CONNECTOR

Tools Tutorials
Mid Summer 2001 - 7.5

Active Queue Management

by *Kostas Pentikousis*

The previous Connector column (*Principles of Network System Design*) discussed end-to-end protocols and the end-to-end argument. This month's focus will be on the network layer and, in particular, on the queuing methods used by routers. **Routers** are network layer devices used to interconnect different networks [9]. Their primary role is to switch packets from input links to output links. In order to do so a router must be able to determine the path that every incoming packet needs to follow, and decide which outgoing link should it be switched to [6].

Braden *et al.* distinguish between two classes of router algorithms that are related to congestion control, i.e., queue management and scheduling: "To a rough approximation queue management algorithms manage the length of packet queues by dropping packets when necessary or appropriate, while scheduling algorithms determine which packet to send next and are used primarily to manage the allocation of bandwidth among flows" [1]. The most common scheduling algorithm is FIFO (First-In First-Out), which means that the packet that first enters the buffer will be the one to leave the buffer first.

An earlier Crossroads article (*Can TCP be the transport protocol of the 21st century?*) pointed out that packet losses are often assumed to result from congestion, rather than, say, packet corruption. But what exactly is congestion and why does it result in packet loss? Congestion is the phenomenon that occurs at a router when incoming packets arrive at a rate faster than the router can switch (or forward) them to an outgoing link. However, it is important to distinguish contention and congestion. As Peterson and Davie note, "contention occurs when multiple packets have to be queued at a switch (or a router) because they are competing for the same output link, whereas congestion means that the switch has so many packets queued that it runs out of buffer space and has to start dropping packets" [9]. Hence, packet drops can be considered as a form of implicit notification of congestion.

Consider the case of a university campus gateway, a router connecting the campus network to the rest of the Internet (Figure 1). All traffic leaving the campus network must pass through a single router outgoing port. The gateway router can become congested because the campus network provides larger bandwidth than the link connecting the campus with the rest of the Internet.

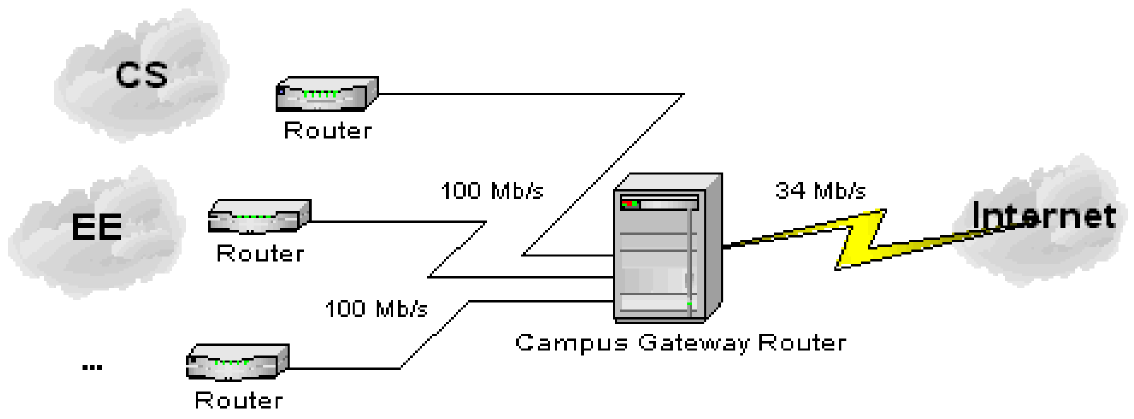


Figure 1 A simplified diagram of a university campus network. Congestion can build up when the aggregate outgoing traffic exceeds the capacity of the link connecting the campus with the rest of the Internet.

Any protocol that runs on top of IP (Internet Protocol), such as TCP (Transmission Control Protocol), can detect packet drops and interpret them as indications of congestion in the network. In particular, a TCP sender will react to these packet drops by reducing its sending rate. This slower sending rate translates into a decrease in the incoming packet rate at the router, which effectively allows the router to clear up its queue.

Queues are used to smooth spikes in incoming packet rates and to allow the router sufficient time for packet transmission. When the incoming packet rate is higher than the router's outgoing packet rate, the queue size will increase, eventually exceeding available buffer space. When the buffer is full, some packets will have to be dropped -- but which ones? A straightforward solution is to drop the packets that are just arriving at the input port; that is, if a packet arrives and finds the queue full it will be dropped. This policy is known as drop tail or tail-drop (DT). Other solutions include dropping the first packet in the queue ("drop front on full") [7] or dropping a random packet already stored in the queue. Each of these solutions has certain advantages and disadvantages (which we will not elaborate on due to space limitations), but DT is the most widespread dropping policy.

Although DT is simple to implement, has been tested and used for many years, it was shown to interact badly with TCP's congestion control mechanisms and to lead to poor performance. For example, studies have shown that DT can cause global synchronization, lockouts, and full queues. Global synchronization occurs when a significant number of TCP senders slows down at the same time and leads to underutilization of scarce network resources (e.g. the bottleneck link). Global synchronization is likely to happen when a router drops many consecutive packets in a short period of time, especially if TCP Tahoe [9] is employed by the sender. Recall that TCP Tahoe always resorts to Slow Start when dropped packets are detected. The Slow Start phase ends when the TCP sender sends more segments than the network can handle. With many senders observing losses during the same period of time, a vicious cycle starts with periods of relatively low network utilization followed by heavy congestion. At the same time, lockouts can happen where a small fraction of flows receive a large proportion of the bottleneck bandwidth, hindering fair allocation and sharing of network resources. Finally, full queues increase per-packet delays and can lead to increased jitter (variation in delay).

Floyd and Jacobson [3] proposed Random Early Detection (RED) gateways back in 1993 in order to solve the above-mentioned problems caused by DT. RED uses randomization to solve both the lockout and full queues problems in an efficient manner, without requiring any changes at the end hosts. Active queue management (AQM) is the pro-active approach of informing the sender about incipient congestion *before* a buffer overflow happens (*cf.* DT) so that senders are informed early on and can react accordingly. The following section is an overview of how RED works.

Random Early Detection

RED is an AQM mechanism, which is not designed to operate with any specific protocol, but performs better with protocols that perceive drops as indications of congestion (e.g. TCP). RED gateways require the user to specify five parameters: the maximum buffer size or queue limit (QL), the minimum (min_{th}) and maximum (max_{th}) thresholds of the "RED region", the maximum dropping probability (max_p), and the weight factor used to calculate the average queue size (w_q). QL can be defined in terms of packets or bytes. Note that when DT is implemented at a router, QL is the only parameter that needs to be set. A RED router uses early packet dropping in an attempt to control the congestion level, limit queuing delays, and avoid buffer overflows. Early packet dropping starts when the *average* queue size exceeds min_{th}. RED was specifically designed to use the average queue size (*avg*), instead of the current queue size, as a measure of incipient congestion, because the latter proves to be rather intolerant of packet bursts, as we will see shortly. If the average queue size does not exceed min_{th} a RED router will not drop any packets. *avg* is calculated as an exponentially weighted moving average using the following formula:

$$avg_i = (1 - w_q) \times avg_{i-1} + w_q \times q$$

where the weight w_q is commonly set to 0.002 [4], and q is the instantaneous queue size. This weighted moving average captures the notion of long-lived congestion better than the instantaneous queue size [9]. Had the instantaneous queue size been used as the metric to determine whether the router is congested, short-lived traffic spikes would lead to early packet drops. So a rather underutilized router that receives a burst of packets can be deemed "congested" if one uses the instantaneous queue size. The average queue size, on the other hand, acts as a low pass filter that allows spikes to go through the router without forcing any packet drops (unless, of course, the burst is larger than the queue limit). The user can configure w_q and min_{th} so that a RED router does not allow short-lived congestion to continue uninterrupted for more than a predetermined amount of time. This functionality allows RED to maintain high throughput and keep per-packet delays low.

```
For every packet arrival {

  Calculate $avg$

  if ($avg \geq max_{th}$) {

      Drop the packet

  }

  else if ($avg > min_{th}$) {

      Calculate the dropping probability $p_a$
```

```
        Drop the packet with probability $p_a$, otherwise forward it
    }
    else {
        Forward the packet
    }
}
```

Listing 1 Pseudocode of the RED algorithm.

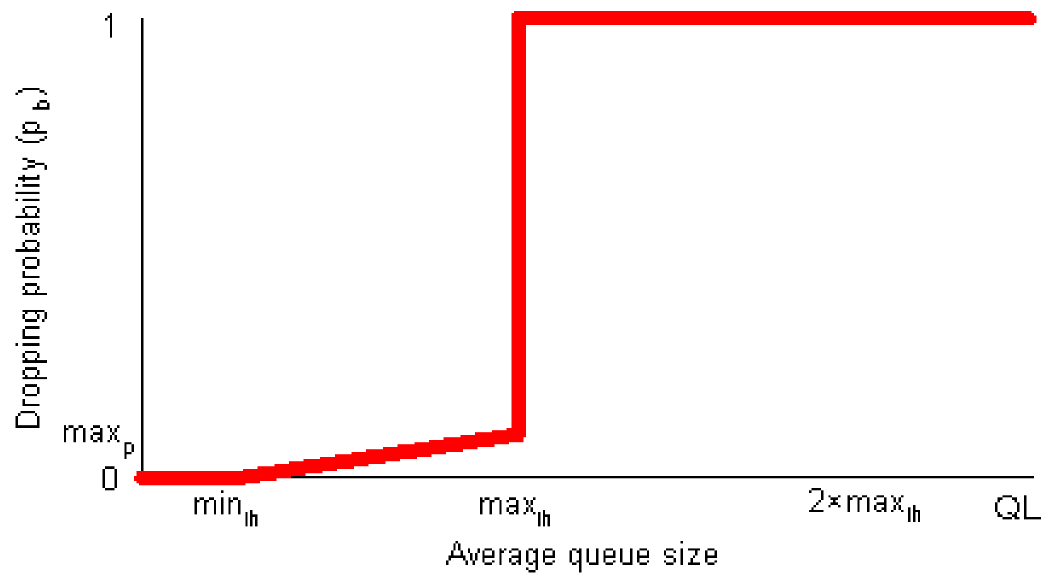
An interesting point about the RED algorithm (Listing 1) is that it separates the decision processes of *when* to drop a packet from *which* packet to drop. As noted earlier, RED uses randomization to decide which packet to drop and, consequently, which connection will be notified to slow down. This is accomplished by making Bernoulli trials (e.g. coin flips) using a probability p_a, which is calculated according to the following formulae:

$$p_b = max_p \times (avg - min_{th}) / (max_{th} - min_{th})$$

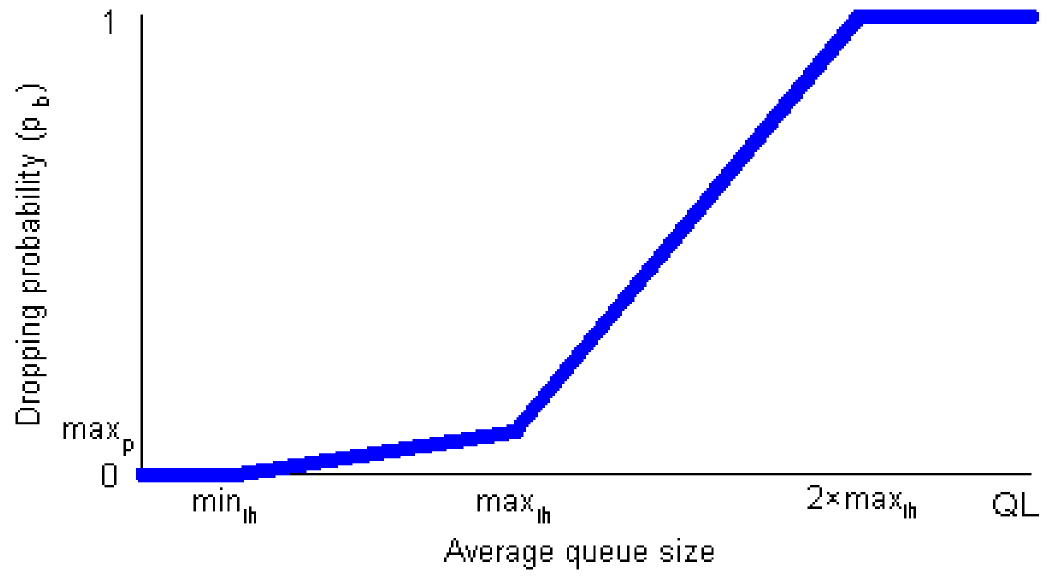
and

$$p_a = p_b / (1 - count \times p_b)$$

where max_p is a user-defined parameter, usually set to 2% or 10% [4], and *count* is the number of packets since the last packet drop (but see also [2]). *count* is used so that consecutive drops are spaced out over time. Notice that p_b varies linearly between 0 and max_p, while p_a, i.e., the actual packet dropping probability increases with count. Originally, max_{th} was defined as the upper threshold; when the average queue size exceeds this limit, *all* packets have to be dropped (Figure 2(a)). Later on, however, Floyd proposed a modification to the dropping algorithm (the "gentle option"), under which packets are dropped with a linearly increasing probability until *avg* exceeds $2 \times max_{th}$; after that all packets are dropped (Figure 2(b)). Although max_{th} can be set to any value, a rule of thumb is to set it to three times min_{th}, and less than QL [4].



(a) RED



(b) RED with the "gentle option"

Figure 2 The packet dropping probability (p_b) in RED as a function of the average queue size (max_p = 10%).

By dropping packets before the buffer overflows, RED attempts to notify some connections of incipient congestion. The responsive ones will limit their sending rates and eventually the network load will decrease. The unresponsive connections will not slow down, but will continue at the same pace or even increase their sending rates. In this case, the unresponsive flows will have more packets reaching the router, effectively providing more candidates for dropping than responsive ones.

Fairness is an important feature of any AQM mechanism, especially in the current Internet incarnation where all connections are treated equally. Clearly, in times of congestion, one would like to notify those connections that use most of the resources. In other words, the router should drop packets that belong to flows currently using most of the bandwidth (and buffer space). RED does not keep per-connection state, i.e., exact statistics of resource utilization (see also [8]), so it cannot be absolutely accurate in dropping packets according to the exact proportion of the resources that a connection is using. Nevertheless, its inventors show that the number of packets that RED drops from a sinlge flow is roughly proportional to its share of the bandwidth, and that it can be effective in notifying the connections with the higher percentage of bandwidth utilization. In contrast, DT tends to cause packet drops in batches without taking into account the proportion of the resources a flow is using [3].

To be continued...

RED can easily be deployed in the Internet because it is supported by router vendors and is implemented in a number of products [5]. Nonetheless, RED is not currently widely used, despite the eight years that have passed since its introduction, the subsequent studies of its efficiency in controlling congestion, and the IETF (Internet Engineering Task Force) recommendation for the wide deployment of RED some years ago [1]. The next Connector column will discuss the advantages and pitfalls of RED, some alternatives, and the potential for better Internet services that active queue management encompasses.

References

1
Braden, B., D. Clark, J. Crowcroft, B. Davie, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang, *Recommendations on Queue Management and Congestion Avoidance*, RFC 2309, April 1998.

2
De Cnodder, S., Elloumi, O., and Pauwels, K, "RED Behavior with Different Packet Sizes", *Proceedings of the Fifth IEEE Symposium on Computers and Communications*, Antibes-Juan les Pins, France, July 2000.

3
Floyd, S., and Jacobson, V., "Random Early Detection Gateways for Congestion Avoidance". In *ACM/IEEE Transactions on Networking*g, 3(1), August 1993.

4
Floyd, S., *RED: Discussions of Setting Parameters*, November 1997, http://www.aciri.org/floyd/REDparameters.txt (June 2001).

5
Floyd, S., *RED Implementations*, http://www.aciri.org/floyd/red.html#implementations, (June 2001).

6
Kurose, J., and Ross, K., *Computer Networking: A Top-down Approach Featuring the Internet*, Addison Wesley, 2001.

7
Lakshman, T. V., Neidhardt, A., and Ott, T., "The Drop From Front Strategy in TCP Over ATM and its Interworking with Other Control Features", *Proceedings of IEEE INFOCOM 1996*, San Francisco, California, April 1996.

8
Lin, D. and Morris, R., "Dynamics of Random Early Detection", *Proceedings of ACM SIGCOMM '97*, pages 127-137, Cannes, France, October 1997.

9
Peterson, L. L., and Davie, B. S., *Computer Networks, 2nd edition*, Morgan-Kaufmann, 2000.