Міністерство освіти і науки України

Тернопільський національний педагогічний університет імені Володимира Гнатюка

Кафедра інформатики

ІНДЗ

Управління трафіком ОС Linux засобами утиліти TC

студента групи І-24

фізико-математичного факультету

Романця Арсена

Науковий керівник: Олексюк Василь Петрович

Тернопіль – 2013

Зміст:

Вступ

Мета: Освоїти основи управління трафіком в ОС Linux, ознайомитися з утилітою TC, а також з її допомогою обмежити швидкість трафіку на одному з мережевих інтерфейсів системи.

Поділ, обмеження і керування трафіком - актуальна і складна задача, вирішити яку можна і за допомогою підсистеми Linux-ядра TrafficControl. Підсистема TrafficControl підтримує безліч методів класифікації, пріоритезації, поділу та обмеження трафіку (як вихідного, так і вхідного). Вона дуже гнучка в налаштуванні, однак складна в реалізації.

Управління трафіком- це термін, об'єднуючий системи обробки черг і функції прийому / передачі пакетів в маршрутизаторі. Вони включають в себе механізми прийняття рішень, які пакети приймати і з якою швидкістю на вхідному інтерфейсі, визначення пакетів для передачі, їх порядку і швидкість передачі на вихідному пристрої.

Управління трафіком складається з набору механізмів і операцій за допомогою яких пакети організовуються в чергу для прийому / передачі через мережевий інтерфейс.

Основна частина

Основні терміни, які стосуються управління трафіком:

Черги. Вони утворюють основу всього управління трафіком і є невід'ємною частиною системи планування. У мережах під чергою розуміють буфер, де пакети (або інші одиниці даних) очікують передачі пристроєм. У простій моделі пакети передаються за принципом «першим прийшов - першим пішов». Такий алгоритм (дисципліна) обробки черги в області інформаційних технологій позначається як FIFO.У такої дисципліни обробки черги немає ніяких механізмів для управління трафіком. Все, що вона вміє робити – це поміщати пакети в чергу і витягувати їх звідти. Набагато цікавіше, коли алгоритм обробки черги включає в себе механізми затримки пакетів, зміни

порядку, втрати і пріоритезації пакетів в декількох чергах. У черг можуть існувати підчерги, які підвищують можливості системи планування.

Всього є 6 типів черг. Серед них 3 типи є classfull (ті, які можуть розділяти трафік по смугах).Інші 3 типи – classless.Classless черги є більш простими об'єктами, ніж classfull, і здатні лише встановлювати певні обмеження на передачу трафіку.

Classless-черги:

tbf (*token bucket filter*) - простий тип черги, що не виконує поділу пакетів, який утримує швидкість передачі пакетів на приблизно постійному рівні (меншому, ніж реальна швидкість інтерфейсу). При цьому, якщо швидкість передачі менша за задане значення, то пакети кладуться в чергу , поки не буде досягнутий певний розмір черги, яка потім передається на обробку (але відбувається деяка затримка даних, тому що відбувається очікування необхідної кількості пакетів для підтримки постійної швидкості передачі). Якщо ж швидкість перевищує задану, то «зайві» пакети просто відкидаються. Такий тип черги не можна порекомендувати для мереж, де дуже різко змінюється завантаженість, тому що можуть виникнути невиправдані затримки при слабкому завантаженні мережі і зниження пропускної здатності інтерфейсу при великому завантаженні.

sqf (*stochastic fairness queueing)* - черга рівномірного випадкового розподілу пакетів). Алгоритм роботи цього типу черг такий: дані, що надходять у чергу, поділяються на досить велику кількість «віртуальних» підчерг (віртуальних, тому що в пам'яті існує одна черга, але вона представляється у вигляді сукупності багатьох підчерг), з підчерги дані витягуються по черзі. Тобто, це нагадує TokenRing - мережі з передачею маркера покільцю. Підчерга, що отримала маркер, передає один пакет даних, а маркер переходить до наступної підчерги.

pfifo (ліміт пакетів) і *bfifo* (ліміт байтів). Ці черги є простими чергами певної довжини, які не виконують ніяких дій над пакетами, які надходять в них. Єдиний параметр таких черг - limit, що означає розмірчерги в пакетах (pfifo) і в байтах (bfifo).

Classfull-черги:

PRIO-черги. Черга такого типу може розділяти трафік між трьома смугами, які є чергами будь-якого типу. Поділ здійснюється на основі фільтрів. Кожна підчерга, що входить до PRIO, має свій пріоритет, який визначається значенням handle. При цьому більший пріоритет має та підчерга, яка відноситься до класу 1:. Тобто, при вилученні пакета з черги, спочатку досліджується підчерга з великим пріоритетом, якщо в останньої немає пакетів для обробки, то вибирається черга з більш низьким пріоритетом.

cbq (*classfull based queueing*). Цей тип дозволяє додавати велику кількість класів і здійснювати вельми нетривіальну обробку трафіку. Розподіл трафіку по классах здійснюється за допомогою фільтрів. Але cbq черги занадто складні і не завжди роблять те, що від них вимагається.

htb (*hierarchial token bucket*) - мають ті ж можливості, що і cbq-черги, але позбавлені недоліку останніх – зайвоїг роміздкості синтаксису.

Потік – це окреме з'єднання або діалог між двома хостами. Будь-який унікальний набір пакетів, що передається між двома хостами,може розглядатися як потік. У протоколі TCP поняття з'єднання з вихідними, цільовими адресами і портами являє собою потік. Аналогічно може бути визначено поняття потоку для протоколу UDP.

Поняття потоку дуже важливе, коли смуга пропускання ділиться на рівні частини між конкуруючими з'єднаннями, особливо якщ одеякі програми спеціальн остворюють велику кількість з'єднань.

Ще два фундаментальних поняття системи управління трафіком – це ***токени*** і ***буфери***.

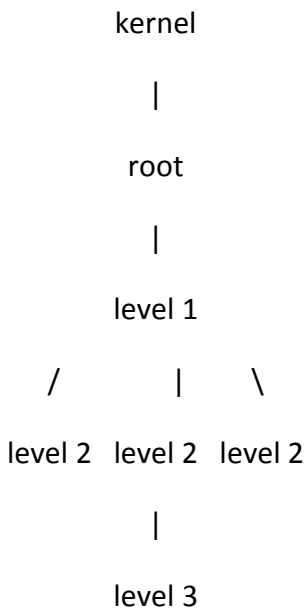
Для того, щоб контролювати швидкість вилучення об'єктів з черги, алгоритм обробки може рахувати число пакетів або байт в об'єкт, який вилучається з черги. Але цей метод вимагає складної реалізації. Інший метод, що набув широкого поширення в управлінні трафіком – це генерація токенів з певною швидкістю, а вилучення об'єктів з черги проводиться тільки в разі наявності **токенів.** Максимальна кількість токенів задається розміром **буфера**, в який поміщаються згенеровані токени. Усі наявні в буфері токени, можуть бути використані без очікування генерації наступних токенів.

Як відбувається вилучення ядром пакета з черги ?

Ядро запитує чергу найвищого рівня (root) про необхідність вилучення пакета. Черга верхнього рівня перевіряє черги нижнього рівня і вибирає ту , яка підходить за певними параметрами (це залежить від типу черги - контейнера і її

налаштувань), після здійснення вибору черга верхнього рівня вилучає пакет і передає його вище (чергзі, що знаходиться вище по ієрархії , або ядру, якщо це черга root). Таким чином, сама ієрархія побудована за принципом мінімальних знань про нижні рівні. Кожен рівень знає про існування тільки одного нижнього рівня та здійснює вибір тільки в рамках одного рівня.

Так приблизно можна уявити схему рівнів черг :



Класичні елементисистеми управління трафіком:

Обмеженнявихідноготрафіку (**shaping**) – це механізм, за допомогою якого пакети затримуються перед передачею для того, щоб швидкість передачі відповідала бажаній. Це один з найбільш часто використовуваних механізмів управління трафіком.

Обмеження вхідного трафіку (**policing**) - елемент системи якості обслуговування, що обмежує трафік. Цей механізм приймає пакети до певної швидкості, а над частиною трафіку, яка перевищила заданий поріг виконується певна дія. Наприклад, можна знищувати трафік, або перекласифікувати його. Не дивлячись на те, що в даному випадку теж використовується концепція буфера токенів, він не підтримує можливість затримки пакетів на відміну від механізму обмеження вихідного трафіку.

Планування (**scheduling**) - це механізм , який дозволяє впорядковувати або змінювати порядок об'єкти між входом і виходом конкретної черги. Прикладами планувальника можуть послужити алгоритми FIFO, WRR та інші.

Класифікація (**classifying**) - механізм, що розділяє пакети для різної обробки, можливо в різні черги.

Знищення (**dropping**) - механізм, що знищує дані. Наприклад, він використовується при переповненні буфера даних обмежувача вихідного трафіку.

Маркування (**marking**) - механізм зміни пакета. Зверніть увагу, що це не fwmark. Цілі MARK і - mark утиліт iptables і ipchains відповідно, модифікують метадані пакета, а не сам пакет.

У обробці трафіку беруть участь три ключових сутності: **дисципліни обробки пакетів**, **класи** та **фільтри**. Налаштування ефективної системи обмеження трафіку неможливе без розуміння механізмів їх роботи, ролі та зв'язку один з одним.

- **Дисципліна обробки пакетів** (**qdisc**) - черга пакетів і закріплений за нею алгоритм обробки.
- **Клас** (**class**) - логічний контейнер, який може містити кілька підкласів або дисципліну.

Класи - це об'єкти, які є складовими частинами classfull-черг, але самі по собі чергами не є.

Класи існують тільки усередині класових дисциплін обробки черги.

- **Фільтр** (**filter**) - механізм класифікації трафіку.

Фільтри представляють собою найбільш складну компоненту системи управління трафіком в Linux. Вони забезпечують зручний механізм з'єднання декількох елементів управління трафіком в єдине ціле.

Як і **ip**, команда **tc** може керувати вище переліченими об'єктами:

qdisc – керування чергами;

class – керування певними частинами черги, наприклад, смугами;

filters – керування фільтрами, фільтр визначає, в яку смугу черги потрапить той чи інший пакет.

Практичне виконання:

Спробуємо за допомогою утиліти TC розділити трафік на класи та обмежити швидкість виходу в інтернет на мережевому інтерфейсі eth1. Нижче наведена послідовність команд:

1)Видаляємо налаштування кореневого класу:

tc qdisc del dev eth1 root

tc qdisc del dev eth1 ingress

2) Створюємо кореневий клас зі швидкістю загального вхідного каналу в 1000 Мбіт:

tc qdisc add dev eth1 root handle 1:0 cbq avpkt 1000 bandwidth 1000mbit

Розглянемо детальніше цю команду:

dev eth1 - це пристрій, до якого ми підключаємо дисципліну обробки черги

root - вказує що це коренева дисципліна

handle 1:0 - задається дескриптор у формі старший номер: молодший номер

cbq - вказуємо тип дисципліни обробки черги

avpkt - визначає усереднений розмір пакету. Для ethernet-інтерфейсів можна вважати його рівним 1000 байт при MTU рівному 1500 байтам.

bandwidth - фізична пропускна здатність нашого пристрою

3)Створюємо кореневий клас для вихідного каналу

#tc qdisc add dev eth1 handle ffff : ingress

4)Створюємо клас з шириною каналу 15200kbit

#tc class add dev eth1 parent 1: class id 1:1 cbq rate 15200kbit allot 1500 prio 5 bounded isolated

classid - для classid молодший номер повинен бути унікальним

rate - цей параметр задає обмеження смуги пропускання

prio – присвоює классам пріоритети – чим менше значення, Тим вище пріоритет

bounded–позначає, що классу забороняється займати вільну смугу пропускання інших класів, налаштованих з параметром sharing

isolated–позначає, що інші класи не зможуть зайняти смугу цього класу, навіть якщо вона буде вільна

5)Класифікуємо трафік за допомогою фільтрів і направляємо його в потрібні класи

#tc filter add dev eth1: protocol ip prio 16 u32 match ip dst 192.168.0.10 flowid 1:1

u32 – це класифікатор, що дозволяє класифікувати пакети на підставі їх атрибутів

flowid - задає дескриптор класу, куди направляються пакети, відповідні заданому фільтру

6) Для більш рівномірного розподілу пропускної здатності між з'єднаннями, приєднаємо до класу дисципліну обробки черги SFQ

#tcqdisc add dev eth1 parent 1:1 sfq perturb 10

sfq – дисципліна для рівномірного розподілу пропускної здатності між з'єднаннями

perturb - цей параметр задає інтервал у секундах, через який відбувається зміна функції

7) Наступною командою ми будемо контролювати різке збільшення трафіку

#tc filter add dev eth1 parent ffff: protocol ip prio 50 u32 match ip src 192.168.0.10/32 police rate 15200Kbit burst 15564800 drop flowid 1:1

burst - контролюєреакціюнарізкезбільшеннятрафіку

Переглянути результат можна за допомогою команди #tc - s qdisc ls dev eth1

Висновки:

Я освоїв основи управління трафіком в ОС Linux, ознайомився з основними термінами, ознайомився із засобами та командами утиліти TC з пакету iproute2, призначеної для керування мережевим трафіком, також ознайомився з основними термінами та поняттями, які стосуються управління трафіком, а на практиці виконав наступне: створив кореневі класи для вхідного та вихідного каналів, а також клас з обмеженою шириною каналу, таким чином я обмежив швидкість вихідного трафіку до 15200 Kbit і, крім того, встановив контроль над різким збільшенням трафіку.

Використаний матеріал: частина теоретичного матеріалу взята з сайту http://pc-inform.ru/