УДК 004.77

АЛГОРИТМ СЕТЕВОГО ПЛАНИРОВЩИКА НА БАЗЕ КОМПОНЕНТОВ С ОТКРЫТЫМ ИСХОДНЫМ КОДОМ

Кочнева О.Р.[1]
Научный руководитель – к.т.н. Соснин В.В.[2]
[1]Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)
[2]ООО “Техкомпания Хуавэй”

Аннотация
В работе предложен новый алгоритм, позволяющий гарантировать минимальную пропускную способность, распределять неиспользованную пропускную способность, а также ограничивать сверху пропускную способность различным классам трафика на основе проприетарного планировщика и компонентов с открытым исходным кодом.

Ключевые слова
QoS, планирование, алгоритм, пропускная способность, Linux, открытый исходный код.

Введение.

Трудно представить жизнь без видеозвонков, использования облачных хранилищ и различных стриминговых сервисов. Поэтому задача обеспечения качества обслуживания (QoS) в сети является как никогда актуальной. Отсутствие настройки сетевого планировщика или его неправильная конфигурация может отразиться на качестве соединения, что особенно ощутимо в высоконагруженных системах. Существует множество, как бесплатных, так и платных планировщиков сетевых пакетов. Наибольший интерес для QoS представляют классовые иерархические планировщики, которые позволяют контролировать определенные типы трафика и распределять сетевые ресурсы в соответствии с их нуждами. Не все существующие решения поддерживают иерархию классов трафика и помимо этого обладают еще рядом недостатков, таких как невозможность гарантировать минимальную пропускную способность, ограничивать сверху пропускную способность или распределять пропускную способность в соответствии с весом класса, либо их интеграция в операционную систему связана с большими накладными расходами. Ни один из доступных планировщиков не удовлетворяет всем этим критериям одновременно в явном виде. В работе предложен алгоритм сетевого планировщика с предъявленными свойствами на основе базового функционала планировщиков HFSC и HTB.

Обзор предметной области.

В обзоре рассмотрены планировщики сетевых пакетов, представленные в открытом доступе и бесплатные для использования в ОС Linux, а также платные альтернативы.

Планировщики на основе веса трафика:

1. SFQ (Stochastic Fairness Queuing) – бесклассовая дисциплина планирования очередей. SFQ позволяет избежать ситуации “вечного” простоя потока трафика и таким образом является представителем честного планировщика.
2. DRR (Deficit Round Robin) – классовая дисциплина обслуживания, является более гибкой заменой SFQ.
3. WFQ (Weighted Fair Queuing) и CBWFQ (Class Based WFQ) – обеспечивают честное разделение полосы пропускания в зависимости от веса потока трафика. В случае перегрузок высокоприоритетные потоки функционируют без изменений, а низкоприоритетные ограничиваются.

HFSC (Hierarchical Fair Service Curve)

HFSC - иерархический планировщик пакетов, основанный на кривых обслуживания. HFSC позволяет предоставить конкретную пропускную способность для листовых узлов в иерархии, распределить пропускную способность между классами согласно их весу, а также ограничить класс трафика сверху с использованием управляющих параметров: ls (linkshare, бит/с) – разделение канала между классами трафика, ul (upper limit, бит/с) – верхняя граница выделенной пропускной способности канала для класса трафика;

HTB (Hierarchical Token Bucket)

HTB - иерархический планировщик, позволяющий контролировать использование исходящей полосы пропускания по заданному каналу с использованием следующих управляющих параметров: ceil (бит/с) для ограничения пропускной способности сверху и rate (бит/с) для перераспределения оставшейся пропускной способности.

Планировщик пакетов в гипервизоре VMware - hClock

Сетевой планировщик hClock, встроенный в гипервизор VMware, обладает следующими возможностями: резервирование минимальной пропускной способности (параметр reservations – R, бит/с), ограничение пропускной способности сверху (параметр Limit – L, бит/с) и перераспределение оставшейся пропускной способности (безразмерный параметр Shares – S). В hClock распределение оставшейся пропускной происходит с использованием одной из семантик SUM (R, S) или MAX (R, S) [1].

- MAX (R, S): распределение пропускной способности происходит в соответствии с пропорциями, указанными в параметре S с учетом ограничений, задаваемых параметрами R и L.
- SUM (R, S): распределение пропускной способности происходит в соответствии с параметром R, а оставшаяся пропускная способность после того, как все узлы получили свой минимум, распределяется на основе S с учетом верхней границы L.

Патент: метод для контроля пропускной способности классов (BW management system)

Система BW management system включает в себя модуль управления BW для управления пропускной способностью потоков, организованных в иерархическое дерево полосы пропускания (Hierarchical Binary Tree - HBT). Система обеспечивает возможность резервирования пропускной способности, установки верхней границы и взвешенного распределения оставшейся пропускной способности [2].

Алгоритм планирования SG-QoS

Алгоритм SG-QoS – это алгоритм планирования, основанный на кривых обслуживания. Он разрабатывался для эффективной поддержки приложений с гарантией задержки, приложений с гарантией IOPS (input/output operations per second) и приложений без требований качеству обслуживания (best-effort applications). Алгоритм SG-QoS более эффективен, чем HFSC в облачных системах хранения [3].

Сравнительный анализ рассмотренных планировщиков

Для проведения сравнения были выделены следующие критерии:

1. Доступность реализации: открытый исходный код или бесплатное использование.
2. Простота интеграции в ОС Linux.
3. Поддержка иерархических конфигураций.
4. Гарантия минимальной пропускной способности (Reservations).
5. Взвешенное перераспределение трафика (Shares).
6. Ограничение пропускной способности сверху (Limit).
7. Наличие контроля доступа (admission control).

Таблица 1. Сравнение аналогов.

Критерий	SFQ	DRR, QFQ	HTB	HFSC	WFQ, CBWFQ	hClock	BW	SG-QoS
1. Доступность	+	+	+	+	+	-	-	-
2. Простота интеграции	+	+	+	+	-	-	-	-
3. Иерархия	-	+	+	+	-	+	+	+
4. Резервирование	-	-	-	-	-	+	+	-
5. Вес	-	+	+	+	+	+	+	+
6. Ограничение	-	-	+	+	-	+	+	+
7. Контроль доступа	-	-	-	-	-	+	+	-

На основе проведенного анализа наибольшему количеству критериев удовлетворяют алгоритмы hClock и BW-management и из алгоритмов, находящихся в открытом доступе и легко интегрируемых в ОС Linux HTB и HFSC.

Для устранения недостатков HFSC и HTB в настоящей работе предлагается новое решение, реализующее для этих дисциплин механизм контроля доступа и управления пропускной способностью, аналогичных механизмам в hClock, и не требующее перекомпиляции модулей ядра Linux. Предложенные новые алгоритмы будем назвать соответственно семантикам hClock следующим образом: HFSC-RLS-SUM, HFSC-RLS-MAX, HTB-RLS-SUM и HTB-RLS-MAX.

Описание алгоритма

Описание алгоритма будет строиться с использованием параметров для планировщика HFSC.

Обозначения и сокращения:

- n = количество классов;
- CT = пропускная способность;
- UT = нераспределенная пропускная способность;
- AC = контроль доступа;
- $R = \{r_i\}_{i\in[1,n]}$ – множество параметров reservations;
- $L = \{l_i\}_{i\in[1,n]}$ – множество параметров limit;
- $S = \{s_i\}_{i\in[1,n]}$ – множество параметров shares;
- $UL = \{ul_i\}_{i\in[1,n]}$ – множество параметров ul для HFSC или ceil для HTB ;
- $LS = \{ls_i\}_{i\in[1,n]}$ – множество параметров ls для HFSC или rate для HTB.

Общая идея алгоритма.

1. Проверить правила контроля доступа для входных параметров R, L.
2. Отобразить параметры R, L, S на выходные параметры LS, UL.

Отображение параметров:

Параметры R, L, S класса трафика отображаются на параметры планировщиков HTB и HFSC. Распределение оставшейся пропускной способности осуществляется с использованием SUM-подобной или MAX-подобной семантики по формулам (1) и (2) соответственно, при условии $l_i = CT,\ i \in [1, n]$.

$$\boldsymbol{SUM}: \boldsymbol{ls_i = r_i + \frac{s_i}{\sum_{j=1}^{n} s_j} * (CT - \sum_{k=1}^{n} r_k)}. \quad (1)$$

$$\boldsymbol{MAX}: \boldsymbol{ls_i = max\ (r_i,\ \frac{s_i}{\sum_{j=1}^{n} s_j} * CT)}. \quad (2)$$

Псевдокод алгоритма HFSC-RLS-SUM в общем виде для произвольного L:

1. for ($i = 1$; $i \leq n$; i ++) do
2. $ul_i = l_i, ls_i = r_i$
3. while ($UT \neq 0$ *or* $LS \neq UL$) do
4. $UT = CT - \sum_{i=1}^{n} ls_i$
5. $SS = \sum_{i=1}^{n} s_i \ , ul_i \neq ls_i$
6. for (i = 1; $i \leq n$; i ++) do
7. if $ul_i == ls_i$ then continue
8. $ls_i = ls_i + \frac{UT * s_i}{SS}$
9. $ls_i = \min(ul_i, ls_i)$
10. return *LS*, *UL*

Псевдокод алгоритма HFSC-RLS-MAX в общем виде для произвольного L. В алгоритм введена дополнительная переменная-маркер is_set_i, определяющая состояние класса i (вычислен ls_i для класса или нет) и *ST* отображающая оставшуюся пропускную способность, которая используется для проверки окончания алгоритма, в отличие от *UT*, которая используется для вычисления *LS*:

1. if $\sum_{i=1}^{n} l_i < CT$ return *LS* = *L*, *UL*=*L*
2. $ul_i = l_i, ls_i = 0, is\_set_i = False,$ where $i \in [1, n]$
3. while (*True*) do
4. *UT* = *CT*, *ST* = *UT*, *SS* = 0
5. $ST = CT - \sum_{i=1}^{n} ls_i$
6. $UT = CT - \sum_{i=1}^{n} ls_i$, where is_set_i
7. $SS = \sum_{i=1}^{n} s_i$, where not is_set_i
8. Do rounding LS
9. if $ST == 0$ then break
10. for (i = 1; $i \leq n$; i ++) do
11. if is_set_i continue
12. $p = \frac{UT * s_i}{SS}$
13. if $p \leq r_i$ then
14. $ls_i = r_i, is\_set_i = True$
15. $ls_i = 0, is\_set_i = False,$ where $ls_i \neq r_i, \ i \in [1, n]$
16. break
17. else $ls_i = \min(p + 1, ul_i)$
18. return *LS*, *UL*

Тестирование алгоритма

Для тестирования работы алгоритма использовались 2 виртуальные машины с Ubuntu Server 18.04 в VirtualBox с настроенными сетевыми интерфейсами (см. рис. 1):

- enp0s8 - виртуальный адаптер хоста для связи с виртуальными машинами серверами через ssh. Ограничений на пропускную способность - нет.
- enp0s9 - внутренняя сеть для взаимодействия виртуальных машин между собой. Машинам присвоены статические адреса. Есть ограничения на пропускную способность в размере 100 Мбит/с.

Для эмуляции трафика и сбора логов о распределении пропускной способности использовалась утилита iperf3. Для конфигурирования дисциплины планирования HFSC - утилита tc.

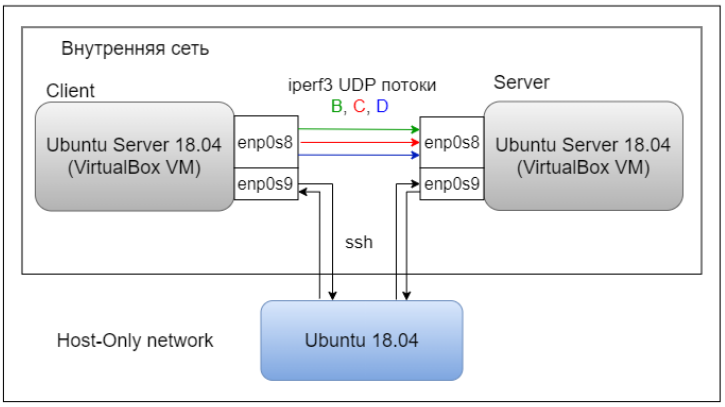

Рисунок 1. Схема экспериментальной установки.

В качестве примера работы алгоритма использовалась следующая иерархия классов (см. рис. 2).

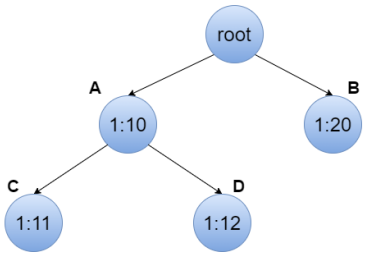

Рисунок 2. Иерархия классов.

Положим, что классы A и B должны делить пропускную способность между собой в отношении 1:1, без ограничений сверху. А дочерние классы C и D должны делить пропускную способность, выделенную классу A в отношении 1:5, причем классу C должно гарантированно быть выделено 20 Мбит/с. В табл. 2 и табл. 3 представлены параметры R, L, S для данной конфигурации и отображённые значения ul, ls для HFSC. Ожидаемые для hClock значения скорости приема RX потоков на стороне Server при условии передачи насыщенных потоков со стороны Client вычислены вручную по формулам (1) и (2) соответственно. Фактические значения получены с помощью применения алгоритма HFSC-RLS-SUM и HFSC-RLS-MAX. Общая пропускная способность составляет 100 Мбит/с.

Таблица 2. Распределение пропускной способности HFSC-RLS-SUM

Поток трафика		**hClock-параметры (Мбит/с)**			**HFSC-параметры (Мбит/с)**		**RX расчетная для сценария (Мбит/с)**		**RX фактическая для сценария (Мбит/с)**	
		R	L	S	ls	ul	B + C + D	C + D	B + C + D	C + D
A	1:10	20	-	1	60	100	-	-	-	-
B	1:20	0	-	1	40	100	40	-	38.9 ± 0.1	-
C	1:11	20	-	1	26	100	26	33	25.3 ± 0.1	42.1 ± 0.1
D	1:12	0	-	5	34	100	34	67	33 ± 0.1	55.1 ± 0.1

Таблица 3. Распределение пропускной способности HFSC-RLS-MAX

Поток трафика	**hClock-параметры (Мбит/с)**			**HFSC-параметры (Мбит/с)**		**RX расчетная для сценария (Мбит/с)**		**RX фактическая для сценария (Мбит/с)**	
	R	L	S	ls	ul	B + C + D	C + D	B + C + D	C + D

A	1:10	20	-	1	50	100	-	-	-	-
B	1:20	0	-	1	50	100	50	-	48.6 ± 0.1	-
C	1:11	20	-	1	20	100	20	20	19.4 ± 0.1	38.9 ± 0.1
D	1:12	0	-	5	30	100	30	80	29.2 ± 0.1	58.3 ± 0.1

Результат, полученный конфигурацией HFSC параметрами, вычисленными алгоритмом, совпадает с результатом работы базового алгоритма hClock при полной загруженности сети. При отсутствии потока, или потоков, трафика распределение пропускной способности может значительно отличаться от расчетной, так для семантики SUM в табл. 2 различие при существовании 2-х из 3-х потоков трафика составляет 36 % и 18 %. Для MAX семантики ожидаемые значения для потоков C и D отличаются от полученных на 100 % и 25 % соответственно. Это объясняется тем, что в HFSC передается 2 параметра, на которые отображены 3 параметра. Точная оценка отличий расчетных значений от фактических на данном этапе проведена не была. Несмотря на то, что различия могут быть существенны, это не влияет на количество зарезервированной пропускной способности и ее ограничении сверху, различия только в распределении оставшейся пропускной способности. Классу трафика гарантированно предоставится пропускной способности не меньше, чем было при насыщенном трафике у всех классов. Данное поведение является особенностью реализованного алгоритма.

Заключение.

Разработанное решение позволяет осуществлять распределение пропускной способности между классами трафика на основе параметров R-reservations, L-limit, S-shares, реализуя SUM-подобную и MAX-подобную семантику, и обладает следующими свойствами:

- распределение пропускной способности аналогично распределению, представленному в алгоритме hClock, если все классифицированные потоки насыщенные или при отсутствии резервирования;
- простота интеграции в ОС Linux, не требующая перекомпиляции или реконфигурирования ядра;
- наличие GPL-лицензии.

Ограничением алгоритма является необходимость в указании пропускной способности сети. Все материалы доступны в открытом виде в репозитории [4]. Разработка может применяться в гипервизорах с открытым исходным кодом (для управления трафиком виртуальных машин), программных маршрутизаторах (для управления трафиком абонентов), Android-устройствах и любых других устройствах, ОС которых основана на ядре Linux (для управления трафиком приложений).

Литература

1. Billaud J. P., Gulati A. hClock: Hierarchical QoS for packet scheduling in a hypervisor //Proceedings of the 8th ACM European Conference on Computer Systems. – 2013. – С. 309-322.
2. Prakash P. et al. System and method to control bandwidth of classes of network traffic using bandwidth limits and reservations : пат. 9929962 США. – 2018.
3. Zhang Y. et al. Dynamic scheduling with service curve for QoS guarantee of large-scale cloud storage //IEEE Transactions on Computers. – 2017. – Т. 67. – №. 4. – С. 457-468.
4. Репозиторий проекта [Электронный ресурс]. Режим доступа: ttps://github.com/OlgaKochneva/rls-network-schedulers (дата обращения: 14.04.2021).