Содержание

1 Постановка задачи

Цель работы

Разработать систему сбора и хранения метрик с пользовательских устройств с помощью протокола SNMP.

Задачи

1. Исследовать протокол SNMP.
2. Настроить конфигурацию службы SNMP на целевых устройствах.
3. Определить стэк разработки.
4. Разработать приложение для сбора метрик с целевых устройств.

2 Обзор литературы

Simple Network Management Protocol (SNMP) представляет собой стандартный протокол управления сетью[**Mauro2005**], который используется для мониторинга и управления сетевыми устройствами в информационных технологиях. Он разработан для обеспечения стандартизированного и эффективного способа сбора информации о состоянии устройств в сети, а также для удаленного управления этими устройствами.

При использовании SNMP один или несколько административных компьютеров, именуемых менеджерами, осуществляют мониторинг или управление группой хостов или устройств в компьютерной сети. На каждом управляемом устройстве установлена постоянно работающая программа, известная как агент, которая через SNMP передает информацию менеджеру.

Сети, управляемые протоколом SNMP, состоят из трех основных компонентов[**snmp**]:

1. Управляемое устройство;
2. Агент - программное обеспечение, работающее на управляемом устройстве или на устройстве, подключенном к интерфейсу управления управляемого устройства;
3. Система сетевого управления (Network Management System, NMS) - программное обеспечение, взаимодействующее с менеджерами для поддержки комплексной структуры данных, отражающей состояние сети.

Управляемое устройство представляет собой элемент сети (оборудование или программное обеспечение), которое реализует интерфейс управления (не обязательно SNMP), позволяющий однонаправленный (только для чтения) или двунаправленный доступ к конкретной информации об элементе. Управляемые устройства обмениваются этой информацией с менеджером. Они могут включать в себя разнообразные устройства, такие как маршрутизаторы, серверы доступа, коммутаторы, мосты, концентраторы, IP-телефоны, IP-видеокамеры, компьютеры-хосты, принтеры и т.д.

Агент - это программный модуль сетевого управления, работающий на управляемом устройстве или на устройстве, подключенном к интерфейсу управления

управляемого устройства. Он обладает локальным знанием управляющей информации и преобразует эту информацию в формат, совместимый с SNMP, или извлекает ее из этого формата (медиация данных).

Система сетевого управления (NMS) включает приложение, которое отслеживает и контролирует управляемые устройства. NMS выполняют основную часть обработки данных, необходимых для сетевого управления. В управляемой сети может присутствовать одна или несколько NMS.

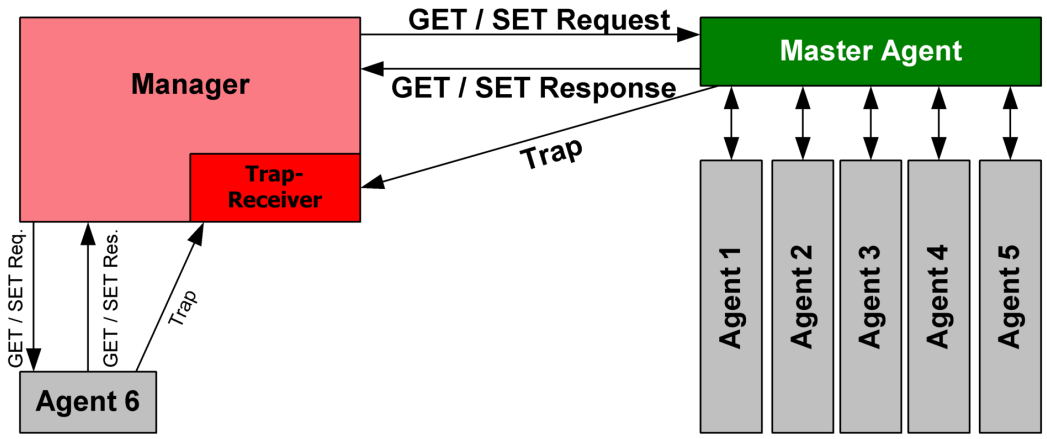

Рис. 1: Принципы коммуникации SNMP

Основными компонентами SNMP являются управляющая станция (SNMP менеджер), агенты SNMP и информационная база данных управляемых объектов (MIB - Management Information Base). MIB представляет собой структурированный набор данных, определяющих атрибуты и параметры управляемых объектов, которые могут быть мониторены и управляемы через SNMP. Каждое устройство, поддерживающее SNMP, имеет свою собственную MIB, определяющую доступные для мониторинга и управления параметры.

Преимущества SNMP включают в себя его стандартизированный характер, простоту использования и эффективность в мониторинге и управлении сетевыми устройствами. Благодаря SNMP администраторы сети могут получать информацию о производительности сети, загрузке устройств, использовании ресурсов, количестве ошибок в передаче данных и других параметрах, что помогает в выявлении и устранении проблем в сети.

В целом, SNMP является важным инструментом для администрирования сетей, обеспечивая возможность мониторинга и управления сетевыми устройствами,

что способствует эффективной работе сети и обеспечивает оперативное реагирование на проблемы.

3 Выбор технологий

Выбор языка программирования Python для разработки системы сбора и хранения метрик с использованием протокола SNMP обоснован рядом преимуществ данного языка. Python - это высокоуровневый язык программирования, который отличается простотой в изучении, читаемостью кода и мощными инструментами для разработки. Его синтаксис интуитивно понятен, что делает процесс написания кода более эффективным и быстрым. Кроме того, Python имеет обширную стандартную библиотеку, которая включает в себя множество полезных модулей для работы с сетями, обработки данных и других задач, что упрощает разработку и сокращает время на написание кода с нуля.

Для работы с протоколом SNMP в Python была выбрана библиотека PySNMP[**pysnmpSNMPLibrary**]. Её преимущества заключаются в том, что она предоставляет простой и удобный интерфейс для работы с SNMP протоколом, обеспечивая полный функционал для отправки запросов к агентам SNMP, получения данных, а также для управления устройствами. PySNMP предоставляет высокоуровневые абстракции для работы с SNMP, что позволяет сосредоточиться на разработке бизнес-логики приложения, минимизируя сложности взаимодействия с протоколом. Кроме того, PySNMP является популярной и активно поддерживаемой библиотекой в сообществе Python, что обеспечивает доступность документации, обновлений и поддержку со стороны разработчиков. Все эти факторы делают PySNMP превосходным выбором для реализации функционала сбора и хранения метрик с использованием протокола SNMP в рамках разрабатываемой системы.

4 Ход Работы

Конфигурация SNMP

Для работы с SNMP нужно установить клиента и демона. Для arch linux существует пакет net-snmpd, который включает в себя всё необходимое.

SNMP v3 добавляет безопасность и шифрованную аутентификацию/коммуникацию. Он использует схему конфигурации в файле /etc/snmp/snmpd.conf и дополнительную конфигурацию в файле /var/net-snmp/snmpd.conf.

Для первого конфигурационного файла в /etc/ можно записать непосредственно в него или использовать мастер настройки snmpconf:

```
$ mkdir /etc/snmp/
$ echo rouser read_only_user >> /etc/snmp/snmpd.conf
$ snmpconf -g basic_setup
```

Мы воспользуемся мастером настройки snmpconf.

Демон

Запускаем службу snmpd.service.

```
$ sudo systemctl start snmpd
```

Тестирование

Если используется SNMP 1 или 2c, используйте одну из следующих команд для тестирования конфигурации:

```
$ snmpwalk -v 2c -c read_only_community_string localhost | less
```

Мы смогли получить значения OID системы, значит конфигурация прошла успешно.

Определение OID для метрик

Прежде чем приступить к разработке приложения, необходимо определить OID для метрик. Мы будем отслеживать следующие метрики:

- Нагрузку процессора.
- Температуру процессора.
- Использование RAM.
- Исходящий и входящий интернет трафик.

Нагрузка процессора

1. **.1.3.6.1.4.1.2021.10.1.3.1** – нагрузка процессора за 1 минуту
2. **.1.3.6.1.4.1.2021.10.1.3.2** – нагрузка процессора за 5 минут
3. **.1.3.6.1.4.1.2021.10.1.3.3** – нагрузка процессора за 15 минут

Данная метрика показывает общую нагрузку процессора в зависимости от количества ядер, где 100% нагрузка равна количествую ядер. Например, в одноядерной системе нагрузка 1 будет соответсвовать 100% нагрузке, когда же в двухядерной системе нагрузка будет равна 50%.

Температура процессора

К сожалению, нельзя узнать температуру процессора через SNMP ”из коробки”. Поэтому мы воспользуемся системой мониторинга **lm_sensors** [**archlinuxLm**]. Она создаст своё MIB хранилище и мы получим доступ к нему через SNMP.

1. **LM-SENSORS-MIB::lmTempSensorsValue.1** – температура процессора

Как можно заметить OID может представляться как числовым значением, так и текстовым.

Интернет трафик

1. **IF-MIB::ifInOctets** – входящий интернет трафик
2. **IF-MIB::ifOutOctets** – исходящий интернет трафик

Данная метрика показывает какое количество октетов (байт) передалось из или в интерфейс с момента начала работы системы. Так как в системе может быть большое количество интерфейсов, то мы провёдем операцию под названием **snmpwalk** и соберём значения со всех интерфейсов.

Использование RAM

1. **UCD-SNMP-MIB::memTotalReal.0** – общее количество RAM на устройстве
2. **UCD-SNMP-MIB::memAvailReal.0** – доступное количество RAM для использования
3. **UCD-SNMP-MIB::memCached.0** – закеширование количество RAM на устройстве

Разработка приложения

Когда мы узнали все нужные для работы OID, мы можем приступить к разработке приложения. Библиотека **PySNMP** предоставляет простой и удобный интерфейс для работы с SNMP протоколом. Был написан класс SNMPClient.

```
from pysnmp.hlapi import *

class SNMPClient:
    def __init__(self, ip_address,  community='read_only_community_string'):
        self.ip_address = ip_address
        self.community = community
        self.snmp_engine = SnmpEngine()

    def __get_snmp_request(self, oid: ObjectIdentity):
        iterator = getCmd(self.snmp_engine,
                   CommunityData(self.community, mpModel=1),
                  UdpTransportTarget((self.ip_address, 161), timeout=1, retries=5,
    tagList=''),
                  ContextData(),
                  ObjectType(oid))
        errorIndication, errorStatus, errorIndex, varBinds = next(iterator)
        if errorIndication:
            print(errorIndication)
            return None
        elif errorStatus:
            print(f"{errorStatus.prettyPrint()} at {varBinds[int(errorIndex) -
   1][0] if errorIndex else '?'}")
            return None
        else:
            return varBinds[0][1].prettyPrint()

    def get_cpu_usage_1min(self):
        return self.__get_snmp_request(ObjectIdentity('.1.3.6.1.4.1.2021.10.1.3.1'
   ))
```

```
27
28      def get_cpu_usage_5min(self):
29          return self.__get_snmp_request(ObjectIdentity('.1.3.6.1.4.1.2021.10.1.3.2'
    ))
30
31      def get_cpu_usage_10min(self):
32          return self.__get_snmp_request(ObjectIdentity('.1.3.6.1.4.1.2021.10.1.3.3'
    ))
33
34      def get_internet_traffic_out(self):
35          return self.__get_snmp_request(ObjectIdentity("IF-MIB", "ifOutOctets", 2))
36
37      def get_internet_traffic_in(self):
38          return self.__get_snmp_request(ObjectIdentity("IF-MIB", "ifInOctets", 2))
39
40      def get_cpu_temperature(self):
41          return self.__get_snmp_request(ObjectIdentity("LM-SENSORS-MIB", "
    lmTempSensorsValue", 1))
42
43      def get_total_ram(self):
44          return self.__get_snmp_request(ObjectIdentity("UCD-SNMP-MIB", "
    memTotalReal", 0))
45
46      def get_avail_ram(self):
47          return self.__get_snmp_request(ObjectIdentity("UCD-SNMP-MIB", "
    memAvailReal", 0))
48
49      def get_cached_ram(self):
50          return self.__get_snmp_request(ObjectIdentity("UCD-SNMP-MIB", "memCached",
     0))
51
52      def get_buffer_ram(self):
53          return self.__get_snmp_request(ObjectIdentity("UCD-SNMP-MIB", "memBuffer",
     0))
54
55      def get_shared_ram(self):
56          return self.__get_snmp_request(ObjectIdentity("UCD-SNMP-MIB", "memShared",
     0))
57
58      def check_connection(self):
59          val = self.__get_snmp_request(ObjectIdentity("SNMPv2-MIB", "sysDescr", 0))
60          return True if val else False
61
62      def get_cpu_count(self):
63          count = 0
64          for (errorIndication, errorStatus, errorIndex, varBinds) in nextCmd(
65              SnmpEngine(),
66              CommunityData(self.community, mpModel=1),
```

```
            UdpTransportTarget((self.ip_address, 161)),
            ContextData(),
            ObjectType(ObjectIdentity("HOST-RESOURCES-MIB", "hrProcessorFrwID")),
            lexicographicMode=False
        ):
            if errorIndication:
                print(errorIndication)
                break
            elif errorStatus:
                print('%s at %s' % (
                    errorStatus.prettyPrint(),
                    errorIndex and varBinds[int(errorIndex) - 1][0] or '?'
                    )
                )
                break
            else:
                for _ in varBinds:
                    count += 1
        return count
```

5 Выводы

В рамках курсового проекта было реализовано приложение для сбора метрик с целевых устройств через SNMP.