", here is a more general test for identifying unresponsive flows. This test is based on the fact that when the steady drop rate increases by a factor x, the offered load for a high-bandwidth flow should decrease by a factor reasonably close to $\sqrt{x}$ (or more).

High-bandwidth flows. In some circumstances, there exist flows that, while being TCP-friendly, use high bandwidth during congestion. The test consists of checking if the flow uses a *disproportionate share* of the bandwidth and comparing its arrival rate with the level of congestion (as indicated by the aggregate packet drop rate).

4.2 Regulating flows

Once non compliant flows have been identified, it is necessary to regulate them. They first have to be classified to be isolated from the rest of the best-effort traffic. This implies to maintain per-flow state for these flows.

The regulation itself can be made simply by a prioritized FIFO scheduler, regulated flows being treated as low priority traffic. Another possibility is to use CBQ (class based queueing) whose mechanisms fit well with the objective. If weighted round robin or weighted fair queueing is used, traffic is separated in two partitions, for restricted and unrestricted traffic. The bandwidth allocation to the partitions must be done in such a way that congestion is always significantly higher in the restricted partition that in the unrestricted one.

5 Drop or Mark?

The RED mechanisms have been designed to be used in an end-to-end congestion control environment, especially TCP. Congestion is detected inside the routers, and is then notified to end hosts by "marking" packets, "marking" meaning using this packet to notify the end host. As TCP congestion control mechanisms have been designed to react to packet drops, "marking" a packet to notify a TCP connection of congestion means dropping it. However, given a transport protocol which is able to recognize a congestion indication piggybacked within a packet, we could avoid dropping the packets, for example by setting a congestion notification bit. This second type of "marking" is called *explicit congestion notification* (ECN).

Using ECN instead of packet drop has some advantages:

- It prevents unnecessary packet drops. Packet drops increase the transmission delay, as the packet has to be re-transmitted after a timer has expired. For bulk data transfer applications, this is not harmful, but for interactive traffic, such as TELNET connections, delay created by packet drops is undesirable. Even in a local area network, where dropped packets should be quickly detected and retransmitted, this is often not the case because of the coarse granularity of the TCP timers.
- It allows a quick and unambiguous congestion notification. With dropped packets, the notification is "received" only after a timer expiration.

On the other hand, ECN has some drawbacks:

- Marked packets could be dropped later in the network before to join the end-host. Thus, the transport protocol would never receive the congestion notification.
- Non-fair applications could pretend being "ECN aware", and then refuse to obey to such congestion notifications. This problem is not specific to ECN, it could also happen with drops but may be more problematic with ECN.

In [1], Floyd discusses the possible inclusion of ECN management in TCP. She also proposes guidelines for TCP's response to ECN, which she explains should not be an exact copy of TCP's response to packet drop.

To conclude, we should note that the two methods could be used at once which is a good point in a heterogeneous network such as the Internet. ECN-aware TCP connections advertise it by setting an "ECN-aware" bit in their packets so that ECN-capable RED routers would mark their packets instead of dropping them. ECN-aware TCP connections would also react to packet drops in the classical way, so that they could have both ECN-capable and non ECN-capable routers in their path.

6 Conclusion

Lately, most of the research work in the field of computer networks has been focused on integrated services. However, best effort traffic constitutes (and will probably still be for years) a large part of the global traffic. An integrated services architecture must provide usable best effort service. As, with the growth of the Internet, more and more applications do not conform to the end-to-end congestion control scheme, ways to encourage them to conform are needed. RED is a solution based on simple mechanisms which can be included in an integrated services architecture. Moreover, it does not require modification of the application thus allowing for progressive deployment.

References

1. Sally Floyd. TCP and Explicit Congestion Notification. Technical report, Lawrence Berkeley Laboratory, Berkeley, CA, 1994.
2. Sally Floyd and Kevin Fall. Router Mechanisms to Support End-to-End Congestion Control. Technical report, Lawrence Berkeley Laboratory, Berkeley, CA, February 1997.
3. Sally Floyd and Van Jacobson. On Traffic Phase Effects in Packet-Switched Gateways. *Internetworking: Research and Experience*, 3(3):115–156, September 1992.
4. Sally Floyd and Van Jacobson. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking*, August 1993.
5. W. Stevens. TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms. RFC 2001, IETF, January 1997.
6. Lixia Zhang and D. Clark. Oscillating Behavior of Network Traffic: A Case Study Simulation. *Internetworking: Research and Experience*, 1(2):101–112, December 1990.