Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 22.Б10-мм

Улучшение анализа сетевого трафика в Miminet

Казбеков Руслан Радикович

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
старший преподаватель кафедры системного программирования, И. В. Зеленчук

Санкт-Петербург
2024

Оглавление

Введение

В современном мире компьютерные сети играют важнейшую роль в обеспечении связности и обмена информацией между различными устройствами. Создание и управление сетями становится все более важной задачей, требующей глубоких знаний и навыков в области информационных технологий.

Компьютерная сеть – это система, позволяющая взаимосвязанным вычислительным устройствам обмениваться данными.

На Математико-механическом факультете Санкт-Петербургского государственного университета преподается предмет «Компьютерные сети и хранилища данных». Для более успешного освоения материала и обучения был создан веб-эмулятор компьютерных сетей Miminet [2].

Веб-эмулятор работает следующим образом:

1. В веб-интерфейсе клиент рисует сеть, которую необходимо построить.

2. Данные о топологии сети отправляются на сервер. На сервере сеть эмулируется с помощью эмулятора компьютерных сетей Mininet [3] (конкретнее, с помощью надстройки над ним — IPMininet [1], позволяющей создавать IP-сети). Для каждого устройства создается сетевой интерфейс со своим уникальным адресом и другими настройками.

3. После построения сеть начинает эмуляцию. Во время эмуляции сети идет захват сетевого трафика, который затем обрабатывается для визуализации.

4. Клиент запускает сгенерированную анимацию в веб-интерфейсе.

В данный момент захват трафика на сетевых интерфейсах ведется только для пакетов, которые являются исходящими. Данный способ позволяет однозначно воссоздать анимацию движения пакетов по сети,

но Mimishark — веб-анализатор, не дает информации о всём трафике, который перемещается в сети.

В связи с этим была поставлена задача придумать и внедрить способ, с помощью которого можно было бы однозначно определять, является ли каждый пакет входящим или исходящим для каждого сетевого интерфейса, на котором осуществляется захват трафика, что позволило бы и построить анимацию, и изучить информацию о всем трафике сети.

1. Постановка задачи

Целью работы является поиск и внедрение способа, с помощью которого можно было бы однозначно определять, является ли каждый пакет входящим или исходящим для конкретного интерфейса. Для её выполнения были поставлены следующие задачи:

1. Провести обзор существующих инструментов захвата сетевого трафика.
2. Найти способ отслеживания пакетов вместе с их направлением.
3. Реализовать способ отслеживания пакетов вместе с их направлением.
4. Внедрить способ отслеживания пакетов вместе с их направлением в Miminet.

2. Обзор

Как происходит захват сетевого трафика в IPMininet во время эмуляции сейчас:

1. Во время построения топологии сети вызывается метод addNetworkCapture(), который добавляет оверлей (класс с методами, направленными на взаимодействие с элементами топологии сети), занимающийся захватом трафика. В аргументы метода передается список интерфейсов, на которых нужно произвести захват, базовое названия файла, в который будет сохранена информация о пакете, и дополнительные аргументы.

2. Затем при конфигурации сети вызывается метод start() NetworkCapture оверлея, который запускает tcpdump[9] на каждом интерфейсе, переданном в addNetworkCapture(), и начинается захват трафика и запись информации о захваченных пакетах в pcap файлы (служат для хранения информации о пакете).

3. Затем после завершения всех job-ов (действий в сети; команда ping - пример job-a), сеть останавливает свою работу и все процессы завершаются.

Для того, чтобы построить корректную анимацию движения пакетов, необходимо знать, является ли конкретный пакет входящим или исходящим для интерфейса. Сейчас это достигается путём захвата только исходящих пакетов, так как tcpdump не сохраняет в pcap информацию о направлении пакета.

2.1. Ближайшие аналоги tcpdump

tcpdump - это "сниффер" командной строки(анализатор трафика). В качестве одного из главных аргументов принимает интерфейс, на котором будет вестись захват трафика.

Были рассмотрены другие "снифферы" коммандной строки, которые могли бы вывести информацию о направлении пакета: tshark[10] и netsniff-ng[8].

tshark и tcpdump, используемый с выбором конкретного интерфейса (применяется в IPMininet в данный момент), не выводят информацию о направлении пакета. Примеры на Листингах 1 и 2.

Листинг 1: Вывод tcpdump

```
sudo tcpdump -q

19:49:06.440309 IP 64.52.120.34.bc.googleusercontent.com.https > RuslanLenovo.54796: tcp 289
19:49:06.440311 IP 64.52.120.34.bc.googleusercontent.com.https > RuslanLenovo.54796: tcp 1122
19:49:06.440312 IP 64.52.120.34.bc.googleusercontent.com.https > RuslanLenovo.54796: tcp 39
19:49:06.440574 IP RuslanLenovo.54796 > 64.52.120.34.bc.googleusercontent.com.https: tcp 0
19:49:06.441076 IP RuslanLenovo.54796 > 64.52.120.34.bc.googleusercontent.com.https: tcp 35
19:49:06.441116 IP RuslanLenovo.54796 > 64.52.120.34.bc.googleusercontent.com.https: tcp 39
19:49:06.447579 IP 64.52.120.34.bc.googleusercontent.com.https > RuslanLenovo.54796: tcp 0
19:49:06.450850 IP 64.52.120.34.bc.googleusercontent.com.https > RuslanLenovo.54796: tcp 0
19:49:06.456157 IP RuslanLenovo.40656 > 150.138.117.34.bc.googleusercontent.com.https: tcp 185
19:49:06.471337 IP 150.138.117.34.bc.googleusercontent.com.https > RuslanLenovo.40656: tcp 0
```

Листинг 2: Вывод tshark

```
sudo tshark

1 0.000000000 34.117.138.150 → 192.168.0.108 TLSv1.2 170 Application Data
2 0.003252954 34.117.138.150 → 192.168.0.108 TLSv1.2 1223 Application Data
3 0.003253932 34.117.138.150 → 192.168.0.108 TLSv1.2 1466 Application Data
4 0.003254910 34.117.138.150 → 192.168.0.108 TLSv1.2 1466 Application Data
5 0.003255329 34.117.138.150 → 192.168.0.108 TLSv1.2 3348 Application Data
6 0.003256306 34.117.138.150 → 192.168.0.108 TLSv1.2 105 Application Data
7 0.003257284 34.117.138.150 → 192.168.0.108 TLSv1.2 163 Application Data
```

tcpdump выводит информацию о направлении, если прослушивать сразу все интерфейсы. Это не подходит, так как если для хостов создается отдельное пользовательское пространство, и там есть только его интерфейсы, то все остальные устройства (например, свитчи) создаются в основном юзерспейсе, где также находится и другие интерфейсы (например eth0 — главный Ethernet интерфейс). Прослушивание таким образом будет ловить ненужные нам пакеты, не относящиеся к конкретной IPMininet сети.

Листинг 3: Вывод tcpdump с ключом -i any

```
sudo tcpdump -i any -q

19:38:45.876670 wlp4s0 In  IP 149.154.167.51.https > RuslanLenovo.33910: tcp 105
19:38:45.876730 wlp4s0 Out IP RuslanLenovo.33910 > 149.154.167.51.https: tcp 0
19:38:45.976595 lo    In  IP localhost.54769 > localhost.domain: UDP, length 44
19:38:45.976916 wlp4s0 Out IP RuslanLenovo.35919 > _gateway.domain: UDP, length 55
19:38:45.983078 wlp4s0 In  IP _gateway.domain > RuslanLenovo.35919: UDP, length 110
19:38:45.983250 wlp4s0 Out IP RuslanLenovo.35919 > _gateway.domain: UDP, length 44
19:38:45.986286 wlp4s0 In  IP _gateway.domain > RuslanLenovo.35919: UDP, length 99
19:38:45.986713 lo    In  IP localhost.domain > localhost.54769: UDP, length 102
19:38:45.987017 lo    In  IP localhost.44815 > localhost.domain: UDP, length 45
19:38:45.987246 wlp4s0 Out IP RuslanLenovo.54881 > _gateway.domain: UDP, length 56
```

netsniff-ng позволяет увидеть направление пакета при выборе конкретного интерфейса. Далее выяснилось, что netsniff-ng и tcpdump делают это на основе адресов отправителя и получателя, содержащихся в пакете. Если пакет находится на промежуточном узле (например, на свитче в сети хост-свитч-хост), то направление будет показано, как «P», что значит «OTHERHOST».

Листинг 4: Вывод netsniff-ng

```
sudo netsniff-ng

< wlp4s0 139 #1 Unknown => Unknown IPv4 173.194.220.95/192.168.0.10
   8 Len 125 TCP 443(https)/41112 F PSH ACK Win 283 S/A 0x3
   940b6bc/0x27f5e39c
> wlp4s0 66 #2 Unknown => Unknown IPv4 192.168.0.108/173.194.220.95
   Len 52 TCP 41112/443(https) F FIN ACK Win 2062 S/A 0x27f
   5e39c/0x3940b705
< wlp4s0 66 #3 Unknown => Unknown IPv4 173.194.220.95/192.168.0.108
   Len 52 TCP 443(https)/41112 F FIN ACK Win 283 S/A 0x3940
   b705/0x27f5e39d
> wlp4s0 66 #4 Unknown => Unknown IPv4 192.168.0.108/173.194.220.95
   Len 52 TCP 41112/443(https) F ACK Win 2062 S/A 0x27f5e39
   d/0x3940b706
```

2.2. Выводы

Ни один из аналогов tcpdump не предоставляет необходимой функциональности для отслеживания направления пакета. После изучения

формата pcap[1] стало понятно, что он не хранит информацию о направлении пакета. Также, как было описано выше, netsniff-ng и tcpdump, выводя информацию о направлении пакета, исходят из адресов отправителя и получателя. Поэтому для того, чтобы понимать какое направление у конкретного пакета, нужно разделять пакеты на исходящие и входящие до их захвата, а потом записывать их в разные pcap файлы.

[1] https://wiki.wireshark.org/Development/LibpcapFileFormat

3. Метод

Были рассмотрены следующие пути решения поставленной задачи:

3.1. Добавление дополнительного оверлея

Существует вариант добавить еще один оверлей NetworkCapture, чтобы запускать два отдельных tcpdump-а на каждый интерфейс — один с аргументом -Qout, другой с аргументом -Qin. Но если сделать так, то количество процессов возрастёт в 2 раза. Если сейчас для обычной сети с 2 хостами и 1 свитчом запускается 8 tcpdump-ов, то для более сложной их будет слишком много. Поэтому этот способ не подходит. Пример простой сети с добавленными tcpdump-ами на Рис. 1.

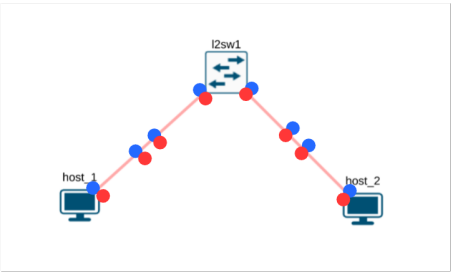

Рис. 1: Интерфейсы (красным выделены tcpdump-ы, которые будут ”ловить” исходящие, синим — входящие. В середине каждого линка добавлены промежуточные свитчи, делающие задержку движения пакетов, и в будущем позволяющие устраивать ”потерю” пакета)

3.2. Применение iptables

Так как ранее рассмотренные методы не помогли решить задачу, было принято решение рассмотреть задачу на более низком уровне. В

качестве инструмента был выбран iptables [5] — утилита командной строки для работы с netfilter [7] — межсетевой экран ядра Linux. Была разработана следующая структура: для каждого интерфейса в iptables создается два правила (действия, применяемые к пакетам при выполнении некоторых условий):

Листинг 5: Команды для добавления правил в iptables.

```
iptables –I INPUT –i {interface_name} –j NFQUEUE ––queue–num 1
iptables –I OUTPUT –o {interface_name} –j NFQUEUE ––queue–num 2
```

Одна очередь для исходящих пакетов, вторая — для входящих. В них по вышеописанным правилам, будут направляться пакеты. Далее при помощи модуля NetfilterQueue [4] пакеты бы доставались из очереди. Но есть одно препятствие: для исходящих пакетов timestamp, который берется из пакета, всегда 0.0, что не позволит нам построить анимацию. Существует еще одно расширение iptables — nflog. nflog создает дополнительный интерфейс, с которого с помощью tcpdump-а можно считывать информацию. Но тут также есть одно препятствие: nflog нет для arptables, который необходим для управления ARP-пакетами.

3.3. Реализация анализатора трафика с помощью библиотеки libpcap

В ходе решения задачи была выдвинута идея реализации анализатора трафика. Была взята библиотека libpcap [6], на основе которой написана утилита tcpdump. Анализатор работает следующим образом: сначала берется список интерфейсов, которые подходят для захвата, с помощью функции pcap_findalldevs(). Далее с помощью pcap_open_live() выбранный интерфейс открывается для захвата. Установка направления осуществляется с помощью функции pcap_setdirection() — в качестве аргументов ей передается дескриптор, используемый для чтения пакетов из сетевого интерфейса, содержащего пакеты, и направление. Так как нам нужно два разных направления, для этого необходимо создать два дескриптора: один для исходящих, другой для входящих.

Заключение

В ходе работы были решены следующие задачи:

1. Проведен обзор существующих инструментов захвата сетевого трафика;
2. Найден способ отслеживания пакетов вместе с их направлением в виде снифферa, реализованного с помощью библиотеки libpcap.
3. Написан снифферp[2].

[2] https://github.com/kazbekovruslan/sniffer

Список литературы

[1] IPMininet. — URL: https://ipmininet.readthedocs.io/en/latest/ (дата обращения: 15 декабря 2023 г.).

[2] Miminet. — URL: https://miminet.ru/ (дата обращения: 15 декабря 2023 г.).

[3] Mininet. — URL: https://mininet.org/ (дата обращения: 15 декабря 2023 г.).

[4] NetfilterQueue. — URL: https://github.com/oremanj/python-netfilterqueue (дата обращения: 15 декабря 2023 г.).

[5] iptables. — URL: https://www.netfilter.org/projects/iptables/index.html (дата обращения: 15 декабря 2023 г.).

[6] libpcap. — URL: https://www.tcpdump.org/ (дата обращения: 15 декабря 2023 г.).

[7] netfilter. — URL: https://www.netfilter.org/ (дата обращения: 15 декабря 2023 г.).

[8] netsniff-ng. — URL: http://netsniff-ng.org/ (дата обращения: 15 декабря 2023 г.).

[9] tcpdump. — URL: https://www.tcpdump.org/ (дата обращения: 15 декабря 2023 г.).

[10] tshark. — URL: https://tshark.dev/ (дата обращения: 15 декабря 2023 г.).