Отчёта по практической работе

Неделя № 3

Саргсян Арам

Содержание

# 1 Список сокращений

**Англоязычные сокращения**

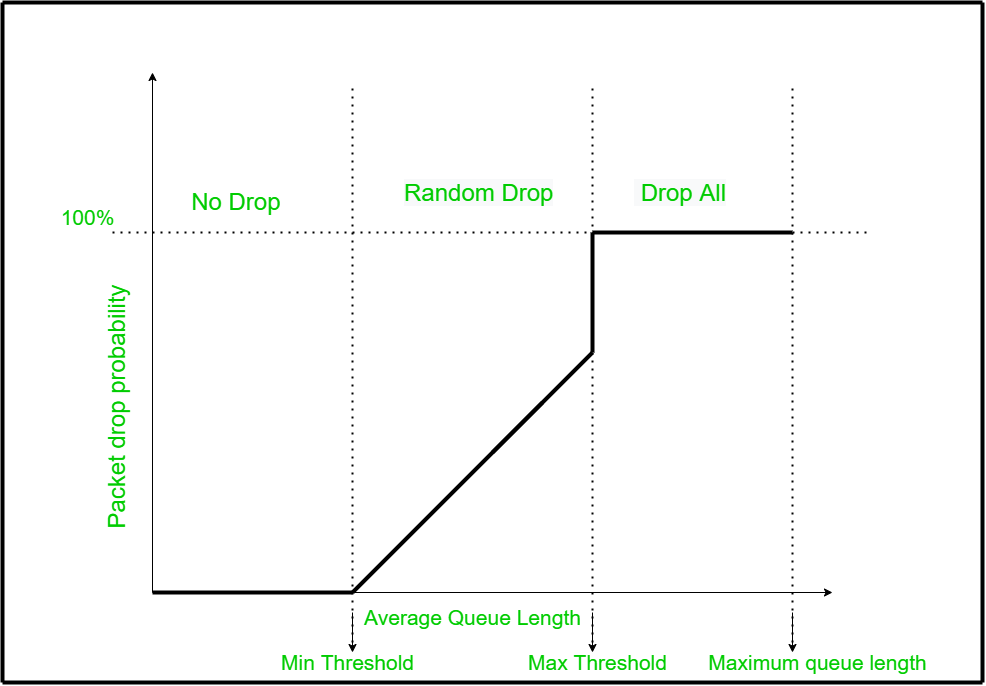
* RED — Random early detection
* GRED — Gentle random early detection
* WRED — Weighted random early detection
* TCP — Transmission control protocol

# 2 Классический RED

## 2.1 Теоритический материал

RED — алгоритм активного управления очередью для управления переполнением очередей маршрутизаторов, с возможность предотвращения перегрузок.

Вероятность маркировки на отбрасывание пакетов представляет собой функцию, линейно зависящую от , минимального и максимального пороговых значений и параметра , определяющего часть отбрасываемых пакетов при достижении средним размером очереди значения и вычисляется следующим образом:



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 maxthresh\_  
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

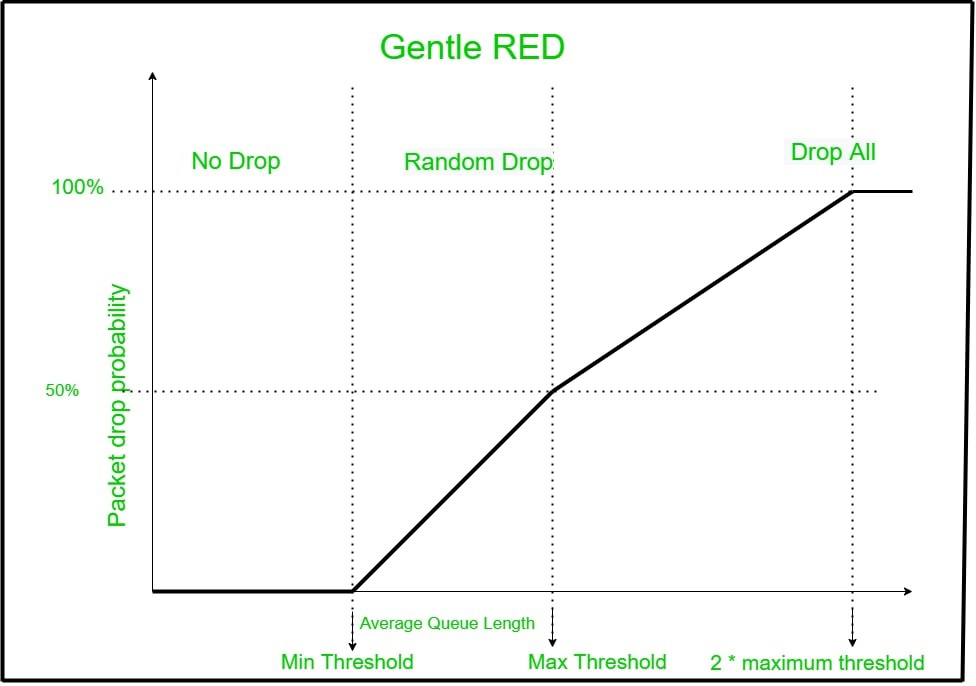
double  
REDQueue::calculate\_p\_new(double v\_ave, double th\_max, int gentle, double v\_a,   
 double v\_b, double v\_c, double v\_d, double max\_p)  
{  
 double p;  
 if (gentle && v\_ave >= th\_max) { //для модификации GRED  
 p = v\_c \* v\_ave + v\_d;  
 } else if (!gentle && v\_ave >= th\_max) { // Превысили пороговое значение в классическом RED  
 p = 1.0;  
 } else { //p в промежутке от 0 до max\_p, тогда 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;  
}

# 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 , тем самым “сглаживая” кривую.

Вычисляется следующим образом:



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.