Отчёта по практической работе

Неделя № 3

Саргсян Арам

Содержание

1 Список сокращений

Англоязычные сокращения

- RED — Random early detection
- GRED — Gentle random early detection
- WRED — Weighted random early detection
- TCP — Transmission control protocol

2 Классический RED

2.1 Теоритический материал

RED — 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:

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

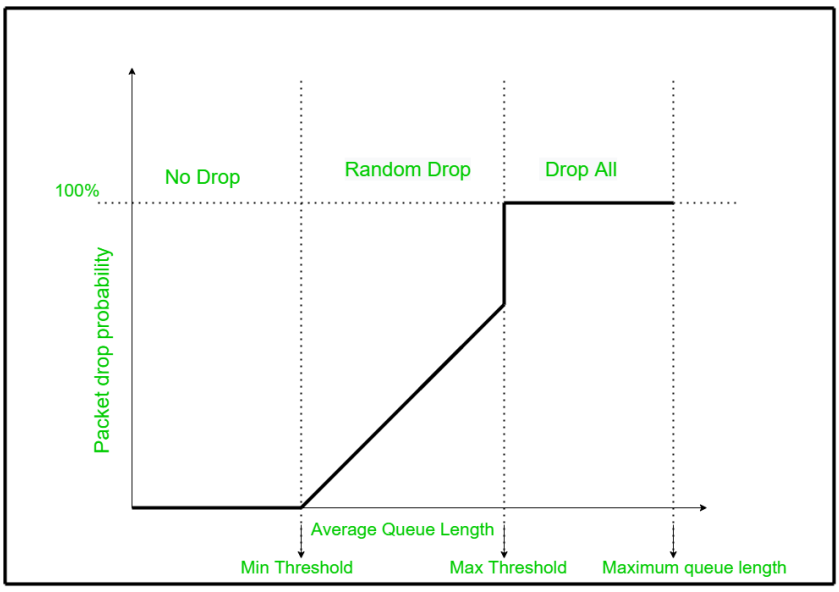

Рис. 2.1: классический RED

2.2 Реализация в NS-2

Вот пример реализации параметров RED в NS-2:

```
# Мониторинг очереди:
set redq [[$ns link $node_(r0) $node_(r1)] queue]
$redq set thresh_ 75 #q_min
$redq set maxthresh_ 150  # q_max
$redq set q_weight_ 0.002 # q_weight
$redq set linterm_ 10 # 1/p_max
$redq set drop-tail_ true # вместо механизма randomdrop используется drop-
tail в случае переполнения очереди или когда средний размер очереди больше maxthr
set tchan_ [open output/all.q w]
$redq trace curq_ # текущий размер очереди
$redq trace ave_ # средний размер очереди
$redq attach $tchan_
```

В NS-2 параметры RED Файлы,указываются в каталоге ns-2.35/queue, там представлены также другие реализации очередей (среди них DropTail, BLUE и т.д.). Вероятность отбрасывания пакета прописана в функции double REDQueue::calculate_p_ne файла red.cc

```
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) { //для модификации GRED
        p = v_c * v_ave + v_d;
        } else if (!gentle && v_ave >= th_max) { // Превысили пороговое значение
                p = 1.0;
        } else { //p в промежутке от 0 до max_p, тогда средний размер очереди в п
                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;
}
```

3 GRED

3.1 Теоритический материал

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 $2q_{max}$, тем самым “сглаживая” кривую.

Вычисляется следующим образом:

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

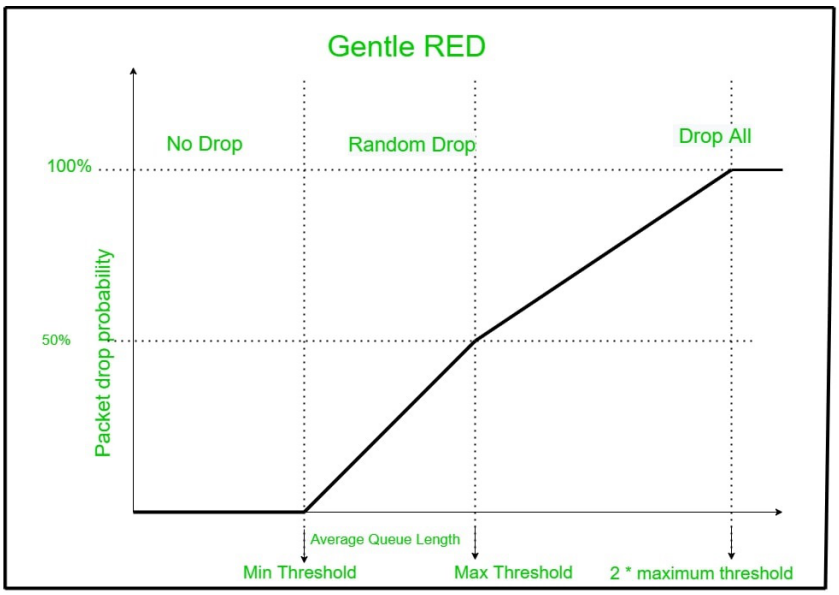

Рис. 3.1: Gentle RED

3.2 Реализация в NS-2

Для реализации модификации в мониторинге очереди нужно прописать в мониторинге очереди следующую строку:

```
$redq set gentle_ true
```

При этом случае в функции double REDQueue::calculate_p_ne файла red.cc переменная gentle примет значение true и

4 WRED

4.1 Теоритический материал

WRED — алгоритм активного управления очередью, является расширением RED. WRED действует в следующем порядке, когда одна очередь может иметь несколько разных наборов порогов очереди. Каждый набор пороговых значений связан с определенным классом трафика. Например, очередь может иметь более низкие пороги для пакетов с более низким приоритетом. Наращивание очереди приведет к отбрасыванию пакетов с более низким приоритетом, тем самым защищая пакеты с более высоким приоритетом в той же очереди.

4.2 Реализация в NS-2

WRED не реализована в классической версии NS-2. Для ее реализации необходимо создать новый класс, который должен наследоваться от базового класса REDQueue, используемого для реализации стандартной версии алгоритма RED в NS-2. После необходимо реализовать алгоритм модификации в методе dropEarly(), который вызывается, когда очередь превышает пороговое значение. Этот метод должен определять, какие пакеты должны быть отброшены, и какая вероятность отбрасывания пакетов должна быть применена.