УДК 004.4, 519.713.1 doi: 10.15827/0236-235X.142.034-042 2024. Т. 37. № 1. С. 34–42

Автоматическое конфигурирование маршрутизаторов для управления настройками сетевой инфраструктуры

Р.Р. Фаткиева [1]✉, А.С. Судаков [1]

[1] Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), г. Санкт-Петербург, 197022, Россия

Ссылка для цитирования
Фаткиева Р.Р., Судаков А.С. Автоматическое конфигурирование маршрутизаторов для управления настройками сетевой инфраструктуры // Программные продукты и системы. 2024. Т. 37. № 1. С. 34–42. doi: 10.15827/0236-235X.142.034-042
Информация о статье
Группа специальностей ВАК: 1.2.1, 2.3.5
Поступила в редакцию: 10.08.2023 После доработки: 04.12.2023 Принята к публикации: 18.12.2023

Аннотация. Современная сетевая инфраструктура включает в себя различные уровни и типы устройств, а также разнообразные протоколы и службы взаимодействия между ними. Это создает сложность в управлении и первоначальном конфигурировании устройств. При массовой настройке однотипных устройств увеличивается вероятность возникновения ошибки. Автоматическое конфигурирование большого количества сетевых устройств позволяет облегчить задачу администрирования, снизить вероятность ошибок и сократить время при развертывании большого количества узлов сети. В данной работе рассмотрены существующие подходы к конфигурированию сетевых устройств. Представлен разработанный метод автоматизации процесса конфигурирования маршрутизаторов с использованием теории конечных автоматов. Показано, что в контексте автоматической конфигурации маршрутизаторов конечные автоматы можно использовать для представления различных состояний и действий, которые могут возникнуть в процессе конфигурирования. Это позволяет разработать систему автоматизации процесса первоначальной настройки сетевой инфраструктуры. На базе представленного метода сформированы алгоритмы для автоматического обнаружения маршрутизаторов и их настройки, а также метод сбора информационных сообщений, появляющихся в процессе настройки устройства. Разработан программный комплекс в виде веб-приложения, позволяющий уменьшить время развертывания сетевой инфраструктуры. На практическом примере показана возможность поиска устройств в сети, анализа производителя оборудования по MAC-адресу, удаленного подключения к нему и автоматического конфигурирования маршрутизаторов компании MikroTik. Разработанный программный комплекс может быть использован для быстрой и удобной настройки маршрутизаторов в средних и больших организациях.

Ключевые слова: конфигурирование, маршрутизатор, конечный автомат, сетевая инфраструктура, настройки сети, алгоритм

Введение. Стремительное развитие интернет-технологий, увеличение числа подключенных устройств, а также усложнение сетевой инфраструктуры приводят к необходимости обработки большого объема как передаваемых данных, так и служебного трафика. В таких условиях подключение и конфигурирование маршрутизаторов являются трудоемкими процессами и могут привести к ошибкам, обусловленным человеческим фактором: настраиваемые вручную маршрутизаторы часто становятся источником ошибок в конфигурации. Это может стать причиной нарушения работы сети или уязвимости в безопасности.

Автоматизация подключения и конфигурации маршрутизаторов позволяет следующее:

– сократить время настройки, снизить вероятность возникновения ошибок и уменьшить усилия, затрачиваемые на управление сетевой инфраструктурой, что особенно актуально для компаний и организаций с динамичной сетевой инфраструктурой, в которых необходимо быстро внедрять изменения в сети или требуется управление большим числом устройств;

– расширить возможности управления за счет быстрой адаптации к постоянно меняющимся условиям, например, путем определения оптимальных маршрутов с изменением их в соответствии с текущей ситуацией, балансировки нагрузки, предотвращения петель и других мероприятий по оптимизации работы сети;

– сформировать одномоментное конфигурирование ко всем маршрутизаторам в сети, позволяющее обеспечить единые настройки на всех маршрутизаторах и тем самым повысить стабильность и надежность сети, что важно для обеспечения однородности работы сетевой инфраструктуры;

– повысить безопасность, поскольку автоматическое конфигурирование может обеспечить одномоментное применение стандартных

политик безопасности к маршрутизаторам, способствуя уменьшению рисков и улучшению защиты сети от угроз;

– использовать концепцию программно-определяемых сетей, тем самым создавая более гибкие управляемые и адаптивные сети.

В работах [1–3] показано, что автоматическое конфигурирование маршрутизаторов значительно упрощает управление сетевой инфраструктурой, обеспечивает высокую надежность и безопасность, позволяет более гибко адаптироваться к изменениям в сети и требованиям организации. В [4] предлагаются сценарии для настройки IP-адресов интерфейсов, однако рассмотрение ограничено применением протокола маршрутизации BGP. В [5, 6] представлены методы настройки сетевого оборудования с использованием скриптов, что ограничивает зону масштабирования.

В исследовании [7] для оптимизации работы сетей и обеспечения высокопроизводительной сетевой инфраструктуры предложен подход, основанный на использовании ПО, способного считывать текущее состояние сети, автоматически проверять и реконфигурировать сетевую систему.

В работе [8] предложен программируемый модуль обработки данных на основе конвейера реконфигурируемых таблиц соответствия для программно-определяемого управления сетевым трафиком. Однако этот модуль обеспечивает изменение сетевых интерфейсов только при наличии подключенного дополнительного оборудования.

Модель управления конфигурацией на основе онтологий для сетевых устройств, представленная в [9, 10], позволяет применять технологию адаптивных команд для разнородного оборудования, основанную на абстрактных атомарных операциях. Результаты показали, что данная модель повышает безопасность при эксплуатации сети, но требует использования специализированного ПО.

В [11] предложена система автоматической диагностики сетевой инфраструктуры, основанная на методах искусственного интеллекта. Система позволяет диагностировать неисправности и осуществлять реконфигурацию сети.

К основным проблемам конфигурирования маршрутизаторов традиционно относятся разработка правил формирования статической и динамической маршрутизации трафика, обеспечение политик сетевой безопасности и обнаружения уязвимостей. Для их формирования необходим ввод обширного набора настроек, что требует более тонкой настройки и конфигурирования. В работах [12–15] представлен подход к автоматизации конфигурации сети с использованием аппарата конечных автоматов, однако указанный подход требует определенных сетевых навыков, что затрудняет его применение.

Таким образом, разработка методов автоматического конфигурирования маршрутизаторов имеет актуальное значение для современных сетевых инфраструктур и важна для повышения эффективности и надежности сетей, а также обеспечения более гибкого и удобного управления ими.

Разработка метода автоматического конфигурирования маршрутизаторов

Разработка метода требует реализации механизма обнаружения новых устройств маршрутизации в сети, их подключения к системе настройки в зависимости от конкретных сетевых требований и используемых технологий, а также настройки устройств для передачи сетевого трафика. Оптимальным вариантом реализации такого механизма является клиент-серверное приложение. Его отличительная особенность в том, что оно устанавливается на операционную систему Linux, а доступ к нему для настройки сетевых устройств осуществляется через web-интерфейс. Это позволяет подключаться с любых устройств вне зависимости от операционной системы. Особенностью также является использование одинакового набора конфигурационных команд для массовой настройки устройств. Это дает возможность простого внесения изменений в эти команды, уникальных для каждого из устройств, что позволяет существенно уменьшить трудности в их конфигурировании.

Общая схема работы алгоритма автоматического конфигурирования маршрутизаторов будет состоять из описанных далее шагов.

Шаг 1. Сканирование подсети и определение множества элементов сетевого оборудования, подлежащих объединению в сетевую инфраструктуру:

$$S = \{s_1, s_2, ..., s_i, ..., s_n\}, \quad (1)$$

где s_i, $i = \overline{1, N}$ – элемент сетевого оборудования.

Для каждого элемента из множества (1) необходимо сформировать частную схему настройки, состоящую из определения следующих составляющих.

1.1. Тип анализируемого элемента сетевого оборудования.

1.2. Множество входных и выходных алфавитов сигналов, отвечающих за настройку характеристик сети, требований к конфигурированию маршрутизаторов и их безопасности, основанных на множестве входных $X = \{x_1, x_2, \ldots, x_i, \ldots, x_n\}$ и выходных $Y = \{y_1, y_2, \ldots, y_i, \ldots, y_n\}$ данных.

1.3. Множество состояний устройства:

$$C = \{c_1, c_2, \ldots, c_k, \ldots, c_M\}, \quad (2)$$

где c_k – состояние устройства.

1.4. Входящие в устройство элементарные передаточные функции, зависящие от входного воздействия и состояния устройства, например $x_i \xrightarrow{\varphi(x)} y_j$, где $\varphi(x)$ – передаточная функция. Совокупность передаточных функций определяет оптимальные настройки для каждого маршрутизатора на основе собранных данных о входящих сигналах, заданных правилах перехода из состояния в состояние. Используя данный подход, сетевое устройство, подлежащее конфигурированию, можно представить в виде конечного автомата [15], формально определенного в виде элементов некоторых множеств:

$S = (C, X, Y, \varphi)$,

а всю сеть – в виде множества конечных автоматов, сопряженных в единую функционирующую структуру, использование которой позволяет не только оперативно перестраивать структуру сети, но и осуществлять мониторинг и управление ее жизнедеятельностью.

1.5. Множество возможных передаточных функций для всех типов подключенных устройств. Это позволит сформировать в устройстве набор переходов состояний для возможных входных воздействий и определить правила типовой автоматической настройки для каждого класса устройств сетевого оборудования.

1.6. Множество управляющих воздействий для настройки сетевого оборудования.

Шаг 2. Конфигурирование и оценка работоспособности.

2.1. Подключение к устройству для его настройки.

2.2. Выбор передаточных функций из множества, найденного на шаге 1 (1.5).

2.3. Формирование набора команд для настройки функционирования передаточных функций.

2.4. Настройка функционирования передаточных функций (например, правил фильтрации трафика).

2.5. Тестирование работы передаточной функции при подключении устройства к сети.

2.6. Формирование коррекции работы передаточной функции в случае некорректной работы на шаге 2 (п. 2.3).

Шаг 3. Синтез сетевой инфраструктуры. Для этого целесообразно определить множество связей, которые необходимо сформировать для построения всей сетевой инфраструктуры элементов из множества (1).

3.1. Выбор устройств из множества (1).

3.2. Сопряжение входов/выходов устройств, выбранных на шаге 2 (п. 2.1).

3.3. Определение передаточной функции для связи устройств и их характеристик (протокол, вид связи, физическое подключение и т.п.).

3.4. Формирование множества управляющих воздействий для настройки управления потоками трафика в сети. Это дает возможность контролировать прохождение трафика через маршрутизаторы, оптимизируя его путь и минимизируя задержки.

Представленный метод автоматического конфигурирования маршрутизаторов основан на централизованном управлении сетью. Он позволяет упростить процесс конфигурирования и обеспечить более высокую гибкость и надежность работы сети за счет настройки на основе актуальной информации о состоянии сети. Это снижает риски ошибок, связанных с ручным настройками, и улучшает стабильность работы сети.

Практические результаты

Предложенный метод применялся при построении логической сетевой инфраструктуры информационно-вычислительной системы и выполнялся на виртуальном стенде в среде VirtualBox с использованием эмулятора EVE-NG 5.0.1-106. Для моделирования конфигурирования маршрутизаторов были выбраны виртуальные маршрутизаторы фирмы Mikrotik с операционной системой Router OS версии 6.40.4. Для их настройки использовалось следующее ПО: клиент для протоколов удаленного доступа PuTTy и утилита Winbox для подключения, управления и мониторинга маршрутизаторов Mikrotik.

На первом этапе была сформирована логическая схема реализации ПО для настройки конфигурации маршрутизатора (шаг 1), представленная на рисунке 1.

Это позволило сформировать логическую схему настройки маршрутизатора в виде на-

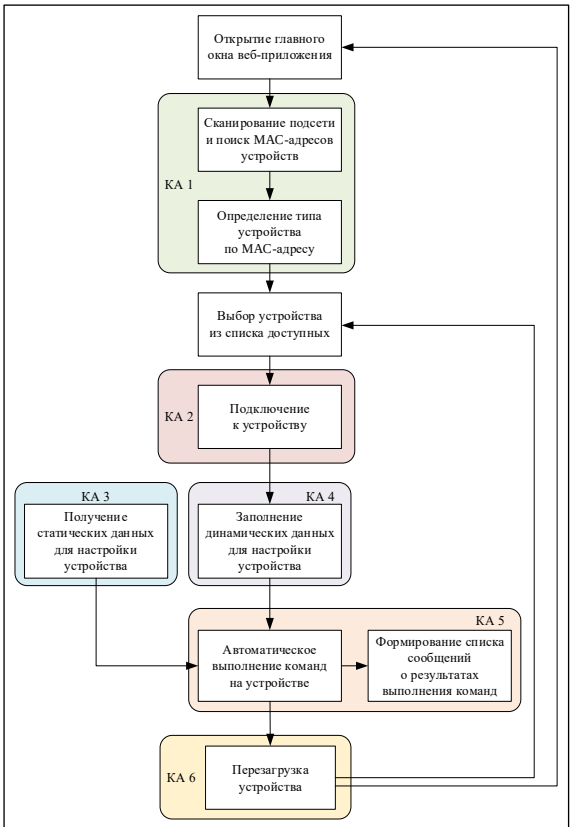

Рис. 1. Логическая схема настройки маршрутизатора

Fig. 1. Logic diagram of router configuration

бора моделей конечных автоматов, а также разработать приложение для автоматизации конфигурации маршрутизаторов. В набор моделей конечных автоматов входят перечисленные далее подсистемы.

• ***Подсистема сканирования подсети*** (рис. 2), состоящая из

– входных элементов $x = \{x_1, x_2, x_3\}$, где x_1 – конфигурационные элементы адреса подсети, x_2 – конфигурационные элементы маски подсети, x_3 – данные по MAC-адресам;

– состояний $c = \{c_1, c_2, c_3, c_4, c_5\}$, где c_1 – начальное состояние устройства, c_2 – получение значений вендоров по MAC-адресам, c_3 – сканирование всего диапазона подсети, c_4 – сопоставление MAC-адреса хоста с информацией в файле с MAC-адресами, c_5 – добавление информации о хосте в массив выводимой информации;

– передаточных функций, состоящих из $c_1 = \xrightarrow{\varphi(x_1, x_2)} c_3$, $\varphi(x_1, x_2)$ – передаточная функция, определяющая диапазон сканирования сети; $c_2 = \xrightarrow{\varphi(x_3)} c_4$, $\varphi(c_3)$ – передаточная функция, определяющая соответствие MAC-адреса и вендора; $c_2, c_3 = \xrightarrow{\varphi(c_2, c_3)} c_4$, $\varphi(c_2, c_3)$ – передаточная функция, сканирующая заданный диапазон сети и определяющая вендор оборудования по MAC-адресу; $c_4 = \xrightarrow{\varphi(c_4)} c_5$, $\varphi(c_4)$ – передаточная функция, определяющая массив результатов сканирования сети; $c_5 = \xrightarrow{\varphi(c_5)} y_1$, y_2, …, y_n, $\varphi(c_5)$ – передаточная функция, формирующая массив результатов сканирования сети;

– выходных элементов $y = \{y_1, y_2, \ldots, y_n\}$, где $y_1, y_2, \ldots, y_n$ – множество найденных хостов в сети.

• ***Подсистема подключения к устройству*** (рис. 3), состоящая из

– входных элементов $x = \{x_1, x_2\}$, где x_1 – адрес устройства и порт подключения, x_2 – данные пользователя (логин и пароль);

– состояний $c = \{c_1, c_2, c_3, c_4\}$, где c_1 – начальное состояние устройства, c_2 – установка соединения по протоколу SSH, c_3 – авторизация на устройстве, c_4 – открытие сессии SSH;

– передаточных функций, состоящих из $c_1 = \xrightarrow{\varphi(x_1)} c_2$, $\varphi(x_1)$ – передаточная функция, устанавливающая SSH-соединение; $c_2 = \xrightarrow{\varphi(x_2, c_1)} c_3$, $\varphi(x_2, c_1)$ – передаточная функция авторизации на устройстве; $c_3 = \xrightarrow{\varphi(c_2)} c_4$, $\varphi(c_2)$ – передаточная функция, создающая SSH-сессию, для обмена информацией; $c_4 = \xrightarrow{\varphi(c_3)} y_1$, $\varphi(c_3)$ – передаточная функция, переводящая устройство в режим приема команд конфигурации;

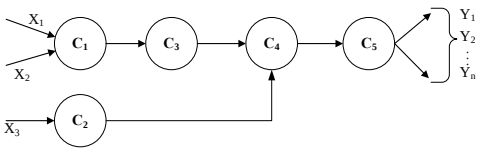

Рис. 2. Графическое представление конечного автомата «Сканирование подсети»

Fig. 2. Graphical representation of finite automata "Subnet scan"

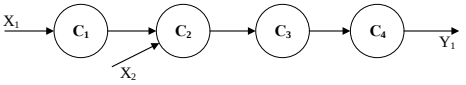

Рис. 3. Графическое представление конечного автомата «Подключение к устройству»

Fig. 3. Graphical representation of finite automata "Connecting to device"

– выходного измененного состояния устройства y_1.

• Подсистема формирования списка статических команд для выполнения на устройстве (рис. 4), состоящая из

– входного элемента x_1, представляющего собой изначальные конфигурационные элементы;

– состояний $c = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$, где c_1 – начальное состояние устройства, c_2 – проверка наличия конфигурационного файла, c_3 – проверка наличия файла констант, c_4 – получение пути к конфигурационному файлу, c_5 – получение пути к файлу констант, c_6 – форматирование пути к конфигурационному файлу, c_7 – форматирование пути к файлу констант, c_8 – формирование списка статических команд из конфигурационного файла и файла констант;

– передаточных функций, состоящих из $c_1 \xrightarrow{\varphi_1(x_1)} c_2$, $\varphi_1(x_1)$ – передаточная функция, определяющая наличие конфигурационного файла; $c_1 \xrightarrow{\varphi_2(x_1)} c_3$, $\varphi_2(x_1)$ – передаточная функция, определяющая наличие файла констант; $c_2 \xrightarrow{\varphi(c_2)} c_4$, $\varphi(c_2)$ – передаточная функция, изменяющая конфигурационные элементы файла конфигурации; $c_3 \xrightarrow{\varphi(c_3)} c_5$, $\varphi(c_3)$ – передаточная функция, изменяющая конфигурационные элементы файла с константами; $c_4 \xrightarrow{\varphi(c_4)} c_6$, $\varphi(c_4)$ – передаточная функция, форматирующая полученный путь для конфигурационного файла; $c_5 \xrightarrow{\varphi(c_5)} c_7$, $\varphi(c_5)$ – передаточная функция, форматирующая полученный путь для файла с константами; $c_6 \xrightarrow{\varphi(c_6)} c_8$, $\varphi(c_6)$ – передаточная функция, загружающая данные на устройство из конфигурационного файла с файлом констант; $c_7 \xrightarrow{\varphi(c_7)} c_8$, $\varphi(c_7)$ – передаточная функция, которая подключается к устройству по sftp и загружает данные из конфигурационного файла с файлом констант; $c_8 \xrightarrow{\varphi(c_8)} y_1$, $\varphi(c_8)$ – передаточная функция, формирующая список статических команд;

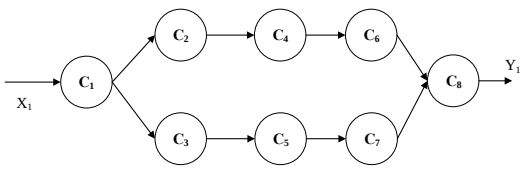

Рис. 4. Графическое представление конечного автомата «Создание списка статических команд»

Fig. 4. Graphical representation of finite automata "Creating a list of static commands"

– выходного измененного конфигурационного элемента y_1.

• Подсистема формирования списка динамических команд из данных, полученных с формы (рис. 5), состоящая из

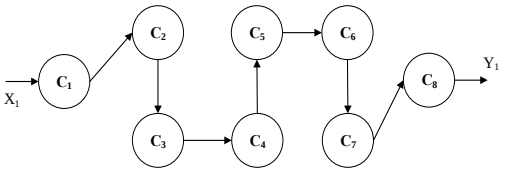

Рис. 5. Графическое представление конечного автомата «Создание списка динамических команд»

Fig. 5. Graphical presentation of finite automata "Creating a list of dynamic commands"

– входного элемента x_1, представляющего собой изначальные конфигурационные элементы;

– состояний $c = \{c_1, c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$, где c_1 – начальное состояние устройства, c_2 – получение адреса устройства, c_3 – проверка и получение имени устройства, c_4 – проверка и получение данных для настройки внешнего интерфейса, c_5 – получение данных для настройки DNS, c_6 – получение данных для настройки маршрута по умолчанию, c_7 – получение данных для настройки локального интерфейса, c_8 – формирование списка динамических команд из веб-формы;

– передаточных функций, состоящих из $c_1 \xrightarrow{\varphi(x_1)} c_2$, $\varphi(x_1)$ – передаточная функция, определяющая адрес устройства; $c_2 \xrightarrow{\varphi(c_1)} c_3$, $\varphi(c_1)$ – передаточная функция, добавляющая адрес устройства; $c_3 \xrightarrow{\varphi(c_2)} c_4$, $\varphi(c_2)$ – передаточная функция, добавляющая команду изменения имени устройства, если данные введены верно; $c_4 \xrightarrow{\varphi(c_3)} c_5$, $\varphi(c_3)$ – передаточная функция, добавляющая команду изменения внешнего интерфейса, если данные введены верно; $c_5 \xrightarrow{\varphi(c_4)} c_6$, $\varphi(c_4)$ – передаточная функция, добавляющая команду изменения сервера DNS, если данные введены верно; $c_6 \xrightarrow{\varphi(c_5)} c_7$, $\varphi(c_5)$ – передаточная функция, добавляющая команду изменения маршрута по умолчанию, если данные введены верно; $c_7 \xrightarrow{\varphi(c_6)} c_8$, $\varphi(c_6)$ – передаточная функция,

добавляющая команду изменения локального интерфейса, если данные введены верно; $c_8 \xrightarrow{\varphi(c_7)} y_1$, $\varphi(c_7)$ – передаточная функция, формирующая список динамических команд;

– выходного измененного конфигурационного элемента y_1.

• Подсистема выполнения команд конфигурации на устройстве (рис. 6), состоящая из

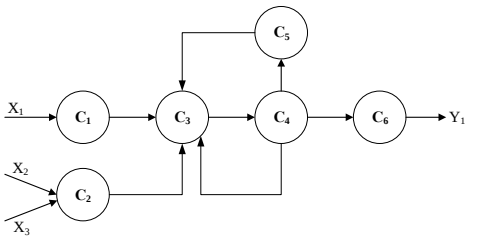

Рис. 6. Графическое представление конечного автомата «Выполнение команд конфигурации на устройстве»

Fig. 6. Graphical presentation of finite automata "Device configuration process"

– входных элементов $x = \{x_1, x_2, x_3\}$, где x_1 – устройство с установленным SSH-соединением, x_2 – массив статических команд, x_3 – массив динамических команд с формы настройки;

– состояний $c = \{c_1, c_2, c_3, c_4, c_5, c_6\}$, где c_1 – начальное состояние устройства, c_2 – массив команд для выполнения на устройстве, c_3 – отправка команды на устройство, c_4 – получение сообщения о выполнении команды, c_5 – запись информации в лог в случае отличия кода ошибки от 0, c_6 – закрытие SSH-канала;

– передаточных функций, состоящих из $c_1 \xrightarrow{\varphi(x_1)} c_3$, $\varphi(x_1)$ – передаточная функция, выполняющая команду на устройстве; $c_2 \xrightarrow{\varphi(x_3, x_4)} c_3$, $\varphi(x_3, x_4)$ – передаточная функция, которая, получив массив статических и динамических команд, отправляет их построчно для выполнения на устройстве; $c_3 \xrightarrow{\varphi(c_1, c_2)} c_4$, $\varphi(c_1, c_2)$ – передаточная функция, получающая результат выполнения команды; $c_4 \xrightarrow{\varphi(c_3)} c_5$, $\varphi(c_3)$ – передаточная функция, формирующая сообщение об ошибке; $c_4 \xrightarrow{\varphi(c_3)} c_3$, $\varphi(c_3)$ – передаточная функция, возвращающая в предыдущее состояние для выполнения очередной команды в случае отсутствия ошибки; $c_4 \xrightarrow{\varphi(c_3)} c_6$, $\varphi(c_3)$ – передаточная функция, разрывающая SSH-соединения в случае отсутствия команды на выполнение;

– выходного настроенного устройства y_1.

• Подсистема перезагрузки устройства (рис. 7), состоящая из

– входного элемента x_1, представляющего собой данные об устройстве (адрес устройства и порт подключения);

– состояний $c = \{c_1, c_2, c_3\}$, где c_1 – начальное состояние устройства, c_2 – запуск перезагрузки, c_3 – отправка сигнала об успешной перезагрузке;

– передаточных функций, состоящих из $c_1 \xrightarrow{\varphi(x_1)} c_2$, $\varphi(x_1)$ – передаточная функция, определяющая адрес устройства для перезагрузки; $c_2 \xrightarrow{\varphi(c_1)} c_3$, $\varphi(c_1)$ – передаточная функция, отправляющая команду перезагрузки на устройство; $c_3 \xrightarrow{\varphi(c_2)} y_1$, $\varphi(c_2)$ – передаточная функция, отправляющая сигнал об успешной перезагрузке;

– выходных данных об изменении устройства y_1.

После развертывания и запуска приложение доступно через веб-браузер. На его главной странице доступна форма для ввода адреса сети, по которой будет произведено сканирование, с целью определения доступных для настройки устройств Mikrotik (http://www.swsys.ru/uploaded/image/2024-1/5.jpg). По окончании сканирования результаты выводятся на отдельную вкладку в таблицу с указанием адреса устройства, MAC-адреса и производителя оборудования, который определяется по предварительно загруженной базе MAC-адресов.

После завершения сканирования можно выбрать необходимый маршрутизатор Mikrotik и произвести его настройку. Настройка осуществляется путем выполнения команд на маршрутизаторе, к которому происходит подключение по протоколу SSH. Команды формируются из статического файла, предварительно загруженного на сервер, и параметров, вводимых на отдельной динамической форме.

В процессе настройки маршрутизатора формируется подробный файл сообщений, куда записываются все выполненные команды и воз-

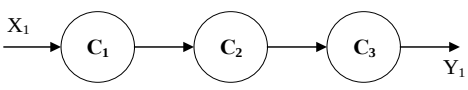

Рис. 7. Графическое представление конечного автомата «Перезагрузка устройства»

Fig. 7. Graphical presentation of finite automata "Device reboot"

никающие ошибки. По окончании настройки устройство перезагружается и становится готовым для дальнейшей эксплуатации.

Набор команд, задаваемых в статическом файле, может быть абсолютно любым, но идентичным для всех настраиваемых устройств. Также в статическом файле конфигурации используются данные из файла констант, которые позволяют задавать такие переменные, как логин и пароль, адрес центрального маршрутизатора и адрес удаленной сети. В частном случае примера при помощи статического файла конфигурации была произведена настройка параметров учетных данных для доступа на устройство виртуальной частной сети, iptables, NAT. Данные, которые вводятся на динамической форме, уникальны для каждого устройства. Это такие параметры, как имя устройства, настройки внешнего и внутренних интерфейсов, DNS и маршрутизация.

При любой настройке сетевого устройства в рамках корпоративной сети существуют общие настройки для всех устройств сети и индивидуальные настройки для каждого устройства. В статье приведен пример минимальных индивидуальных настроек сетевого устройства. В случае необходимости их набор и количество могут быть изменены, но пока только с помощью редактирования программного кода. Изменить их в пользовательском интерфейсе невозможно. Этот функционал можно рассмотреть в рамках развития данного метода.

Обсуждение

Использование разработанных подсистем позволяет не только обнаруживать и подключать новые устройства в сети, но и способствует более простому масштабированию и расширению сетевой инфраструктуры, имеющей в своем составе достаточное количество однотипных устройств. В этом случае при оценке эффективности применяемого метода целесообразно ограничиться двумя характеристиками: временем, затраченным персоналом на изменение конфигураций этих устройств вручную, в сравнении со временем, потраченным при использовании данного программного комплекса, и количеством подключенных однотипных устройств. Практика показывает, что в сетевой инфраструктуре при использовании метода должно быть как минимум от 5 до нескольких десятков однотипных устройств, работающих в диапазоне стандартных протоколов передачи данных, стека протоколов TCP/IP. Указанные ограничения можно отнести и к конфигурированию сетевой инфраструктуры при разработке политики безопасности. Однако следует учитывать, что при ограниченном количестве однотипных устройств эффективность метода снижается за счет применения в том числе статических методов, которые позволяют обеспечить более тонкую настройку сетевой инфраструктуры и политик безопасности в ручном режиме. В этом случае возникает необходимость расширения стандартного функционала конечного автомата, что влечет за собой увеличение количества входных воздействий и переходных состояний автомата, введение аппарата обратной связи, а также приводит к снижению надежности и отказоустойчивости разрабатываемой системы или увеличению ее сложности, что зачастую для небольших сетей, использующих статическую маршрутизацию, экономически невыгодно, но экономит значительные ресурсы в условиях необходимости масштабирования однотипной сетевой инфраструктуры. Это подтверждается тестированием, которое показало, что при конфигурировании одного устройства предложенным методом увеличивается время конфигурации в пределах 50–75 % (зависит от квалификации инженера и сложности конфигурации) от времени, заданного нормативами обслуживания сетевой инфраструктуры. Однако при увеличении количества конфигурируемых устройств время сокращается на 30–50 % от нормативного показателя.

Существующие аналогичные решения, представленные на рынке (ansible, chef, puppet), устраняют схожие проблемы автоматизации конфигурирования, однако больше ориентированы на масштабирование и поддержку серверной инфраструктуры и требуют определенных навыков для первоначальной адаптации и дальнейшего сопровождения. Наиболее приближенным по функционалу является ПО ansible, однако в нем отсутствует автоматический поиск для подключения к сетевым устройствам. Указанные программные средства также требуют определенных сетевых навыков и знания языка программирования для их использования в отличие от предложенного подхода.

В рамках данного ограниченного исследования были рассмотрены конфигурации одной виртуальной подсети, дальнейшей проработке подлежат вопросы интероперабельности в сетевую инфраструктуру удаленных устройств и их масштабирования. Однако применение предложенного подхода может быть затруд-

нено для сетевой инфраструктуры с различными производителями и используемыми протоколами передачи данных. В этом случае для устранения данного ограничения целесообразно использовать аппарат кусочно-линейных агрегатов с общим пулом данных о характеристиках сетевого оборудования, который позволяет переиспользовать уже разработанные конечные автоматы.

Заключение

Применение представленного метода автоматического конфигурирования маршрутизаторов может значительно упростить и улучшить процессы управления сетью, обеспечивая оптимальную производительность и удовлетворение потребностей современных сетевых сред. К дальнейшему направлению работ целесообразно отнести исследования различных устройств и типов маршрутизаторов, что может способствовать развитию более эффективных и универсальных методов автоматического конфигурирования сетевых устройств. Это обусловит более надежную и гибкую управляемость сетевой инфраструктуры. Однако следует отметить, что критическим аспектом при реализации метода является безопасность, поэтому также необходимо проработать механизмы аутентификации и авторизации настраиваемых сетевых устройств для предотвращения несанкционированного доступа к системам управления и защите сетевой инфраструктуры от внешних угроз.

Список литературы

1. Muthu T.S., Kamal S., Visalaxi S. Automatic selection of routers using YANG tool. Proc. ICAMCCT 2021, 2022, vol. 2385, no. 1, art. 050009. doi: 10.1063/5.0071070.
2. Csengody Z., Macko D., Jelemenska K. Automated evaluation of network device configuration. Proc. ICETA, 2018, pp. 99–104. doi: 10.1109/ICETA.2018.8572175.
3. Boskov I., Vucnik M., Fortuna C., Mohorcic M. Automated initial configuration of wireless embedded devices in the internet of things. Proc. BalkanCom, 2019, pp. 1–5.
4. Datta A., Asif Imran A.T.M., Biswas Ch. Network automation: Enhancing operational efficiency across the network environment. ICRRD J., 2023, vol. 4, no. 1, pp. 101–111. doi: 10.53272/icrrd.v4i1.1.
5. Elezi A., Karras D. On automating network systems configuration management. CRJ, 2023, vol. 1, no. 2, pp. 18–31. doi: 10.59380/crj.v1i1.639.
6. Ou B., Dong P. Automatically configuring the IP addresses for mobile network in collaborative network. Proc. ITOEC, 2020, pp. 1020–1025. doi: 10.1109/ITOEC49072.2020.9141797.
7. Simunic I., Grgurevic I. Automation of network device configuration using zero-touch provisioning. In: LNICST. Proc. FABULOUS, 2021, vol. 382, pp. 105–119. doi: 10.1007/978-3-030-78459-1_8.
8. Liu Z., Lv G., Wang J., Yang X. Domain-specific programming router model. emerging networking architecture and technologies. In: CCIS. Proc. ICENAT, 2023, vol. 1696, pp. 26–37. doi: 10.1007/978-981-19-9697-9_3.
9. Li D., Chen Z., Wang G., Zhou X., Ren M. The research on operation automation model for information equipment. J. Phys.: Conf. Ser., 2021, vol. 2095, art. 012048. doi: 10.1088/1742-6596/2095/1/012048.
10. Carvalho P., Lima S.R., Sabucedo L.Á., Santos-Gago J.M., Silva J.M.C. Towards a holistic semantic support for context-aware network monitoring. Computing, 2020, vol. 102, pp. 2565–2585. doi: 10.1007/s00607-020-00840-7.
11. Bideh E., Amiria M.F., Vahidib J., Iranmaneshc M. Automatic fault diagnosis of computer networks based on a combination BP neural network and fuzzy logic. IJNAA, 2022, pp. 1–12.
12. Zhang Z., Xia C., Chen S., Yang T., Chen Z. Reachability analysis of networked finite state machine with communication losses: A switched perspective. IEEE JSAC, 2020, vol. 38, no. 5, pp. 845–853. doi: 10.1109/JSAC.2020.2980920.
13. Leivadeas A., Falkner M. A survey on intent-based networking. IEEE Communications Surveys & Tutorials, 2023, vol. 25, no. 1, pp. 625–655. doi: 10.1109/COMST.2022.3215919.
14. Shu Z., Yan G. IoTInfer: Automated blackbox fuzz testing of IoT network protocols guided by finite state machine inference. IEEE Internet of Things J., 2022, vol. 9, no. 22, pp. 22737–22751. doi: 10.1109/JIOT.2022.3182589.
15. Акинина Ю.С., Тюрин С.В. Теория автоматов. М.: Ай Пи Ар Медиа, 2023. 156 с.

Software & Systems doi: 10.15827/0236-235X.142.034-042 2024, 37(1), pp. 34–42

Method of automatic configuration of routers

Roza R. Fatkieva [1]✉, Anton S. Sudakov [1]

[1] St. Petersburg Electrotechnical University "LETI", St. Petersburg, 197022, Russian Federation

For citation
Fatkieva, R.R., Sudakov, A.S. (2024) 'Method of automatic configuration of routers', *Software & Systems*, 37(1), pp. 34–42 (in Russ.). doi: 10.15827/0236-235X.142.034-042

Article info
Received: 10.08.2023 After revision: 04.12.2023 Accepted: 18.12.2023

Abstract. Modern network infrastructure includes various layers and types of devices, as well as a variety of protocols and services for interaction between devices. This complicates their management and first-time configuration. Mass configuration of the same type of devices enhances the likelihood of errors during their configuration. Automatic configuration of network devices facilitates the administration task, reduces the likelihood of errors and time for deploying a large number of network nodes. The paper considers existing approaches to configuring network devices. It presents a method of automating the process of configuring routers using the theory of finite automata. It is shown that in terms of automatic configuration of routers, finite automata can be used to represent various states and actions that may occur during a configuration process. This allows developing an automated system responding to changes and events in the network, adapting to new conditions. The presented method is a base for algorithms of automatic detection of routers, their configuration and a method for collecting information messages that occur during device configuration. There is also a developed software package represented by a web application, which allows reducing the time of network infrastructure deployment. A practical example shows the ability to search for devices in the network, to analyze the equipment manufacturer by MAC address, remotely connect to it and automatically configure MikroTik routers. The developed software package can be used for quick and easy configuration of routers in medium and large organizations.
Keywords: configuration, routers, state machine, network infrastructure, network settings, algorithm

References

1. Muthu, T.S., Kamal, S., Visalaxi, S. (2022) ‘Automatic selection of routers using YANG tool’, *Proc. ICAMCCT 2021*, 2385(1), art. 050009. doi: 10.1063/5.0071070.
2. Csengody, Z., Macko, D., Jelemenska, K. (2018) ‘Automated evaluation of network device configuration’, *Proc. ICETA*, pp. 99–104. doi: 10.1109/ICETA.2018.8572175.
3. Boskov, I., Vucnik, M., Fortuna, C., Mohorcic, M. (2019) ‘Automated initial configuration of wireless embedded devices in the internet of things’, *Proc. BalkanCom*, pp. 1–5.
4. Datta, A., Asif Imran, A.T.M., Biswas, Ch. (2023) ‘Network automation: Enhancing operational efficiency across the network environment’, *ICRRD J*., 4(1), pp. 101–111. doi: 10.53272/icrrd.v4i1.1.
5. Elezi, A., Karras, D. (2023) ‘On automating network systems configuration management’, *CRJ*, 1(2), pp. 18–31. doi: 10.59380/crj.v1i1.639.
6. Ou, B., Dong, P. (2020) ‘Automatically configuring the IP addresses for mobile network in collaborative network’, *Proc. ITOEC*, pp. 1020–1025. doi: 10.1109/ITOEC49072.2020.9141797.
7. Simunic, I., Grgurevic, I. (2021) ‘Automation of network device configuration using zero-touch provisioning’, in *LNICST. Proc. FABULOUS*, 382, pp. 105–119. doi: 10.1007/978-3-030-78459-1_8.
8. Liu, Z., Lv, G., Wang, J., Yang, X. (2023) ‘Domain-specific programming router model. emerging networking architecture and technologies’, in *CCIS. Proc. ICENAT*, 1696, pp. 26–37. doi: 10.1007/978-981-19-9697-9_3.
9. Li, D., Chen, Z., Wang, G., Zhou, X., Ren, M. (2021) ‘The research on operation automation model for information equipment’, *J. Phys.: Conf. Ser.*, 2095, art. 012048. doi: 10.1088/1742-6596/2095/1/012048.
10. Carvalho, P., Lima, S.R., Sabucedo, L.Á., Santos-Gago, J.M., Silva, J.M.C. (2020) ‘Towards a holistic semantic support for context-aware network monitoring’, *Computing*, 102, pp. 2565–2585. doi: 10.1007/s00607-020-00840-7.
11. Bideh, E., Amiria, M.F., Vahidib, J., Iranmaneshc, M. (2022) ‘Automatic fault diagnosis of computer networks based on a combination BP neural network and fuzzy logic’, *IJNAA*, pp. 1–12.
12. Zhang, Z., Xia, C., Chen, S., Yang, T., Chen, Z. (2020) ‘Reachability analysis of networked finite state machine with communication losses: A switched perspective’, *IEEE JSAC*, 38(5), pp. 845–853. doi: 10.1109/JSAC.2020.2980920.
13. Leivadeas, A., Falkner, M. (2023) ‘A survey on intent-based networking’, *IEEE Communications Surveys & Tutorials*, 25(1), pp. 625–655. doi: 10.1109/COMST.2022.3215919.
14. Shu, Z., Yan, G. (2022) ‘IoTInfer: Automated blackbox fuzz testing of IoT network protocols guided by finite state machine inference’, *IEEE Internet of Things J*., 9(22), pp. 22737–22751. doi: 10.1109/JIOT.2022.3182589.
15. Akinina, Yu.S., Tyurin, S.V. (2023) *Automata Theory*. Moscow, 156 p. (in Russ.).

Авторы
Фаткиева Роза Равильевна [1], к.т.н., доцент, rikki2@yandex.ru
Судаков Антон Сергеевич [1], аспирант, asudakov.mail@gmail.com

[1] Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина), г. Санкт-Петербург, 197022, Россия

Authors
Roza R. Fatkieva [1], Cand. of Sci. (Engineering), Associate Professor, rikki2@yandex.ru
Anton S. Sudakov [1], Postgraduate Student, asudakov.mail@gmail.com

[1] St. Petersburg Electrotechnical University "LETI", St. Petersburg, 197022, Russian Federation