Автоматизация работы сетевого администратора

В ежедневной работе сетевого администратора можно выделить следующие задачи: управление конфигурацией сети и ее компонентами, мониторинг, обеспечение отказоустойчивости и безопасности. Наличие в инфраструктуре компании большого количества различных устройств не способствует использованию единого интерфейса управления и потенциально может приводить к ошибкам в конфигурации из-за человеческого фактора.

Автоматизация работы сетевого администратора осуществляется с применением различных инструментов и решений для более быстрого и эффективного выполнения рутинных задач, а самое главное для получения детерминированных результатов.

В настоящее время не существует единого подхода по автоматизации при работе с сетевым оборудованием, это связано с использованием различного проприетарного программно-аппаратного обеспечения. Также отсутствует большое количество литературы на русском языке, описывающей методы и способы внедрения автоматизации в сетевую инфраструктуру.

Наша задача - рассказать вам об архитектуре построения программируемых сетей и на практических примерах показать инструменты, которые вы сможете использовать в своей инфраструктуре, но обо всем по порядку.

История возникновения и теория

Программно-конфигурируемая сеть или Software Defined Networking – сеть передачи данных, в которой уровень управления сетью отделен от уровня передачи данных и реализуется программно.

Идея о создании подобных систем возникла еще в 2006 году, инициаторами выступили специалисты университета Стэнфорда и Беркли. Данное предложение активно поддержали крупные IT-компании: Google, Facebook, Microsoft, Oracle, Dell и др.

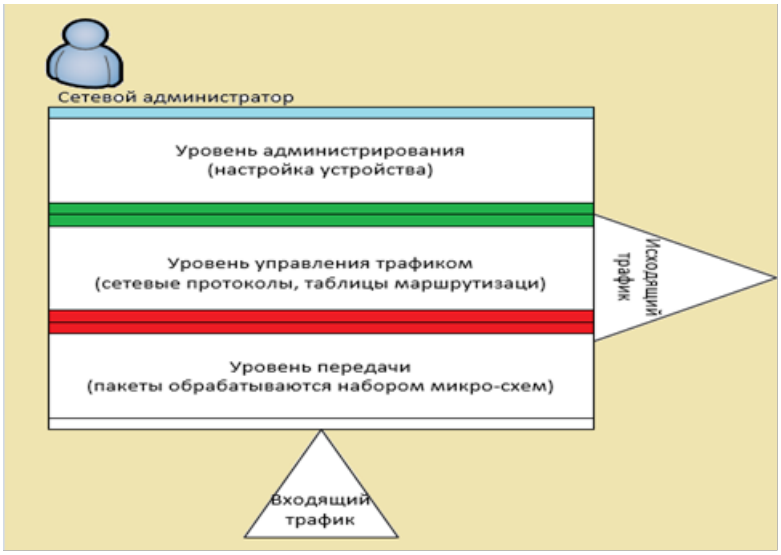
Потребность в разработке новых видов сетей была связана с увеличением общего объема трафика, что было вызвано ростом числа активных абонентов по всему миру. Начиная с 2007 года ежегодные темпы роста пропускной способности сетей во всем мире составляли около 60%, однако исследования специалистов IEEE показывают, что пропускную способность каналов связи требуется увеличивать вдвое раз в два года.

В тоже время с ростом нагрузки на сети усложнялись задачи по администрированию, увеличивался перечень поддерживаемых протоколов, повышались требования к безопасности и отказоустойчивости. Современная топология сетей состоит из устройств (узлов), аппаратно-программное обеспечение которых постоянно усложняется, для того чтобы поддерживать все современные технологии.

В качестве основных идей при разработке SDN были выдвинуты следующие:

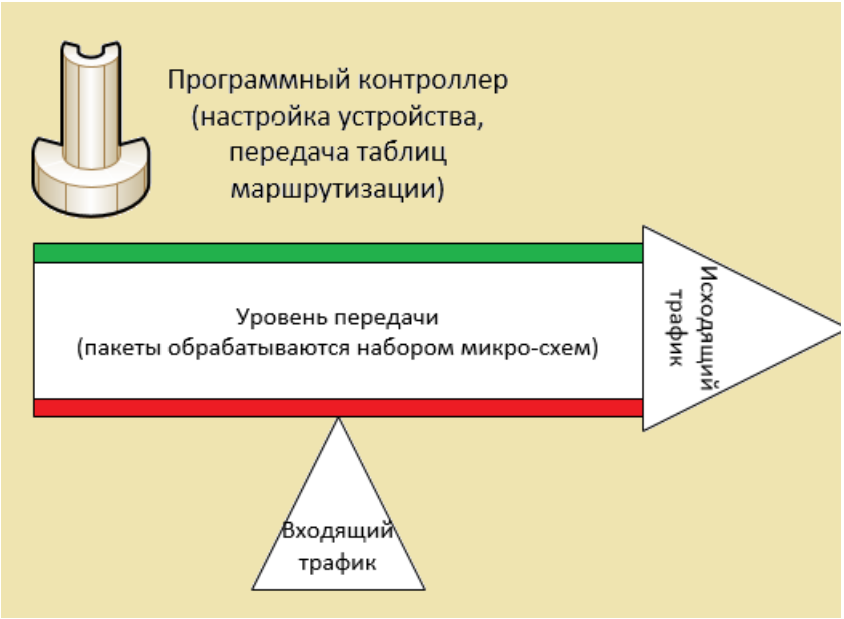
- разделение процессов обработки трафика в архитектуре сетевого устройства;
- централизованный интерфейс, не зависящий от производителя, служащий для управления элементами сети (узлами);
- виртуализация физических ресурсов сети.

В работе сетевого устройства можно выделить три абстрактных уровня: уровень передачи (Data Plane), уровень управления трафиком (Control Plane) и уровень администрирования (Management Plane):



1. **Уровень передачи** отвечает за физическую отправку трафика через сетевое устройство с использованием набора микросхем.
2. **Уровень управления** служит для организации логики распределения маршрутов сетевого устройства. Благодаря нему выбираются оптимальные пути, согласно сетевому протоколу, что позволяет обмениваться трафиком между узлами.
3. **Уровень администрирования** предназначен для управления оборудованием, к нему относится: конфигурирование, передача настроек, сбор данных.

Подход SDN изменил концепцию взаимодействия между абстрактными уровнями сетевых устройств. На рисунке показано, что за уровень управления и администрирования стал отвечать отдельный программный контроллер, сетевой узел остался работать на уровне передачи.



Данное решение способствовало развитию технологий и инструментов, которые упрощают выполнение задач администрирования сети, среди которых можно выделить :

1. Протокол управления OpenFlow, который обеспечивает передачу служебного трафика (таблицы маршрутизации, информация о топологии сети).
2. Виртуализация, позволяющая перенести сетевые функции в программную среду, тем самым заменяя аппаратные решения.
3. Автоматизация, позволяющая с помощью программных средств реализовать реконфигурацию, запрос данных и мониторинг узлов сети.
4. Сетевые фабрики представлены в виде комплексной системы, которая позиционирует набор сетевого оборудования, как единое целое.
5. Software-Defined Networking in a wide area network (SD-WAN) осуществляющая интеллектуальное управление потоками трафика учитывая его род и назначение.

Таким образом, мы с вами кратко познакомились с концепцией и архитектурой построения

программируемых сетей. В рамках наших статей мы рассмотрим инструменты по инвентаризации, сбору данных, конфигурировании различного сетевого оборудования без привязки к производителю.

Сбор данных и конфигурирование сетевого оборудования

За долгие годы развития ИТ-индустрии способ управления сетями оставался не изменчив. В это же время появлялись новые протоколы для взаимодействия, расширялись возможности самих сетевых устройств. Тем не менее Command Line Interface (CLI) остался широко распространенным и популярным среди сетевых администраторов.

Данный метод в основном используется по умолчанию при первичной настройке сетевого оборудования, он поддерживает различные типы подключения (Telnet, SSH и другие). Администратор сети взаимодействует с программной оболочкой устройства, которая передает команды в ядро ОС и тем самым вносит изменения в конфигурационный файл.

Файл конфигурации – текстовый файл, содержащий настройки системы или ее части (утилиты, демона и т. п.). Обычно инициализируется при запуске ОС и доступен для резервного копирования.

Ежедневно администраторы опрашивают сетевые устройства и вносят изменения в их конфигурацию, подключаясь с помощью CLI, отправляя команды из заранее подготовленных файлов-шаблонов. Данный метод несет за собой большие риски и не является эффективным в части трудозатрат, ведь ошибка при настройке одного из устройств может повлиять на работоспособность всей сети.

Развитие языков программирования способствовало автоматизации управления и администрирования сетевых устройств. Алгоритм по автоматической отправке команд до оборудования состоит из двух шагов: формирование списка команд или конфигурационного файла, далее его доставка.

Command Line Interface (CLI)

Прежде чем мы познакомимся с программными инструментами автоматизации, необходимо обсудить классический процесс взаимодействия с сетевым оборудованием. В качестве сервера выступает сетевое оборудование, в качестве клиента – запущенный терминал на самом устройстве или на ПК администратора сети, формат ввода-вывода команд - текстовая строка.

Преимуществом использования данного подхода для взаимодействия с сетевым устройством является:

- малая требуемая вычислительная мощность на стороне клиента;
- мгновенная реакция на ввод команд со стороны сервера-системы;
- возможность вводить последовательность заранее подготовленных команд, как вручную, так и с помощью программных средств автоматизации;
- полная документация команд от различных производителей оборудования.

Как правило, при работе через CLI существует несколько режимов доступа, заранее запрограммированных в сетевом ПО: режим без привилегий (первичное подключение к устройству), привилегированный режим (просмотр конфигурации устройства, базовые команды), режим конфигурации (внесение изменений, осуществление настроек).

```
R1(config-if)# do sh running-config
Building configuration...

Current configuration:
!
frr version 6.0.2
frr defaults traditional
hostname base-sys
log syslog informational
no ipv6 forwarding
hostname R1
service integrated-vtysh-config
!
```

При внедрении средств автоматизации управления сетевыми устройствами - метод CLI рассматривается в первую очередь, потому что его поддержка заявлена каждым производителем. Он обладает полной документацией команд и высокой отказоустойчивостью, не требует больших вычислительных мощностей со стороны подключаемого клиента. Для перехода в интерфейс CLI необходимо установить соединение с узлом, для этого используется один из сетевых протоколов (Telnet, SSH и др).

Python для сетевого администратора

Автоматизация процесса обслуживания сетевых устройств невозможна без использования одного из языков программирования.

Для 2021 года базовое знание Python это уже не преимущество кандидата (системного администратора, сетевого инженера) , а одно из требований при трудоустройстве на работу. В рамках наших статей мы не будем обучать вас основам языка, потому что в Интернете большое количество ресурсов на эту тему и вы сможете подготовиться самостоятельно.

Наша задача показать и разобрать практические примеры, где вы можете использовать автоматизацию для упрощения ежедневных рутинных процессов.

Итак, одно из весомых преимуществ языка Python - большое количество сторонних библиотек для решения задач из различных областей. Естественно, энтузиасты и разработчики в рамках работы с сетью и сетевым оборудованием написали достаточное количество модулей. Давайте перечислим популярных из них, поддерживающие режим взаимодействия CLI:

- **telnetlib** - встроенная библиотека, она используется для обеспечения доступа к сетевому устройству и отправке команд по Telnet протоколу (не является безопасным).
- **pexpect** - внешняя библиотека работает для передачи данных по протоколам: ssh, telnet, sftp и др. Считается универсальным инструментом для передачи команд как в ОС, так и на сетевое оборудование. Логика работы основана на ожидании получения соответствующей строки от устройства и дальнейшей отправки команды.
- **paramiko** - внешняя библиотека, обеспечивающая организацию SSH-соединения. Работает в режиме клиента и сервера, соответственно в качестве передачи данных может использоваться только протокол SSH.
- **netmiko** - внешняя библиотека от сообщества сетевых инженеров и администраторов, была разработана специально для взаимодействия с сетевым оборудованием различных производителей (адаптация под сетевую ОС). Netmiko включает в себя paramiko, описанный раннее.

Пример с telnetlib

В рамках сегодняшней статьи мы рассмотрим лишь библиотеку telnetlib языка Python3. Мы будем подключаться по Telnet к маршрутизатору Cisco (IOS15), вы же можете использовать оборудование другого производителя.

Наша задача: сконфигурировать два сетевых интерфейса (задать ip-адресацию), включить поддержку динамического протокола маршрутизации OSPF.

После символа # будет следовать комментарий к строке, файл написан в формате ".py"

```
import getpass   # импорт библиотеки getpass
import telnetlib # импорт библиотеки getpass
```

```
HOST = "192.168.122.71"  # ip-адрес маршрутизатора Cisco
user = input("Enter your telnet username: ") # запрос ввода имени
password = getpass.getpass() # запрос ввода пароля в зашифрованном виде

tn = telnetlib.Telnet(HOST) # инициализация подключения telnet

tn.read_until(b"Username: ") # ожидание строки Username от CLI
tn.write(user.encode("ascii") + b"\n")#передача данных пользователя и ENTER

"ЕСЛИ требуется пароль, то передать"
if password:
    tn.read_until(b"Password: ")
    tn.write(password.encode("ascii") + b"\n")

tn.write(b"enable\n")# вход в режим enable
tn.write(b"cisco\n") # ввод пароля для enable, не всегда требуется
tn.write(b"conf t\n")# переход в конфигурационный режим
tn.write(b"int loop 0\n") # настройка интерфейса loop 0
tn.write(b"ip address 1.1.1.1 255.255.255.255\n")# задание ip-адреса
tn.write(b"int loop 1\n") # настройка интерфейса loop 1
tn.write(b"ip address 2.2.2.2 255.255.255.255\n")# задание ip-адреса
tn.write(b"router ospf 1\n") # включение протокола OSPF
tn.write(b"network 0.0.0.0 255.255.255.255 area 0\n") # настройка OSPF
tn.write(b"end\n")# выход из режима конфигурации
tn.write(b"exit\n")# выход из режима enable

print(tn.read_all().decode("ascii") # вывести на экран команды и результат
```

Приведенный выше скрипт написан для начала знакомства с автоматизацией, в дальнейшем будут приведены более сложные примеры. Но даже он позволяет вам подключаться к различному сетевому оборудованию и отправлять команды конфигурации или сбора данных, после чего выводить их на экран или в файл.

Таким образом, сегодня мы познакомились с понятием "автоматизация при работе с сетевым оборудованием", рассмотрели основные концепции и начали изучать инструменты работы с сетью на языке Python. Мы написали скрипт, позволяющий вам взаимодействовать с оборудованием и получать предсказуемый результат. Далее будет более подробное знакомство с различными инструментами, приведем практические примеры использования, оставайтесь на связи!