Санкт-Петербургский государственный университет

Дроздов Всеволод Святославович

Выпускная квалификационная работа

Web-интерфейс для создания и визуализации сетевых эмуляторов

Уровень образования: бакалавриат

Направление *02.03.03 «Математическое обеспечение и администрирование информационных систем»*

Основная образовательная программа *СВ.5006.2018 «Математическое обеспечение и администрирование информационных систем»*

Профиль *Технология программирования*

Научный руководитель:
Доцент кафедры системного программирования, к.т.н. М.А. Серов

Рецензент:
Программист ООО “Интеллиджей Лабс” Д.С. Косарев

Санкт-Петербург
2022

Saint Petersburg State University

Vsevolod Drozdov

Bachelor's Thesis

Web-interface for creating and visualizing network emulations

Education level: bachelor

Speciality *02.03.03 «Software and Administration of Information Systems»*

Programme *CB.5006.2018 «Software and Administration of Information Systems»*

Profile: *Programming Technology*

Scientific supervisor:
Docent, C.Sc. M.A. Serov

Reviewer:
Software developer at "IntelliJ" Labs Co. Ltd. D.S. Kosarev

Saint Petersburg
2022

Оглавление

1. Введение

По устройству и работе компьютерных сетей проводится множество курсов, большинство из которых обладают недостатком в виде отсутствия должного количества примеров и демонстраций работы компьютерных сетей. Для качественного обучения устройству компьютерных сетей обучающимся требуется возможность проводить практическую работу с сетями, составлять собственные сети и наблюдать механизмы их работы. Реализация примеров работы компьютерных сетей при различных топологиях, протоколах и устройствах в сети без установки соответствующих описываемой сети физических устройств требует использования программного обеспечения, эмулирующего работу компьютерных сетей. Среди решений в данной области стоит выделить такие продукты как GNS3 [17] и Cisco Packet Tracer [7], обладающие возможностью визуализации сетевого трафика и топологии сети, что делает их пригодными для использования в образовательных целях. Однако программные решения в данной сфере являются платными, или же написаны не для образовательных целей, а для людей с высоким уровнем подготовки и хорошим пониманием устройства работы компьютерных сетей. Некоторые из них, такие как NS-3 [22] и nemu [13], требуют написания программ-сценариев для взаимодействия с ними, что требует дополнительного изучения документации по данным продуктам. Другим недостатком использования существующих программных решений является тот факт, что многие продукты в данной сфере являются классическими настольными приложениями, что вовлекает необходимость установки и снижает уровень их доступности.

В связи с перечисленными недостатками существующих на данный момент решений, была разработана концепция web-сервиса, позволяющего конструировать компьютерные сети через web-интерфейс и производить их эмуляцию на серверной стороне web-сервиса, демонстрируя пользователю результаты эмуляции описанной им компьютерной сети. Были сформированы требования, основными из которых являются возможность задания топологии и конфигурации узлов сети, возмож-

ность визуализации маршрутов пакетов, отправленных внутри эмулируемой сети, возможность отображения ARP [27] таблиц маршрутизации для различных узлов, а также возможность отображения содержимого каждого пакета, отправленного внутри сети. На основе разработанной концепции был реализован соответствующий web-сервис, нацеленный на ознакомление обучающихся с устройством работы компьютерных сетей и протоколами взаимодействия в них.

2. Постановка задачи

Цель работы — реализовать web-сервис, позволяющий создавать, эмулировать и визуализировать компьютерную сеть. Для ее достижения были поставлены следующие задачи:

1. Провести обзор существующих решений.
2. Сформировать набор необходимых инструментов и технологий.
3. Реализовать сервис, в который входят следующие возможности:
    - создание модели компьютерной сети через web-интерфейс;
    - запуск эмуляции с помощью JSON объекта (JavaScript Object Notation), описывающего сеть;
    - просмотр результатов эмуляции в web-интерфейсе.
4. Реализовать статические компоненты сервиса.
5. Разместить и опубликовать созданный сервис.
6. Провести апробацию web-сервиса.

3. Требования

Консультантом данной работы были сформированы следующие функциональные требования для реализуемого web-сервиса:

1. Описываемая сеть должна поддерживать следующие типы узлов:
   - пользовательский хост;
   - сетевой концентратор;
   - openflow-свитч [25].
2. Описываемая сеть должна поддерживать следующие протоколы:
   - ICMP [28];
   - TCP [33];
   - UDP [29];
   - ARP [27].
3. В реализованном интерфейсе должны быть следующие компоненты:
   - редактор графа для построения топологии компьютерной сети;
   - анимация маршрутов пакетов между узлами сети;
   - просмотр пакетов в формате PCAP [37], прошедших через определенный интерфейс;
   - просмотр таблиц ARP [27] для хостов в разные моменты времени внутри эмуляции.
4. Должны быть реализованы следующие статические компоненты сервиса:
   - главная страница проекта;
   - страница с руководством по использованию сервиса.

4. Обзор

4.1. Аналогичные продукты

4.1.1. Приложения с поддержкой визуализации и графическим интерфейсом для создания сетей

1. GNS3 [17] — данное приложение обладает встроенным анимато-ром и web-интерфейсом, что делает его одним из наиболее близких по цели проектом, однако данное приложение не обладает возможностью просмотра PCAP [37] файлов средствами интерфейса, а использование механизмов виртуализации для эмуляции сетей осложняет возможное масштабирование проекта для общего использования. Код данного приложения находится в открытом доступе в виде Github-репозиториев [14].

2. Cisco Packet Tracer [7] — классическое настольное приложение, используемое при подготовке к экзаменам от Cisco Network Academy [6]. Данное приложение обладает набором функций, ориентированных на образовательные цели, например, возможностью пошагово отображать прохождение пакетов в сети. Продукт является проприетарным, исходный код закрыт.

3. Boson NetSim [5] — данный продукт так же используется для прохождения сертификации Cisco [6]. Представляет собой сборник лабораторных работ, предназначенных для подготовки пользователя к успешной сдаче сертификации, однако также предоставляет возможность строить собственные топологии сетей в визуальном редакторе. Данное приложение является платным, исходный код закрыт.

4. EVE-NG [10] — данное приложение обладает похожим на GNS3 [17] функционалом. Предоставляет пользователю web-интерфейс, поддерживает коллаборативную работу, работает через взаимодействие с виртуальной машиной, образ которой является платным продуктом. Как и в случае GNS3 [17], для анализа содержи-

мого пакетов в формате PCAP [37] требуется отдельное использование Wireshark [38]. Исходный код закрыт.

4.1.2. Библиотеки для эмуляции сетей

Среди других решений в области эмуляции компьютерных сетей были найдены библиотеки, предоставляющие API (Application Programming Interface) для эмуляции сетей. Среди них стоить выделить:

1. NS-3 [22] — эмулятор компьютерных сетей, предоставляющий библиотеку для программ, написанных на C++ , с помощью которых можно описывать сценарий работы эмулируемой компьютерной сети и генерировать данные о поведении этой сети.

2. nemu [13] — библиотека для языка программирования Python, использующая механизмы проекта Linux Containers [20] для эмуляции компьютерных сетей. Как и NS-3 [22] работает через описание сценария работы эмулируемой компьютерной сети в виде программы на Python. Исходный код доступен в виде проекта на Github [13].

4.1.3. Web-ориентированные решения

1. Antidote [18] — эмулятор компьютерных сетей с web-интерфейсом, поддержка которого была приостановлена, репозитории с кодом заархивированы. Исходный код библиотеки открыт и доступен в виде заархивированного Github-репозитория [19].

2. WNetSim [23] — эмулятор, оставшийся на стадии проекта.

3. CS4G Netsim [1] — web-сервис, предоставляющий набор упражнений и иллюстрирующий различные аспекты работы компьютерных сетей. Данный сервис не обладает возможностью создания и эмулирования сети, однако обладает доступным интерфейсом. Код серверной и клиентских частей открыт и доступен по ссылке на Github-репозиторий [2].

Выбор эмулятора, для которого будет создаваться web-интерфейс, происходил среди решений с открытым исходным кодом.

4.2. Выбор эмулятора

Среди всех существующих программных средств для эмуляции компьютерных сетей была выбрана библиотека NS-3. Было принято решение, что данная библиотека подходит для решения всех описанных задач и удовлетворяет всем приведенным требованиям по следующим причинам:

- Доступность — исходный код проекта открыт и задокументирован с помощью Doxygen [9].

- Сценарий использования — создание и запуск компьютерных эмуляций для NS-3 производится с помощью написания программ на языке C++. Данный сценарий использования является достаточно гибким для адаптации под web-сервис.

- Модель работы — библиотека NS-3 [22] предоставляет модели того, как реализованы и работают компьютерные сети в строго контролируемой, воспроизводимой среде с дискретными событиями, не используя механизмы виртуализации. Такая модель работы позволяет эмулировать несколько сетей одновременно в одном системном процессе и удобна в сценарии работы, когда требуется эмулировать сеть, динамически заданную объектом, создаваемым пользователем, а также позволяет контролировать набор требуемого функционала.

- Спектр возможностей — NS-3 [22] предоставляет как модули верхнего уровня для облегчения типичных задач, так и доступ к низкоуровневым методам. Данная библиотека позволяет явным образом создавать устройства эмулируемой сети и конфигурировать заданные в них приложения, имея при этом набор вспомогательных классов, дающих возможности настройки соединения между

узлами и генерации данных о работе сети и состояниях узлов, что позволяет решить все описанные задачи web-сервиса.

4.3. Программные средства

Ниже приведено описание программных средств, которые использовались в процессе разработки:

1. NS-3 [22] был выбран в качестве целевого эмулятора сети.

2. В качестве программной платформы для серверной части сервиса был использован Node.js [24] версии v12.22.0 для платформы x86_64, работающий с фреймворком Express.js [12] версии 4.17.1. Express.js [12] — это фреймворк для создания web-приложений, предоставляющих REST (Representational State Transfer) [11] API для доступа из клиентской части приложения. Данный фреймворк решает задачу передачи данных о работе эмуляции клиентской стороне web-сервиса.

3. Node-gyp [40] версии v9.0.0 был выбран как средство создания Node.js-обертки для NS-3. Решение создать Node.js-обертку для библиотеки NS-3 было принято в связи с выбором Express.js [12] в качестве платформы для разработки серверной части приложения.

4. Для разработки клиентской части приложения была использована библиотека React [30] версии 18.0.0. Причиной выбора данной библиотеки послужило наличие компонентов — интегрируемых частей создаваемого пользовательского интерфейса, находящихся в свободном доступе в виде Javascript-модулей и решающих некоторые задачи, стоящие перед web-сервисом. Одним из таких модулей является React-digraph [41], предоставляющий визуальный редактор графов в пользовательском web-интерфейсе.

5. В качестве программной платформы для клиентской части сервиса был выбран фреймворк Next.js [34] версии 11.1.2. Данный

фреймворк упрощает создание web-сервисов с использованием библиотеки React [30]. Также для данного фреймворка существует хостинг-сервис Vercel [35], предоставляющий функционал CI (Continuous Integration).

6. Docker [8] версии 20.10.14 был использован для развертывания серверной части приложения. Данный продукт позволяет облегчить процесс сборки библиотеки NS-3 [22] и ее зависимостей.

7. Для серверной и клиентской частей был использовал пакетный менеджер yarn [39] версии 1.22.11.

8. Для менеджмента процессов использовался pm2 [26] версии 4.5.5.

9. Git [16] — система управления версиями.

Для разработки использовались языки программирования Javascript и C++.

5. Описание решения

5.1. Разработка эмулятора

Сборка библиотеки NS-3 произведена через систему сборки Waf [36]. Также был установлен дополнительный модуль для openflow-устройств [21], который требует наличие C++ библиотеки Boost[4] версии ≤ 1.68, которая была собрана с помощью системы сборки Make. Полный алгоритм сборки был записан в виде Dockerfile для используемого образа Docker [8].

Эмуляция компьютерной сети, задаваемой пользователем, происходит в серверной части приложения. Через REST [11] API серверная часть принимает JSON (JavaScript Object Notation) объект, описывающий эмулируемую сеть, и обрабатывает с помощью написанного Node-gyp [40] модуля. Данный объект содержит в себе граф, описывающий топологию построенной сети. Каждая вершина графа имеет определенный тип, поддерживаются следующие типы вершин:

- пользовательский хост;
- сетевой концентратор (хаб);
- Openflow-свитч [25].

Также вершины графа содержат информацию о сервисах, которые должны работать в соответствующем узле сети. Информация о сервисе включает в себя его тип, а также информацию, необходимую для конкретного типа сервиса. Поддерживаются следующие типы сервисов:

- TCP [33] клиент;
- TCP [33] сервер;
- UDP [29] клиент;
- UDP [29] сервер;
- Ping — ICMP [28] клиент;

- Packet sink — сервис-заглушка для принятия сетевых пакетов.

Ребра графа описывают соединения между узлами. В каждом ребре хранится фиксированный IP [32] адрес для каждого узла, соединяемого этим ребром, имеющего тип пользовательского хоста. Данный IP [32] адрес используется для назначения его сетевому интерфейсу, соответствующему этому соединению. Пример JSON-объекта, обрабатываемого в серверной части web-сервиса приведен в Листинге 1 секции Приложение.

Входной JSON-объект обрабатывается в Node-gyp [40] модуле методами `getNodes`, `getGraph`, `getOptions`, `getAddrInfo`, использующими классы библиотеки Node-gyp [40] для обработки Node.js [24] объектов, и переводится в объекты пользовательских классов C++. После чего происходит инстанцирование соответствующих конфигурации сети объектов библиотеки NS-3 [22] для каждого узла сети с последующей итерацией по вершинам заданного графа и инициализацией соединений между узлами сети с помощью инстанцирования объектов вспомогательных классов библиотеки NS-3 и вызовов их методов. Далее происходит инстанцирование объектов для установки сервисов, описанных во входных данных, на каждый узел сети.

Вызовы методов библиотеки NS-3 [22], позволяющих получить клиенту всю необходимую информацию происходят в функции `initTracing`. Одна из функций модуля представлена в листинге 2 секции Приложение.

Результатом работы такой программы является файл в формате XML (eXtensible Markup Language), описывающий каким образом необходимо анимировать маршруты пакетов в сети.

Одной из целей создаваемого эмулятора была генерация файлов в формате PCAP [37] для детального изучения работы созданной компьютерной сети. Данный функционал предоставляется библиотекой NS-3, однако, для будущего отображения этих данных, был написан модуль, переводящий их в формат JSON (JavaScript Object Notation), использующий для этого утилиту tshark [31] версии 3.6.3.

Для предоставления пользователю информации об ARP [27] табли-

цах пользовательских хостов в сети был использован дополнительный модуль `Ipv4RoutingHelper`. Фрагмент данного модуля представлен в Листинге 3 секции Приложение.

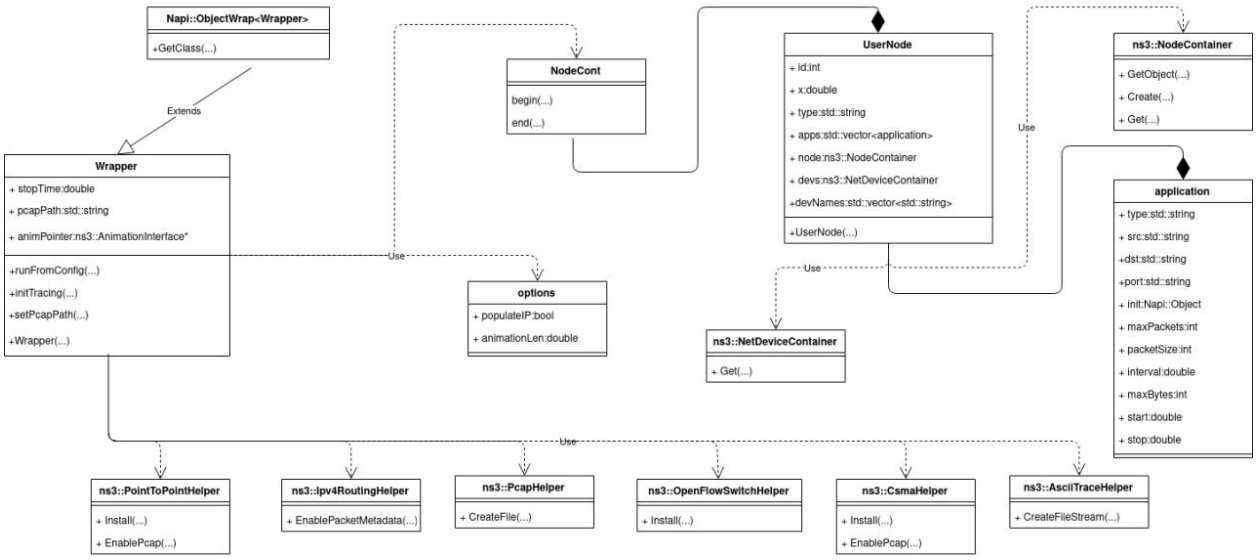

Рис. 1: Диаграмма некоторых классов модуля

5.2. Серверная часть web-сервиса

REST [11] API для приема и обработки JSON-объекта с последующей эмуляцией описываемой им сети предоставляется с помощью фреймворка Express.js [12], для которого созданы точки доступа REST [11], позволяющие передать в API JSON-объект с описанием сети и получить необходимую информацию о проведенной эмуляции сети. Код одной из точек доступа REST [11] приведен в Листинге 4 секции Приложение.

5.3. Разработка web-интерфейса для взаимодействия с эмулятором

Пользовательский интерфейс реализован на React [30].

Логически взаимодействие с сервисом проходит в три этапа:

1. Конструирование графа, описывающего сеть с помощью библиотеки React-digraph [41], в конце данного этапа с клиентской части

отправляется запрос на сервер с JSON-объектом, описывающим сеть.

2. После получения ответа происходит запуск анимации, демонстрирующей маршруты пакетов в сети.

3. Последующий анализ PCAP [37] файлов с помощью соответствующего созданного компонента, внешне подобного приложению Wireshark [38], а также отображение ARP [27] таблиц пользовательских хостов. Для отображения каждой таблицы происходит запрос на серверную часть с прикрепленными к нему идентификатором произведенной эмуляции и информации о выбранном узле.

Данная логика реализуется с помощью созданного React-контекста `GraphContext` и реализованных пользовательских хуков `useAnimation`, `useIPPool`, а также нативных хуков библиотеки React [30]. Фрагменты кода, содержащие логику клиентской части представлены в Листингах 5 и 6 секции Приложение. Для запросов к серверной части была использована библиотека Axios [3].

Интерфейс создания компьютерной сети состоит из следующих компонентов:

- редактор графов, использующий компонент React-digraph [41];

- контекстное меню, открывающееся при нажатии на вершину графа рядом с выбранной вершиной, в котором на первом этапе взаимодействия с интерфейсом отображаются установленные на соответствующий узел сервисы, присваиваемые IP [32] адреса добавленным интерфейсам, на третьем этапе отображаются доступные для каждого подключения PCAP [37] файлы, а также имя файла, содержащего информацию о таблице маршрутизации соответствующего узла сети. При нажатии на имя файла открывается окно с таблицей анализа PCAP [37] файла или окно с ARP [27] таблицей маршрутизации в разные моменты времени в зависимости от выбранного файла;

- кнопка, производящая запуск, остановку или возврат к первому этапу работу в зависимости от текущего этапа;
- всплывающее окно и кнопка, открывающая его, для настройки длительности эмуляции и скорости воспроизведения анимации;
- всплывающее окно и кнопка, открывающая его, для добавления новых вершин в текущий граф;
- окно с таблицей, отображающей информацию о пакетах, отправленных или полученных через определенное соединение узла;
- окно с ARP [27] таблицей маршрутизации узла.

Для стилизации компонентов была выбрана библиотека geist-ui [15]. На Рис. 2 представлен фрагмент интерфейса конструирования сетей, находящийся на 2 этапе работы:

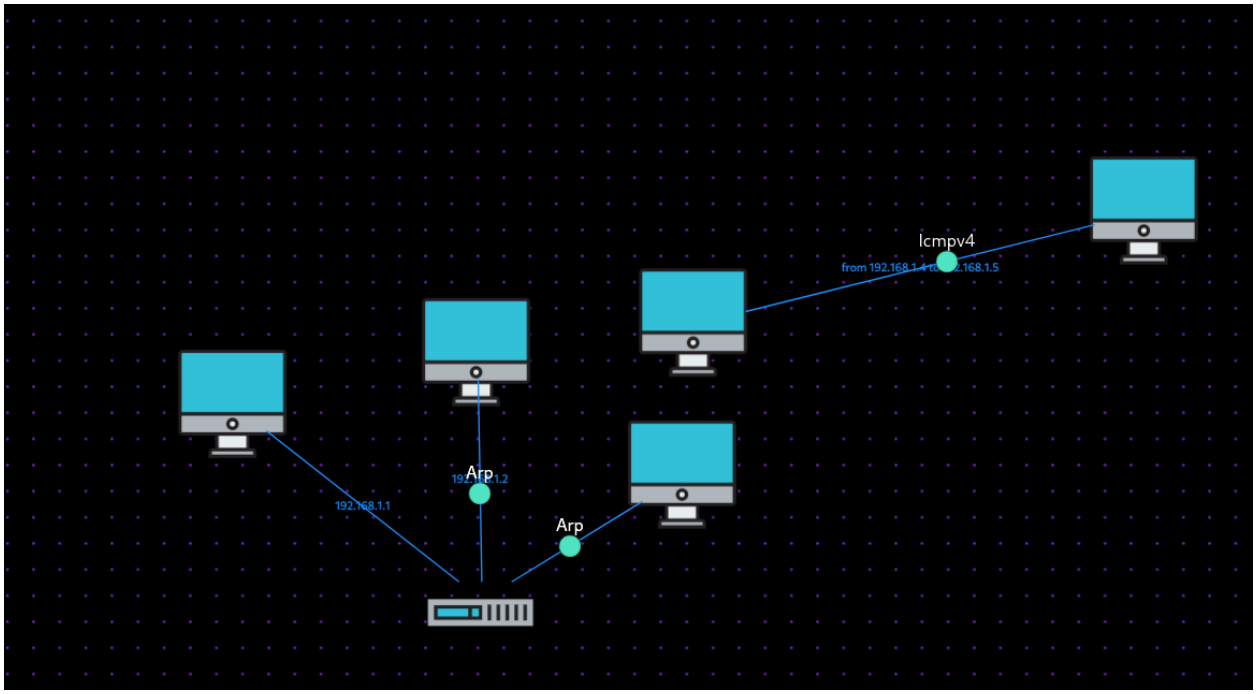

Рис. 2: Анимация маршрутов пакетов

5.4. Другие компоненты web-сервиса

Web-сервис состоит из трех страниц: главной страницы, страницы с web-интерфейсом для создания эмулируемых компьютерных сетей и

страницы с руководством по использованию. На главной странице и странице с руководством содержится инструкция по взаимодействию с интерфейсом. Страницы web-сервиса расположены по следующим маршрутам:

- / — маршрут главной страницы;
- /playground — маршрут страницы с интерфейсом для создания эмуляций;
- /help — маршрут страницы с руководством по использованию.

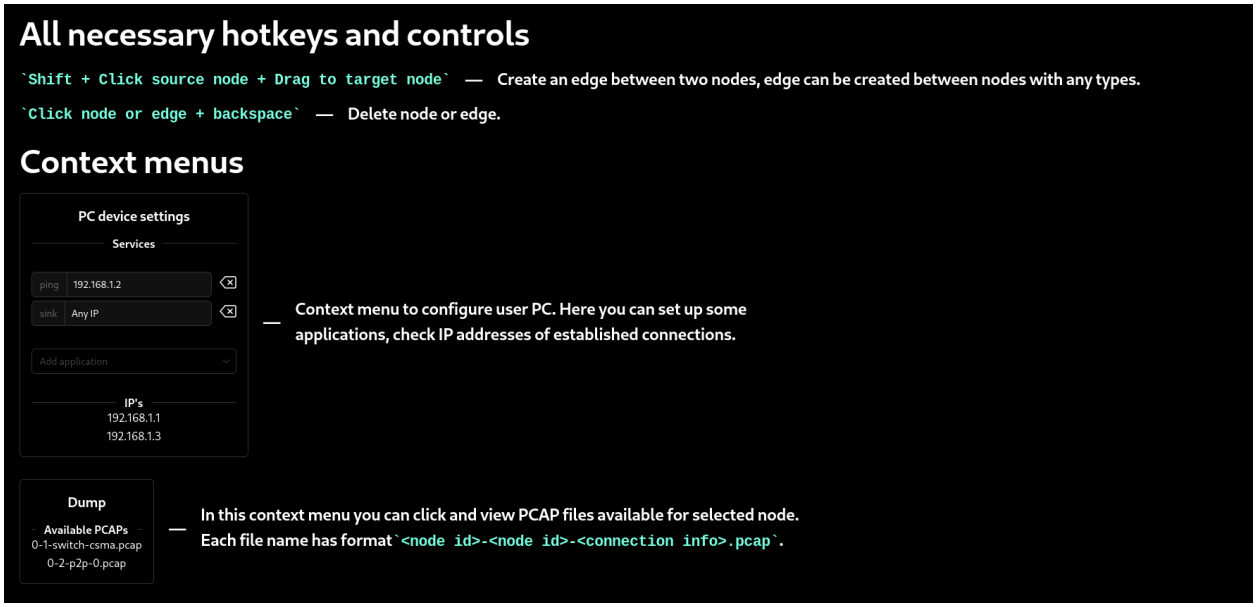

Рис. 3: Фрагмент страницы с руководством по использованию сервиса

5.5. Размещение web-сервиса

Next.js [34]-часть проекта была размещена на сервисе Vercel [35]. Ввиду большого количества зависимостей для библиотеки NS-3 [22], для размещения серверной части приложения был создан Docker [8]-образ, собранный и запущенный на облачной виртуальной машине.

6. Апробация

Для апробации web-сервиса были использованы методы System Usability Scale и Single Ease Question. В опросе приняли участие 10 человек. Каждому участнику предлагалось пройти опрос, состоящий из двух этапов:

- Single Ease Question — выполнение действий в интерфейсе для создания и эмуляции компьютерных сетей с последующей оценкой сложности каждого действия по шкале от 1 (Очень сложно) до 7 (Очень легко).
- System Usability Scale — ответ на вопросы по шкале от 1 до 5.

Результаты опроса по методу Single Ease Question представлены на Рис. 4.

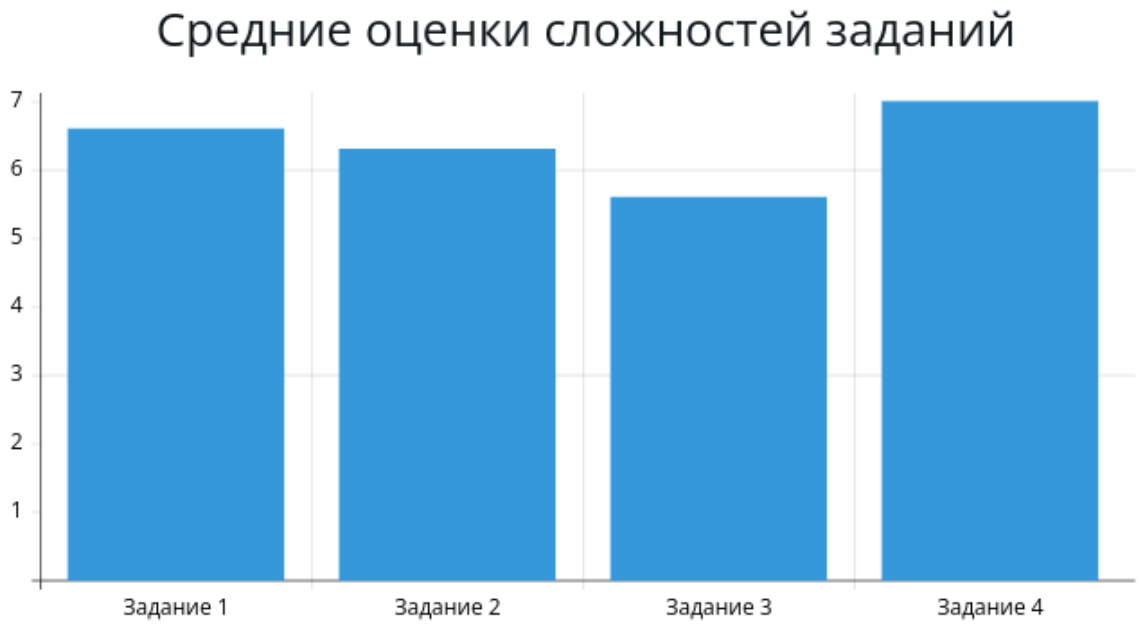

Рис. 4: Single Ease Question

Среднее количество баллов по шкале System Usability Scale — 76.8, что соответствует оценке “B” — хорошо.

7. Заключение

В ходе проведения данной работы было сделано следующее:

1. Был проведен обзор существующих решений.
2. Был сформирован весь программный стек, необходимый для запуска сервиса.
3. Был реализован сервис, в который входят следующие возможности:
   - создание модели компьютерной сети через web-интерфейс;
   - запуск эмуляции с помощью JSON объекта, описывающего сеть;
   - просмотр результатов эмуляции в web-интерфейсе.
4. Описываемая сеть поддерживает следующие типы узлов:
   - пользовательский хост;
   - сетевой концентратор;
   - openflow-свитч [25].
5. Описываемая сеть поддерживает следующие протоколы:
   - ICMP [28];
   - TCP [33];
   - UDP [29];
   - ARP [27].
6. Был реализован интерфейс, в котором есть следующие компоненты:
   - редактор графа для построения компьютерной сети;
   - анимация маршрутов пакетов между узлами сети;

- просмотр пакетов в формате PCAP [37], прошедших через определенный интерфейс;
- просмотр таблиц ARP [27] для хостов в разные моменты времени внутри эмуляции.

7. Сервис был развернут и опубликован.
   - Интерфейс был размещен на Next.js [34] хостинге Vercel [35].
   - Был создан Docker-контейнер для сервиса запуска эмуляций, серверная часть была запущена в нем на облачной виртуальной машине.
8. Реализованы статические компоненты сервиса, среди которых есть:
   - главная страница проекта;
   - страница с руководством по использованию сервиса.
9. Проведена апробация web-сервиса.

Исходный код проекта доступен по адресу:
https://gitlab.com/vsevolod2000/net-runner

Список литературы

[1] Atwater Erinn. Netsim is a simulator game intended to teach you the basics of how computer networks function, with an emphasis on security. — Access mode: https://netsim.erinn.io/ (online; accessed: May 10, 2022).

[2] Atwater Erinn. Web-based network simulator for teaching hacking to high schoolers. — Access mode: https://github.com/errorinn/netsim (online; accessed: May 10, 2022).

[3] Axios. Promise based HTTP client for the browser and node.js. — Access mode: https://github.com/axios/axios (online; accessed: May 10, 2022).

[4] Boost. Boost provides free peer-reviewed portable C++ source libraries. — Access mode: https://www.boost.org/ (online; accessed: May 10, 2022).

[5] Boson. Most Advanced Network Simulator Designed for Cisco Certification Training. — Access mode: https://www.boson.com/netsim-cisco-network-simulator (online; accessed: May 10, 2022).

[6] Cisco. Cisco Network Academy. — Access mode: http://web.archive.org/web/20220408223855/https://www.netacad.com (online; accessed: May 10, 2022).

[7] Cisco. Cisco packet tracer. — Access mode: http://web.archive.org/web/20220408223855/https://www.netacad.com/ru/courses/packet-tracer (online; accessed: May 10, 2022).

[8] Docker. Docker is a set of platform as a service (PaaS) products that use OS-level virtualization to deliver software in packages called containers. — Access mode: https://www.docker.com/ (online; accessed: May 10, 2022).

[9] Doxygen. Doxygen is the de facto standard tool for generating documentation from annotated C++ sources. — Access mode: http://www.doxygen.nl (online; accessed: May 10, 2022).

[10] EVE-NG. The Emulated Virtual Environment For Network, Security and DevOps Professionals. — Access mode: https://www.eve-ng.net/ (online; accessed: May 10, 2022).

[11] Erik Wilde Cesare Pautasso. REST: From Research to Practice. — Springer Science Business Media.

[12] Express. Fast, unopinionated, minimalist web framework for Node.js. — Access mode: https://expressjs.com/ (online; accessed: May 10, 2022).

[13] Ferrari Martina. A lightweight network emulator embedded in a small python library. — Access mode: https://github.com/NightTsarina/nemu (online; accessed: May 10, 2022).

[14] GNS3. GNS3 repositories. — Access mode: https://github.com/GNS3 (online; accessed: May 10, 2022).

[15] Geist. An open source design system for building modern websites and applications. — Access mode: https://geist-ui.dev/en-us (online; accessed: May 10, 2022).

[16] Git. Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency. — Access mode: https://git-scm.com/ (online; accessed: May 10, 2022).

[17] LLC Galaxy Technologies. The official guide and reference for GNS3. — Access mode: https://docs.gns3.com/ (online; accessed: May 10, 2022).

[18] Labs NRE. Antidote is a network emulator combined with a presentation framework designed to create and deliver networking technology

training. Its user interface operates in a web browser, including the terminals that students use to run commands on emulated network devices and servers. — Access mode: https://docs.nrelabs.io/ (online; accessed: May 10, 2022).

[19] Labs NRE. Source code of Antidote project (No longer maintained). — Access mode: https://github.com/Antidote-for-Tox/Antidote (online; accessed: May 10, 2022).

[20] Linux. Linux Containers. — Access mode: https://linuxcontainers.org/ (online; accessed: May 10, 2022).

[21] NS-3. OpenFlow switch support. — Access mode: https://www.nsnam.org/docs/release/3.13/models/html/openflow-switch.html (online; accessed: May 10, 2022).

[22] NS-3. ns-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. ns-3 is free, open-source software, licensed under the GNU GPLv2 license, and maintained by a worldwide community. — Access mode: https://www.nsnam.org/ (online; accessed: May 10, 2022).

[23] Nenad M Jovanovic R. Popovic Zoran Jovanovic. WNetSim: A Web-Based Computer Network Simulator. — Access mode: https://www.researchgate.net/publication/233691072_WNetSim_A_Web-Based_Computer_Network_Simulator (online; accessed: May 10, 2022).

[24] Node.js. Node.js is a JavaScript runtime built on Chrome's V8 JavaScript engine. — Access mode: https://nodejs.org/en/ (online; accessed: May 10, 2022).

[25] ONF. Collaboratively transforming network infrastructure. — Access mode: https://opennetworking.org/ (online; accessed: May 10, 2022).

[26] PM2. PM2 is a daemon process manager that will help you manage and keep your application online 24/7. — Access mode: https://pm2.keymetrics.io/ (online; accessed: May 10, 2022).

[27] Plummer David C. An Ethernet Address Resolution Protocol. — Access mode: https://datatracker.ietf.org/doc/html/rfc826 (online; accessed: May 10, 2022).

[28] Postel J. INTERNET CONTROL MESSAGE PROTOCOL. — Access mode: https://datatracker.ietf.org/doc/html/rfc792 (online; accessed: May 10, 2022).

[29] Postel J. User Datagram Protocol. — Access mode: https://datatracker.ietf.org/doc/html/rfc768 (online; accessed: May 10, 2022).

[30] React. A JavaScript library for building user interfaces. — Access mode: https://en.reactjs.org/ (online; accessed: May 10, 2022).

[31] Tshark. Dump and analyze network traffic. — Access mode: https://www.wireshark.org/docs/man-pages/tshark.html (online; accessed: May 10, 2022).

[32] University Southern California. Internet Protocol. — Access mode: https://datatracker.ietf.org/doc/html/rfc791 (online; accessed: May 10, 2022).

[33] University Southern California. TRANSMISSION CONTROL PROTOCOL. — Access mode: https://datatracker.ietf.org/doc/html/rfc793 (online; accessed: May 10, 2022).

[34] Vercel. The React Framework for Production. — Access mode: https://nextjs.org/ (online; accessed: May 10, 2022).

[35] Vercel. Vercel combines the best developer experience with an obsessive focus on end-user performance. Our platform enables frontend teams to do their best work. — Access mode: https://vercel.com/ (online; accessed: May 10, 2022).

[36] Waf. Waf is a piece of software used to help building software projects. The goal of this tutorial is to provide a quick overview of how to set up the scripts for a project using Waf. — Access mode: https://waf.io/apidocs/tutorial.html (online; accessed: May 10, 2022).

[37] Wikipedia. pcap is an application programming interface (API) for capturing network traffic. — Access mode: https://ru.wikipedia.org/wiki/Pcap (online; accessed: May 10, 2022).

[38] Wireshark. Wireshark is the world's foremost and widely-used network protocol analyzer. — Access mode: https://www.wireshark.org/ (online; accessed: May 10, 2022).

[39] Yarn. Package Manager. — Access mode: https://yarnpkg.com/ (online; accessed: May 10, 2022).

[40] node.js. Node.js native addon build tool. — Access mode: https://github.com/nodejs/node-gyp (online; accessed: May 10, 2022).

[41] uber. A library for creating directed graph editors. — Access mode: https://github.com/uber/react-digraph (online; accessed: May 10, 2022).

8. Приложение

Листинг 1: Пример javascript-объекта конфигурации сети

```
{
  nodes: [
    {
      id: 0,
      x: 100,
      y: 100,
      type: 'pc',
      applications: [
        {
          type: 'ping',
          dst: '192.168.1.2',
        },
      ],
    },
    {
      id: 1,
      x: 150,
      y: 150,
      type: 'hub',
    },
    {
      id: 2,
      x: 200,
      y: 200,
      type: 'pc',
      applications: [
        {
          type: 'sink',
          dst: '192.168.1.2',
        }
```

```
    ],
  },
  {
    id: 3,
    x: 300,
    y: 300,
    type: 'pc',
    appplicatoins: [],
  }
],
edges: [
  {
      source: 0,
      target: 1,
      type: 'default',
      sourceIP: '192.168.1.1',
      targetIP: ''
  },
  {
      source: 1,
      target: 2,
      type: 'default',
      sourceIP: '',
      targetIP: '192.168.1.2'
  },
  {
      source: 1,
      target: 3,
      type: 'default',
      sourceIP: '',
      targetIP: '192.168.1.3'
  },
],
```

```
  options: {
    animeLen: 10,
    populateIP: false,
  },
}
```

Листинг 2: Верхняя в логической иерархии функция

```
Napi::Value Wrapper::fromConfig(const Napi::CallbackInfo& info) {
  Napi::Env env = info.Env();
  if (info.Length() < 1) {
    Napi::TypeError::New(env, ''Specify config'')
        .ThrowAsJavaScriptException();
  }
  auto config = info[0].As<Napi::Object>();

  auto myNodes = getNodes(config);
  auto graph = getGraph(config);
  auto options = getOpitons(config);
  auto addrInfo = getAddrInfo(config);

  debug << ''[*] graph:'' << endl;
  for (int i = 0; i < graph.size(); ++i) {
    for (auto v : graph[i]) {
      debug << i << ' ' << v << endl;
    }
  }

  setupNodes(myNodes);
  debug << ''[*] setted up nodes'' << endl;
  setupConnections(myNodes, graph, addrInfo);
  debug << ''[*] setted up connections'' << endl;
  setupApplications(myNodes, options);
  debug << ''[*] setted up applications'' << endl;
```

```
  debug << ''[*] setted up'' << endl;

  if (options.populateIP) {
    Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
  }
  initTracing(env, myNodes);
  this → stopTime = options.animeLen;

  return env.Null();
}
```

Листинг 3: Пример фрагмента кода из функции initTracing позволяющего считывать значения ARP таблиц каждую 0.1 секунду и записывать в текстовый файл результаты

```
  AsciiTraceHelper ascii;
  for (auto& keyVal : myNodes) {
    auto& e = keyVal.second;
    if (e.type == ''pc'') {
      auto path = fns::path{this → tracePath} /
        fns::path{to_string(e.id) + ''-arp.txt''};
      Ipv4RoutingHelper::PrintNeighborCacheEvery(
        Seconds(0.1),
        e.node.Get(0),
        ascii.CreateFileStream(path)
      );
    }
  }
```

Листинг 4: Точка доступа REST

```
router.post('/launch', async function(req, res, next) {
  try {
    return res.json({
```

```
      message: 'ok',
      data: await configController.launch(req.body),
    })
  } catch(e) {
    logger.debug(e);
    return next(e);
  }
});
```

Листинг 5: Инициализация некоторых логических объектов интерфейса

```
  const [stage, setStage] = useState(0);
  const [resp, setResp] = useState(null);

  const [state, setState] = useState({ selected: null });

  const [options, setOptions] = useState({
    speed: 0.00015,
    animeLen: 4,
    populateIP: false,
  });

  const [graph, setGraph] = useState(sampleGraph);


  const animationNodes = useAnimation({ // user hook
    active: stage == 2 && resp,
    onFinish: setStage.bind(null, 3),
    anime: resp?.data?.anime,
    options, graph,
  });
```

Листинг 6: Инициализация объекта для хранения назначенных IP адресов

```
const defaultBase = '192.168.1.0';
const [IPPool, setIPPool] = useState(sampleGraph.edges.reduce
((res, e) =>
  ({ ...res, [e.sourceIP]: true, [e.targetIP]: true }), {})
);
```