1 RED

1.1 О чем это?

RED (Random Early Detection - Произвольное раннее обнаружение) – Алгоритм активного управления очередью для управления переполнением очередей маршрутизаторов, с возможность предотвращения перегрузок.

Вероятность p_b маркировки на отбрасывание пакетов представляет собой функцию, линейно зависящую от $\hat{q}$, минимального q_{min} и максимального q_{max} пороговых значений и параметра p_{max}, определяющего часть отбрасываемых пакетов при достижении средним размером очереди значения q_{max} и вычисляется следующим образом:

$$p_b = \begin{cases} 0, & 0 < \hat{q} \leq q_{min} \\ 1, & \hat{q} > q_{max} \\ \frac{\hat{q} - q_{min}}{q_{max} - q_{min}} p_{max}, & q_{min} < \hat{q} \leqslant q_{max} \end{cases}$$

1.2 Разбор алгоритм работы RED

Пакет при поступлении в систему попадает в модуль сброса. Решение о сбросе пакета принимается на основе значения вероятности p, получаемого от управляющего модуля. Вероятность p сброса пакетов зависит от экспоненциально взвешенного скользящего среднего размера длины очереди $\hat{q}$, также вычисляемого управляющим модулем, основы- ваясь на текущем значении длины очереди q.

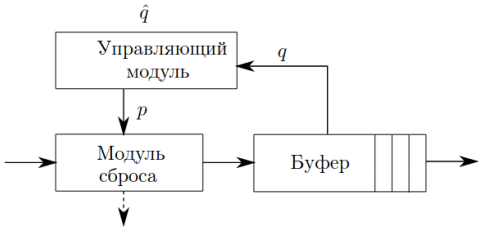

Figure 1: Модуль RED

Вероятность потери пакета в зависимости от среднего размера очереди будет выглядеть следующим образом:

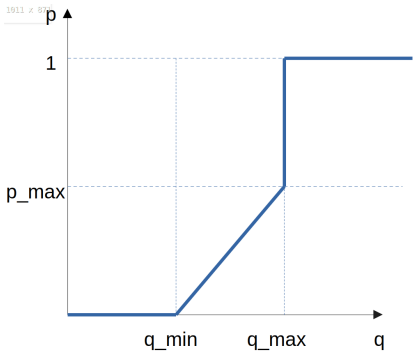

Figure 2: RED

1.3 Разбор реализации в NS2

Файлы, связанные с RED находятся по пути `ns-2.35/queue`, там представлены различные реализации очередей (среди них DropTail, RED и т.д.). Для нас важны два файла:

- `red.h` – Заголовочный файл
- `red.cc` – Исходники

В файле red.cc для нас интересна функция `REDQueue::estimator` и функция `double REDQueue::calculate_p_new`. Первая функция отвечает за расчет средней длины очереди. Вторая – расчитывает вероятность потери пакета.

Разберем исходный код второй функции:

```
double
REDQueue::calculate_p_new(double v_ave, double th_max, int gentle, double v_a,
    double v_b, double v_c, double v_d, double max_p)
{
    double p;
    if (gentle && v_ave >= th_max) {
        // Необходимо для GRED (подробнее о GRED ниже)

        // p находится в промежутке от max_p до 1,
```

```
        // тогда как средний размер очереди в промежутке
        // q_max и 2*q_max
        p = v_c * v_ave + v_d;
        } else if (!gentle && v_ave >= th_max) {
        // Превысили пороговое значение в классическом RED
        // p приравниваем к 1
                p = 1.0;
        } else {
        // p в промежутке от 0 до max_p, тогда как
        // средний размер очереди в промежутке
        // th_min до th_max
                p = v_a * v_ave + v_b;
                // p = (v_ave - th_min) / (th_max - th_min)

                p *= max_p;
        }
    if (p > 1.0)
        p = 1.0;
    return p;
}
```

1.4 Проблемы RED

Одна из фундаментальных проблем RED заключается в том, что он полагается на длину очереди в качестве показателя загруженности. Хотя наличие постоянной очереди указывает на перегрузку, ее длина дает очень мало информации о серьезности перегрузки.

Поскольку алгоритм RED зависит от длины очереди, ему присуща проблема определения степени перегрузки. В результате RED требует широкого диапазона параметров для корректной работы в различных сценариях перегрузки. Хотя RED может достичь идеальной рабочей точки, он может сделать это только при наличии достаточного объема буферного пространства и правильных параметров.

2 WRED

2.1 О чем это?

WRED (Weighted random early detection - Взвешенное произвольное раннее обнаружение) – Алгоритм активного управления очередью, является расширением RED.

Взвешенный алгоритм произвольного раннего обнаружения предоставляет различные уровни обслуживания пакетов в зависимости от вероятности их отбрасывания и обеспечивает избирательную установку параметров механизма RED на основании типа трафика.

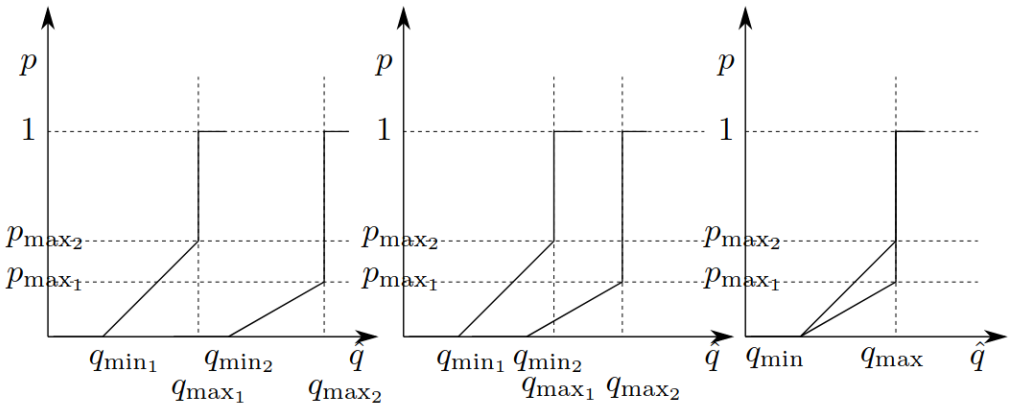

Figure 3: WRED

Алгоритм WRED работает с единой очередью пакетов, для которой, как и в RED, по формуле рассчитывается экспоненциально взвешенное скользящее среднее. Для каждого типа трафика задаются собственные параметры (пороговые значения, максимальный уровень сброса) и вычисляется вероятность сброса.

Например, очереди могут иметь более низкие пороговые значения для более низких приоритетов пакета. Это приведет к отбрасыванию пакетов с низким приоритетом, а следовательно, к защите пакетов с более высоким приоритетом в той же очереди.

3 GRED

GRED (Gentle random early detection - мягкое/аккуратное произвольное раннее обнаружение) – Алгоритм активного управления очередью, является расширением RED.

Gentle RED расширяет RED тем, что добавляет дополнительное максимальное пороговое значние, которое равно $2 * q_{max}$, тем самым “сглаживая” кривую.

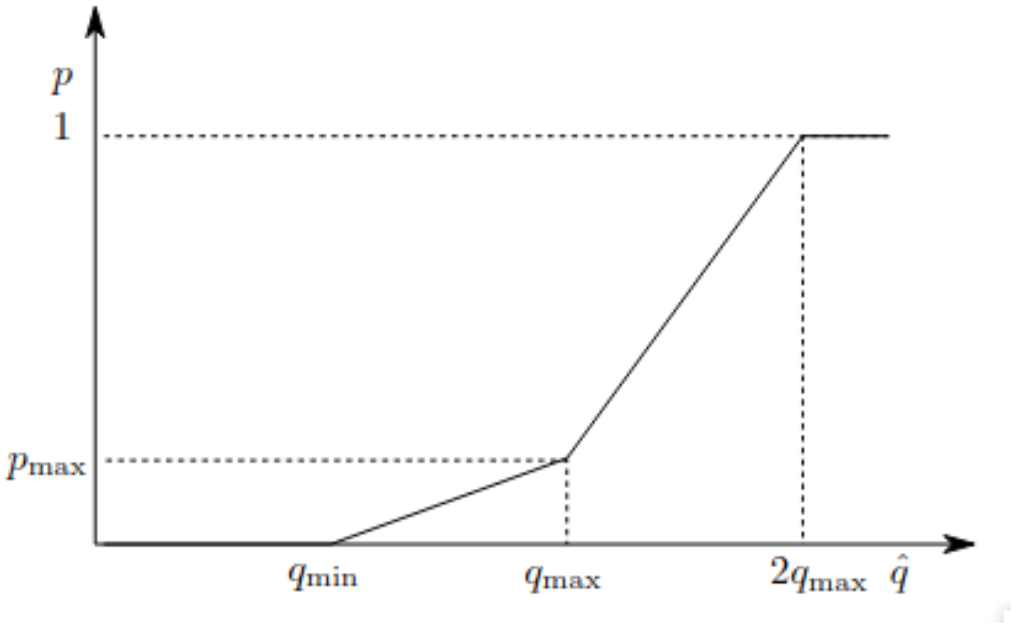

Figure 4: GRED

Вычисляется следующим образом:

$$p_b = \begin{cases} 0, & 0 < \hat{q} \leq q_{min} \\ \frac{\hat{q}-q_{min}}{q_{max}-q_{min}} p_{max}, & q_{min} \leqslant \hat{q} < q_{max} \\ \frac{\hat{q}-q_{min}}{q_{max}}(1-p_{max}) - p_{max}, & q_{max} \leqslant \hat{q} < 2q_{max} \\ 1, & \hat{q} \geqslant q_{max} \end{cases}$$