Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Обнаружение узлов в сети.

Курсовая работа студента 345 группы

Свидерского Павла Юрьевича

Научный руководитель: Смирнов К.К.

Санкт-Петербург

2011

Содержание

1 Введение

С развитием сетевых технологий и ростом компьютерных сетей задача управления компьютерными сетями становится все более актуальной. Любая компьютерная сеть характеризуется своей внутренней структурой — топологией, т.е. схемой расположения и соединения сетевых устройств, входящих в рассматриваемую сеть. Знания о структуре сети могут быть получены различными методами и инструментами. Существующие решения, используемые протоколы и алгоритмы были рассмотрены в курсовой работе «Анализ и построение структуры сети» [1] на кафедре системного программирования Санкт-Петербургского Государственного Университета. Зная топологию компьютерной сети, можно определить наиболее нагруженные, а также подверженные сетевым атакам узлы. Следует понимать, что в таком представлении компьютерной сети нуждается только определенный круг людей — системные администраторы, обслуживающие данную сеть. С другой стороны, внутренняя структура сети, а также её изменения должны быть прозрачны и недоступны конечным пользователям.

Одна из основных задач системных администраторов — эффективное управление компьютерными сетями. А в случае сбоев в сети — немедленное выявление причины нестабильной или отсутствующей функциональности и её устранение максимально незаметно для конечных пользователей сети. Нестабильная работа сети может быть вызвана как некорректной работой самих сетевых устройств, так и намеренными действиями пользователей. Таким образом, наряду с построением топологии компьютерной сети стоит проблема поиска конкретного узла в сетевой структуре. Быстрое определение месторасположения источника сбоев в топологии сети дает возможность администраторам в кратчайшие сроки принять необходимые меры по устранению неисправности, включая вывод данного устройства из рабочей среды, вплоть до отключения канала связи с ним. Тем самым, обеспечить стабильную работу всей оставшейся инфраструктуры.

1.1 Задача обнаружения узлов

В рамках данной работы мы будем рассматривать компьютерные сети, построенные на базе управляемых сетевых коммутаторов. Управление коммутатором может осуществляться через соответствующий web-интерфейс, предоставленный производителем, а также посредством протоколов управления сетевыми устройствами SNMP, RMON и т. п.

Основная задача обнаружения узлов, рассматриваемая в данной курсовой работе, заключается в поиске месторасположения заданных узлов в структуре компьютерной сети. Сетевой узел определяется и идентифицируется в пределах рассматриваемой сети уникальным ад-

ресом сетевого или канального уровней. Соединение сетевого узла с сетевым коммутатором, безусловно, характеризуется номером физического порта коммутатора, к которому подключен узел. Именно поиск в топологии сети определенного сетевого коммутатора и отыскание соответствующего порта, непосредственно предоставляющего канал связи заданному узлу, ставится главной задачей этой курсовой работы.

1.2 Готовые решения и типичные проблемы

Существуют готовые программные продукты, решающие поставленную задачу, такие как 3Com Network Supervisor [3], HP OpenView Network Node Manager [4], Wiremaps [5], NetDisco [6], 10-strike LanState и другие. Однако, многие из них обладают различными недостатками.

- Программные инструменты нацелены на работу с сетевым оборудованием конкретного производителя и не применимы для использования в гетерогенных сетях, построенных на базе устройств от разных производителей.

- Необходимость поддержки оборудованием определенных протоколов сетевого уровня, таких как STP (Spanning Tree Protocol) [7] и LLDP (Link Layer Discovery Protocol) [8], в том числе проприетарных: CDP (Cisco Discovery Protocol), EDP (Extreme Discovery Protocol), SONMP (SynOptics Nortel Management Protocol). Вышеперечисленные протоколы канального уровня позволяют сетевому оборудованию оповещать сеть о своем существовании и характеристиках, а также собирать такие же оповещения, поступившие от соседнего оборудования. Множество старого сетевого оборудования, а также оборудование бюджетного класса не реализует данные протоколы.

- Некорпоративные решения низкого качества, как правило созданные для конкретной сети и неспособные подстраиваться под другие инфраструктуры и производителей оборудования.

- Решения являются платформозависимыми.

- Высокая стоимость, обусловленная решением широкого спектра задач сетевого администрирования и неприемлемая для небольших компаний.

1.3 Цель работы

Ни один из вышеперечисленных готовых продуктов не справился с задачей поиска узлов в тестовой гетерогенной сети, что и привело автора курсовой работы к разработке мак-

симально гибких методов и программного обеспечения для решения поставленной задачи. Программное решение должно обладать следующими свойствами:

- возможность функционирования в разнородных сетях, построенных на базе оборудования от разных производителей;
- не должно зависеть от поддержки сетевым оборудованием протоколов, позволяющих оповещать сеть о существовании данного оборудования и его характеристиках, таких как LLDP, STP, CDP;
- работа в непривилегированном режиме, используя для сбора данных только SNMP и DNS запросы;
- легкость в настройке при первом развёртывании;
- автоматическая адаптация к замене оборудования и изменениям топологии сети;
- способность автоматически строить топологию сети;
- платформонезависимость, то есть возможность работы на различных платформах, таких как GNU/Linux, Microsoft Windows и Mac OC X.

Вследствие отсутствия требования к поддерживаемым оборудованием протоколам оповещения, возникает нетривиальная задача построения топологии сетевых коммутаторов, возможные решения которой также следует предоставить.

2 Обзор решений Netdisco и Wiremaps

Исходя из цели работы, нам следует рассмотреть уже существующие проекты с похожей функциональностью и выбрать наиболее гибкое решения для последующего улучшения и реализации на его базе поставленных задач. Во-первых, проект должен быть с открытым исходным кодом, а также лицензия, по которой он распространяется, должна позволять нам использовать его код и вносить в него изменения. Наиболее интересными проектами, которые соответствуют вышеуказанным ограничениям, оказались Netdisco и Wiremaps. Ниже приведем более подробное описание и обзор имеющейся функциональности каждого из проектов.

2.1 Netdisco

Netdisco — это web-ориентированный инструмент сетевого администрирования, решающий следующие задачи:

- Сбор данных с сетевых устройств, учитывая их модель, производителя, версию прошивки и операционную систему;
- Определение устройства в сети по MAC или IP адресу и указание порта коммутатора, к которому оно подключено;
- Отключение порта коммутатора с соответствующей записью в контрольный журнал;
- Отправка отчета о IP адресе и использовании порта коммутатора в заданное и текущее время;
- Создание графического изображения карты сетевых устройств.

Netdisco не использует интерфейс командной строки, а получает все данные о сетевых устройства с помощью SNMP и DNS запросов. Автоматическое построение топологии сети осуществляется используя протоколы сетевого уровня, такие как LLDP, CDP, FDB и SONMP. Если оборудование не поддерживает выше перечисленные протоколы автоматического обнаружения, сетевую топологию можно задать вручную в конфигурационном файле. Netdisco использует FDB таблицы коммутаторов и ARP таблицы маршрутизаторов для сопоставления узлов в сети портам коммутаторов и обращения к ним по их IP адресам. Полученная и обработанная информация с сетевого оборудования записывается и хранится в SQL базе данных. Для показа графического изображения топологии сети используется пакет утилит визуализации графов GraphViz. Инструмент Netdisco является свободным проектом с открытым исходных кодом, полностью написан на языке Perl и распространяется по лицензии BSD License.

2.2 Wiremaps

Wiremaps, также как и Netdisco, является web-ориентированным инструментом управления сетью, очень схож по функциональности с Netdisco. Из главных задач, решаемых Wiremaps, стоит отметить:

- Сбор данных с управляемых сетевых устройств, подстраиваясь под модель, производителя и версию программно-аппаратных средств оборудования;

- Детальный вывод информации по каждому порту выбранного устрайства;
- Поиск узла в сети по заданному доменному имени, MAC или IP адресу, указание устройства и порта, к которому оно подключено;
- Хранение всех изменений конфигурации устройств и воможность просмотра состояния устройства в заданное время в прошлом.

Wiremaps для автоматического построения топологии сети использует протоколы обнаружения сетевого уровня (LLDP, CDP, FDB, SONMP), однако, в отличие от Netdisco, не имеет возможности ручного задания связей между устройствами в случае отсутствия поддержки оборудованием соответствующих протоколов. Определение IP адреса по доменному имени осуществляется в Wiremaps с помощью DNS запросов, сопоставление IP адресу MAC адреса и наоборот — с помощью записей, полученных из ARP таблиц управляемых устройств сети. Текущие данные об устройствах, а также вся история изменений хранится в SQL базе данных, что увеличивает гибкость и скорость приложения.

Wiremaps является свободным открытым продуктом написанным на объектно-ориентированном языке Python. Для повышения производительности использует событийно-ориентированный сетевой фреймворк Twisted [9], который предоставляет автоматическое распараллеливание обработки данных, а также web-сервер для пользовательского интерфейса управления. Таким образом, в отличии от Netdisco, Wiremaps не требует стороннего web-сервера. Пользовательский интерактивный интерфейс, предоставляемый Wiremaps, основан на AJAX подходе, что обеспечивает динамическую подгрузку и отображение обновлённых данных с сетевых устройств. Из минусов Wiremaps стоит отметить отсутствие возможности получить графическое представление топологии сети. Что касается лицензии, то данный продукт распространяется по GNU General Public License v3.

2.3 Вывод

Исходя из обзора на данный момент времени готовых решений и предоставляемой ими функциональности, создание нового проекта с нуля становится неоправданным. Оба рассмотренных инструмента управления сетью с практически равной степенью сложности подходят для разработки и внедрения решения поставленной задачи. Также лицензии, по которым они распространяются, не противоречат использованию их исходного кода. Однако, простота в установке, настройке и первом запуске, быстрота и параллельный сбор данных с сетевого оборудования, а также более удобный динамический пользовательский web-интефейс про-

екта Wiremaps стали весомыми аргументами для выбора именно его в качестве основы для доработки в рамках поставленной задачи.

3 Решение задачи обнаружения узлов

Мы рассматриваем задачу обнаружения узлов в компьютерной сети, построенной на базе управляемых сетевых коммутаторов. Изначально мы должны знать список адресов всех управляемых коммутаторов, участвующих в построении рассматриваемой сети. Всё остальное оборудование, вовлечённое в сеть, является узлами сети, например, компьютеры или сетевые принтеры.

3.1 Сбор данных

Исходя из цели работы, сбор данных с сетевых коммутаторов должен осуществляться по протоколу Simple Network Management Protocol [10]. SNMP — это стандартный протокол, разработанный и используемый для управления оборудованием в IP сетях. Архитектура SNMP подразумевает существование управляемых и управляющих сторон. В состав управляемой системы (устройства) входит компонент, называемый агентом. Агент способен собирать различную информацию от устройства, котороя впоследствии может быть получена управляющей системой. Получение информации управляющей стороной от управляемой происходит через операции протокола GET, GETNEXT и GETBULK. На Рис. 1 изображена схема сбора данных управляющей системой (Wiremaps) с сетевых коммутаторов.

Переменные, доступные через SNMP, организованы в иерархию. Эта иерархия и другие метаданные (такие, как тип и описание переменной) описываются базами MIB (Management Information Base) [12]. Таким образом, SNMP не определяет, какую информацию управляемая система должна предоставлять. Базы MIB используют иерархическое пространство имен, содержащее уникальные идентификаторы объектов (object identifier, OID), каждый такой идентификатор однозначно определяет переменную, которая может быть прочитана или установлена через команды протокола SNMP. Возвращаемое значение (управляемый объект) может быть одного из двух видов: скалярное или таблица. Базы MIB используют нотацию, определённую в ASN.1 [11].

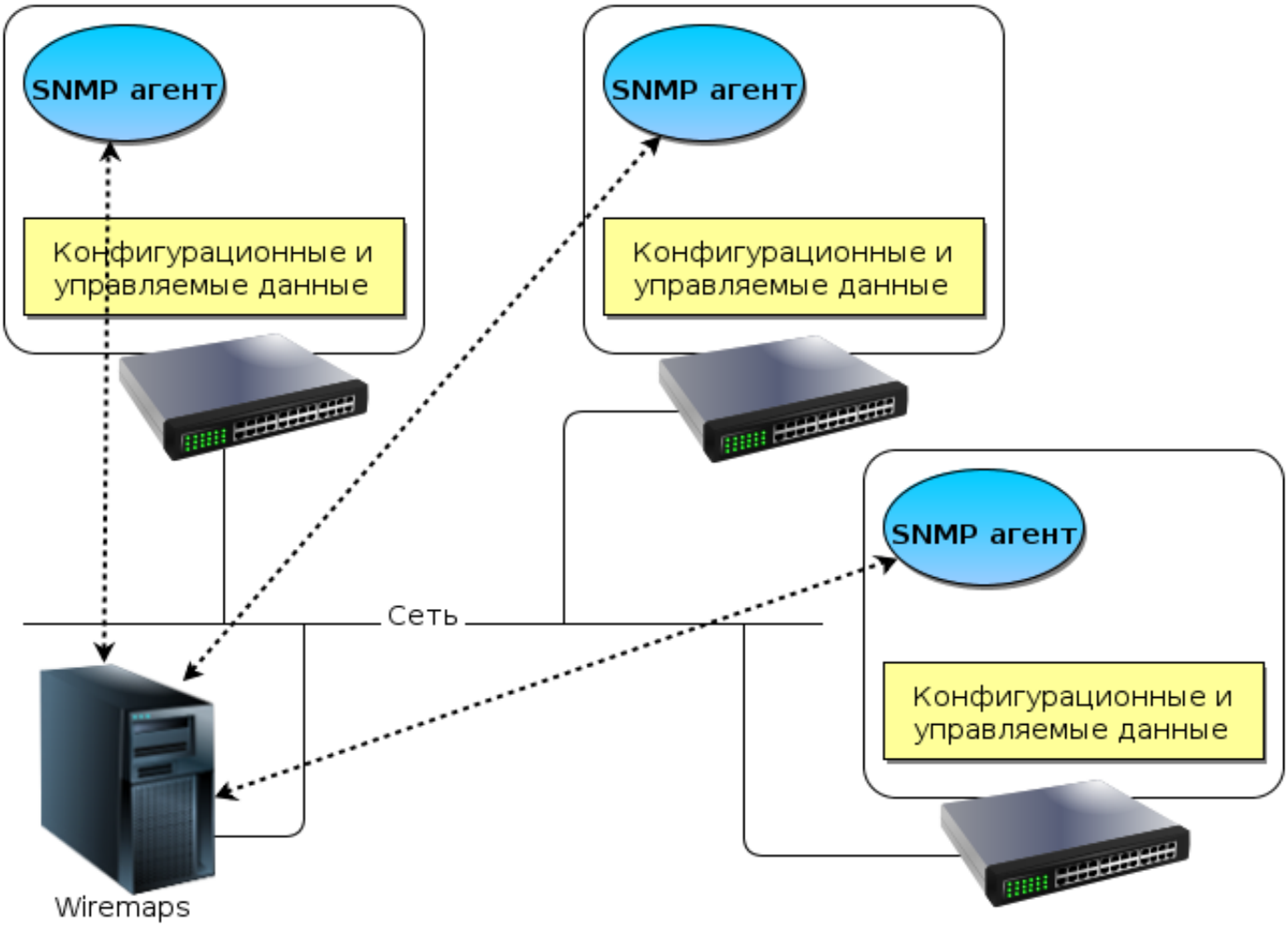

Рис. 1: Сбор данных управляющей системой (Wiremaps) с сетевого оборудования по протоколу SNMP

Прежде всего для решения нашей задачи необходима информация о подключенных устройсвах к каждому из коммутаторов. Данная информация может быть получена из таблиц коммутации (FDB - forwarding database) соответствующих устройств. Необходимые таблицы предоставляются следующими SNMP-объектами: `BRIDGE-MIB::dot1dTpFdbAddress` и `BRIDGE-MIB::dot1dTpFdbPort` [13]. Первый возвращает таблицу всех MAC-адресов, о которых есть информация о пересылке или фильтрации. Второй — таблицу сопоставления экземпляра из первой таблицы и номера порта, через который проходил пакет с адресом, совпадающим с соответствующим экземпляром `BRIDGE-MIB::dot1dTpFdbAddress`.

3.2 Поиск узла по MAC-адресу

Из полученных в предыдущем разделе таблиц коммутации мы можем получить информацию о том, через какие коммутаторы и соответствующие порты проходили пакеты от

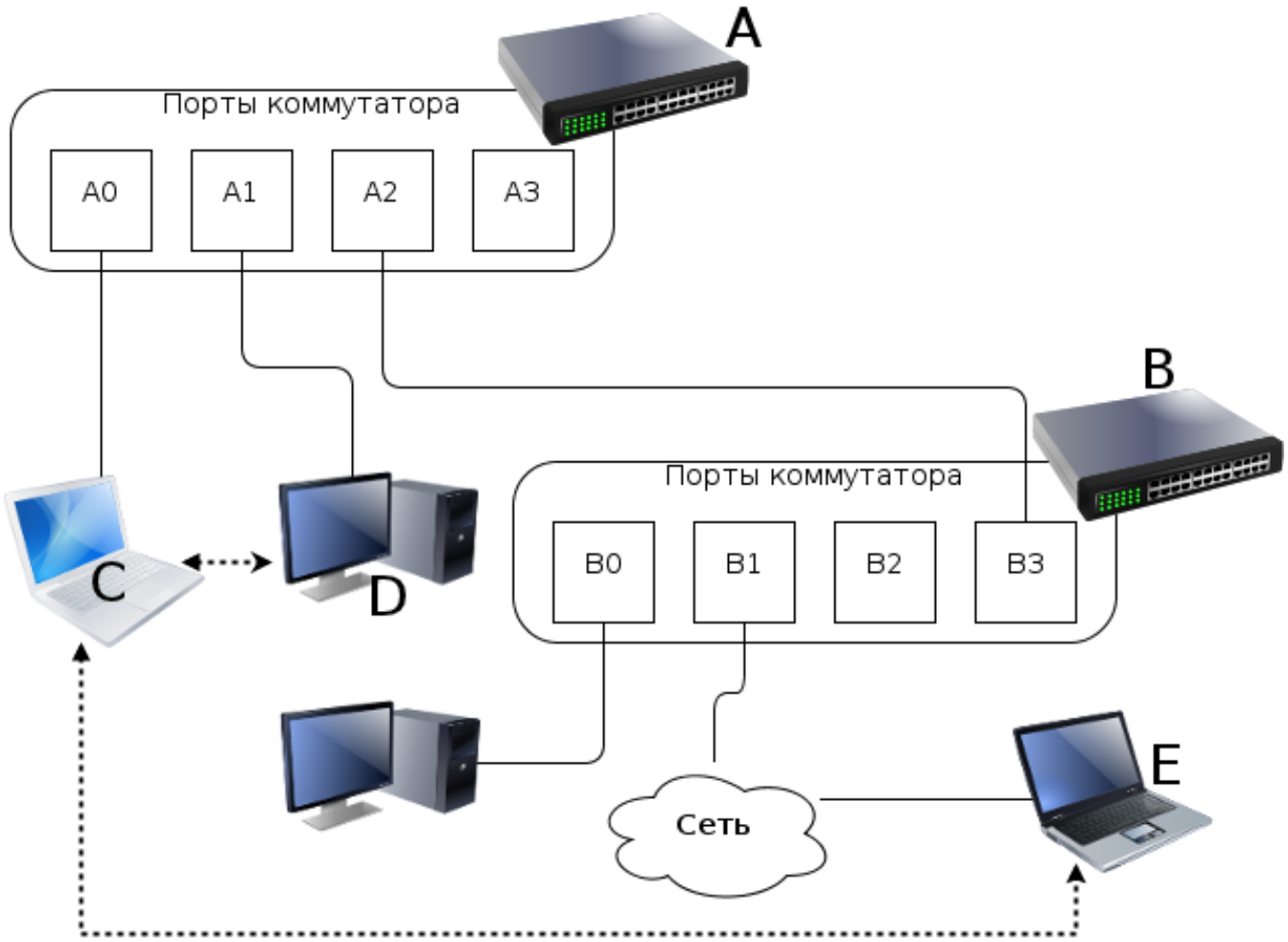

Рис. 2: Взаимодействие узлов в сегменте сети, состоящем из двух управляемых коммутаторов

заданного узла. Если узел за последние X минут (в зависимости от конфигурации коммутаторов) отправил хотя бы один пакет в сеть, то в таблице коммутации сетевого коммутатора, непосредственно к которому подключен узел, появится или обновится запись о MAC-адресе данного узла. Нельзя гарантировать, что список коммутаторов, через которые проходили пакеты от заданного узла, будет содержать ровно один элемент. Однако, если узел за последнее время проявлял какую-либо активность в сети, то этот список будет точно содержать элемент — искомый коммутатор и порт.

Развернув записи таблицы, полученные SNMP-объектом `dot1dTpFdbPort`, в обратном порядке, каждому порту мы имеем сопоставление списка MAC-адресов устройств, пакеты от которых проходили через данный коммутатор. Рассмотрим подробнее на примере, какие списки мы можем получить. На Рис. 2 изображен сегмент сети, состоящий из двух управляемых коммутаторов `A` и `B`. При взаимодействие в между собой в сети узлов `C` и `D`, список адресов порта $A0 = \{MAC(\texttt{C})\}$ и порта $A1 = \{MAC(\texttt{D})\}$, то есть получаем однозначное со-

ответствие узлов портам коммутатора. Давайте теперь рассмотрим взаимодействие узлов C и E. При отправке пакета узлом E узлу C, пакет приходит на коммутатор B и в соответствующей таблице коммутации устройства появляется запись для порта $B1$, то есть список адресов порта $B1 = \{MAC(\texttt{E}),\ \ldots\}$. Далее пакет от коммутатора B через порт $B3$ отправляется на порт $A2$ коммутатора A. После чего список сопоставления адресов порту $B1$ будет содержать MAC-адрес узла E, $B3$ — узла C, а список сопоставления порта $A2$ будет содержать MAC-адрес узла E. В итоге, рассматривая взаимодействие в сети всего двух узлов, мы получили следующие множества адресов, сопоставленные портам:

$$A0 = \{MAC(\texttt{C})\},\ A2 = \{MAC(\texttt{E}),\ \ldots\}$$

$$B1 = \{MAC(\texttt{E}),\ \ldots\},\ B3 = \{MAC(\texttt{C}),\ \ldots\}$$

Таким образом, мы видим неоднозначность в определении к какому непосредственно коммутатору подключен как узел C, так и узел E.

3.3 Определение топологии

В предыдущем разделе мы рассмотрели способ сопоставления узла порту коммутатора, к которому он подключен. Однако, данный способ является неоднозначным, если неизвестна топология сети. Чтобы добиться однозначности при поиске узлов в сети предложенным способом, мы не должны рассматривать порты, соединяющие непосредственно два коммутатора между собой. В рассмотренном выше примере таковыми являются порты $A2$ и $B3$. Убрав их из рассмотрения, мы получаем однозначность в сопоставлении портам коммутаторов узлов C и D.

Способ построения топологии коммутаторов, рассмотренный в курсовой работе «Анализ и построение структуры сети» [1], к сожалению, применим не в любых сетях. Вследствие чего, был разработан другой алгоритм, определяющий соединительные порты коммутаторов.

Воспользуемся тем фактом, что управляемые коммутаторы, как и остальные узлы сети, имеют свои уникальные MAC-адреса. Заголовки сетевого уровня для пакетов, посылаемых им, например, при SNMP запросах, ничем не отличаются от тех, которые адресованы любым другим узлам сети. Это означает, что при общении узла с агентами управляемых коммутаторов, их адреса сетевого уровня также заносятся в таблицы коммутации соответствующего оборудования.

Рассмотрим схему, изобаженную на Рис. 2, и предположим, что в нашем сегменте имеется еще третий управляемый коммутатор F, и порт $B2$ соединен с портом $F0$. Пусть узел C

будет являться управляющей системой. Он запрашивает данные от коммутатора `F`, например, SNMP-объект `sysDescr`, возвращаемый текстовое описание управляемого устройства. При такого рода общении в сети, в таблице коммутации всех промежуточных коммутаторов, ведущих от узла `C` к устройству `F`, появится запись, сопоставляющая MAC-адрес `F` соответствующему порту, через который направлялись пакеты к `F`. В свою очередь при ответном пакете от устройства `F` узлу `C` в таблицы коммутации промежуточных коммутаторов занесется запись о MAC-адресе узла `C`. То есть множества адресов, сопоставленные портам будут выглядеть следующим образом:

$$A0 = \{MAC(\texttt{C})\},\ A2 = \{MAC(\texttt{F}),\ \ldots\}$$

$$B2 = \{MAC(\texttt{F}),\ \ldots\},\ B3 = \{MAC(\texttt{C}),\ \ldots\}$$

$$F0 = \{MAC(\texttt{C}),\ \ldots\}$$

Если в списке сопоставленных MAC-адресов для конкретного порта встречается хотя бы один MAC-адрес управляемого коммутатора или MAC-адрес управляющего узла, то такой порт будем считать соединительным — непосредственно связывающим два коммутатора. В рассмотренном примере такими являются $A0$, $A2$, $B2$, $B3$ и $F0$.

Данный метод работает вне зависимости от месторасположения в топологии сети управляющего узла. Для появления необходимых данных в таблицах коммутации устройств и последующего точного определения всех портов, связывающих между собой коммутаторы, управляющему узлу необходимо опросить (обменяться пакетами) хотя бы все листья (устройства) в дереве коммутаторов. Однако, предложенный метод имеет и минус в том, что один из портов коммутатора, непосредственно к которому подключен управляющий узел, будет определен некорректно как соединительный.

4 Реализация в Wiremaps

4.1 Анализ архитектуры Wiremaps

Wiremaps написан на языку Python и построен на базе событийно-ориентированного сетевого фреймворка Twisted [9]. Данный фреймворк предоставляет возможность автоматического распараллеливания выполняющихся независимых процедур. Также предоставляет реализацию web-сервера, который в Wiremaps используется для обеспечения работы управляющего пользовательского web-интерфейса.

Базовым компонентом Wiremaps является класс `CollectorService`, наследующий базовый класс сервиса приложения фреймворка `twisted.application.service.Service`. Основная задача данного компонента — запуск сбора данных с заданного управляемого оборудования, а также запись полученной и обработаной информации в базу данный PostgreSQL.

Для сбора данных с устройств, в Wiremaps разработаны классы, реализующие общий интерфейс `ICollector`. Для большого числа оборудования от распространённых производителей, таких как 3Com, Hewlett-Packard, Cisco и Nortel разработаны отдельные классы-коллекторы, которые при сборе данных учитывают специфику устройства, модель, версию прошивки.

В пакете `helper` находятся компоненты-помощники, которые занимаются непосредственно сбором определенных данных с устройства, обращаясь к ним по SNMP. Например, класс `PortCollector` реализует получение только таблицы коммутации с заданного ему устройства, класс `ArpCollector` — соответствующей ARP-таблицы устройства, класс `LldpCollector` реализует сбор LLDP-информации, которой обладает заданное устройство. Экземпляры классов, реализующие интерфейс `ICollector`, используют эти компоненты-помощники внутри себя. Исходя из модели, производителя и поддерживаемых протоколов оборудования, коллекторы решают, какие компоненты-помощники использовать для сбора данных. Полученные и обработанные данные `CollectorService` отправляет экземпляру класса `DatabaseWriter`, который по каждому оборудованию заносит информацию в базу данных.

База данных Wiremaps содержит таблицу `equipment`, в которой хранятся адреса, имена, описания, даты создания и обновления управляемых устройств, с которых собираются данные. Также для каждого компонента-помощника имеется своя таблица, например, для компонентов `PortCollector`, `ArpCollector` и `CdpCollector` имеются таблицы `port`, `arp` и `cdp` соответственно. Для всех вышеперечисленных таблиц имеются также таблицы для ротации старых данных и хранения истории изменений. Эти таблицы созданы по той же схеме, а имена отличаются приписанным суффиксом `_past`.

4.2 Реализация метода определения топологии

Исходя из рассмотренной архитектуры проекта Wiremaps, автором курсовой работы был разработан новый класс компонент-помощник `Port2BridgeCollector`. Конструктор класса принимает параметры `equipment`, `proxy` и `equipProxies`. Объект `equipment` представляет экземпляр класса `Equipment` и соответственно устройство, для которого решается задача обнаружения соединительных портов. Объект `proxy` является экземпляром класса `AgentProxy`,

который предоставляет высокоуровневый интерфейс для работы по протоколу SNMP с нашим устройством. Параметр `equipProxies` содержит список экземпляров класса `AgentProxy`, указывающих на все известные управляемые устройства в рассматриваемой сети.

Как упоминалось в разделе 3.3, для появления необходимых данных в таблицах коммутации устройств, управляющему узлу необходимо обменяться пакетами хотя бы со всеми листьями (устройствами) в дереве коммутаторов. Так как мы не знаем топологию сети, то и идентифицировать, является ли данное управляемое устройство листовым в дереве связей, мы не можем. Для каждого устройства из списка `equipProxies` нам следует узнать его MAC-адрес, который предоставляется SNMP-объектом `dot1dBaseBridgeAddress`. Таким образом, получив MAC-адреса всех устройств из предоставленного списка, мы гарантированно обменялись пакетами со всеми коммутаторами, тем самым выполнили начальное условие реализуемого метода.

Далее, используя объект `proxy`, мы запрашиваем у нашего устройства содержание в его таблице коммутации (SNMP-объект `dot1dTpFdbPort`) информации о каждом из полученных MAC-адресов, а также о MAC-адресе управляющего узла, заданного в конфигурационном файле Wiremaps. Соответствующий номер порта для каждого найденного в таблице адреса мы заносим в список `equipment.ports2bridge`. Таким образом, после работы компонента-помощника `Port2BridgeCollector`, список `equipment.ports2bridge` будет содержать все порты, соединяющие непосредственно данный коммутатор с другим.

4.3 Запись в базу данных

Для хранения полученного списка портов, в базе данных была создана новая таблица `port2bridge`. Следуя общей архитектуре проекта, была также создана таблица `port2bridge_past` для ведения истории изменений. За основу схем новых таблиц была взята схема таблицы `lldp`. Таблица `port2bridge` содержит четыре поля: `equipment`, `port`, `created` и `deleted`, определяющие адрес устройства, порт, время создания и время удаления записи соответственно. Были также внесены изменения в реализацию класса `DatabaseWriter`, позволяющие записывать полученный порты из списка `equipment.ports2bridge` в новую созданную таблицу базы данных.

4.4 Пользовательский интерфейс

Для осуществления поиска узлов в сети был доработан пользовательский интефейс проекта Wiremaps, а именно создана отдельная страничка с относительным адресом `/discover`,

отличающаяся от основной только обработкой поискового запроса. В поле поиска пользователь может ввести доменное имя, IP или MAC-адрес желаемого узла. Система попытается определить, что именно из предложенного ввел пользователь, а также попытается получить MAC-адрес узла, если были заданы его доменное имя или IP-адрес с помощью DNS запросов или из `arp` таблицы базы данных соответственно. Если удалось получить MAC-адрес узла или он был введён непосредственно в поле поиска, системе Wiremaps будет послан AJAX запрос на определение месторасположения данного MAC-адреса в топологии сети. Получение ответа на данный запрос происходит путем формирования запроса к базе данных, а точнее к таблице `fdb`, содержащей таблицы коммутации всех управляемых устройств. При формировании данного запроса добавляется фильтрация портов, хранящихся в таблице `port2bridge`. Таким образом, если узел был найден в сети, пользователю выводится подробная информация об устройстве и соответствующем порту, к которому непосредственно подключен заданный узел. На Рис. 3 изображен пользовательский интерфейс Wiremaps с выполненным запросом обнаружения узла.

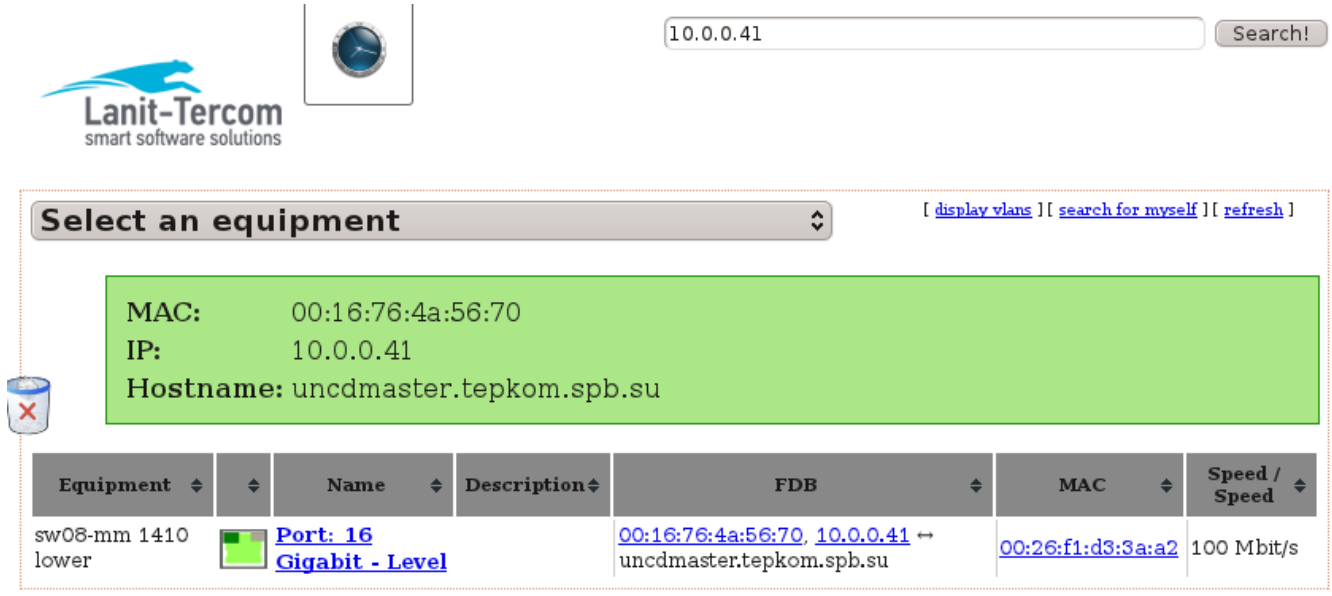

Рис. 3: Пользовательский интерфейс инструмента Wiremaps

5 Заключение

В рамках курсовой работы были достигнуты следующие результаты:

- проведён детальный обзор предметной области, готовых продуктов и используемых протоколов для решения поставленной задачи;
- разработан эвристический алгоритм обнаружения узлов в сети, не имеющий описанных недостатков существующих решений;
- реализован алгоритм обнаружения узлов в проекте Wiremaps;
- доработанный инструмент Wiremaps внедрён в инфраструктуру компании «Lanit-Tercom».

Список литературы

[1] Гущина В.М. Анализ и постоение структуры сети / Кафедра системного программирования Санкт-Петербургского Государственного университета — Санкт-Петербург, 2010.

[2] Barthel A. Analysis, Implementation and Enhancement of Vendor dependent and independent Layer-2 Network Topology Discovery / Chemnitz University of Technology — Chemnitz, Germany, 2005.

[3] Getting the Best from 3Com®Network Supervisor / Part No. DUA1510-0AAA04-PDF, 3Com Corporation — Santa Clara, California, 2001. URL: 2http://h20000.www2.hp.com/bc/docs/support/SupportManual/c02583701/c02583701.pdf (дата обращения: 31.05.2011).

[4] HP OpenView network node manager 6.4 and network node manager extended topology 2.0 / Hewlett-Packard Company, 2003. URL: http://www.openview.ru/products/nnm_et_pb_feb03.pdf (дата обращения: 31.05.2011).

[5] Bernat V. Wiremaps manual — 2011. URL: https://github.com/vincentbernat/wiremaps/wiki/Wirema manual (дата обращения: 31.05.2011).

[6] Miller E. et al Netdisco 1.1 - README / University of California Santa Cruz's NTS department — Santa Cruz, California, 2011. URL: http://www.netdisco.org/readme.html (дата обращения: 31.05.2011).

[7] IEEE Standard for Local and metropolitan area networks — Media Access Control (MAC) Bridges / IEEE — New York, USA, 2004. URL: http://standards.ieee.org/getieee802/download/802.1D-2004.pdf (дата обращения: 31.05.2011).

[8] IEEE Standard for Local and metropolitan area networks — Station and Media Access Control Connectivity Discovery / IEEE — New York, USA, 2009. URL: http://standards.ieee.org/getieee802/download/802.1AB-2009.pdf (дата обращения: 31.05.2011).

[9] Fettig A. Twisted Network Programming Essentials // O'Reilly Media; 1 edition, 2005. ISBN 0-596-10032-9.

[10] Case J., Fedor M., Schoffstall M., Davin J. A Simple Network Management Protocol (SNMP) / RFC 1157 — 1990. URL: http://www.faqs.org/rfcs/rfc1157.html (дата обращения: 31.05.2011).

[11] Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation / ITU-T X.680 — 2002. URL: http://www.itu.int/ITU-T/studygroups/com17/languages/X.680-0207.pdf (дата обращения: 31.05.2011).

[12] McCloghrie K., Rose M. Management Information Base for Network Management of TCP/IP-based internets: MIB-II / RFC 1213 — 1991. URL: http://tools.ietf.org/html/rfc1213 (дата обращения: 31.05.2011).

[13] Decker E., Langille P., Rijsinghani A., McCloghrie K. Definitions of Managed Objects for Bridges / RFC 1286 — 1991. URL: http://tools.ietf.org/html/rfc1286 (дата обращения: 31.05.2011).