МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В. ЛОМОНОСОВА

Факультет
вычислительной математики
и кибернетики

ПРОГРАММНЫЕ СИСТЕМЫ И ИНСТРУМЕНТЫ

Тематический сборник

№ 23

МАКС Пресс

МОСКВА — 2023

МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА

Факультет вычислительной математики и кибернетики

ПРОГРАММНЫЕ СИСТЕМЫ И ИНСТРУМЕНТЫ

Тематический сборник

№ 23

Под общей редакцией
чл.- корр. РАН, профессора Р. Л. Смелянского

МАКС ПРЕСС

Москва – 2023

УДК 519.6+517.958
ББК 22.19
П75

https://elibrary.ru/ukkuyr

Печатается по решению Редакционно-издательского совета факультета вычислительной математики и кибернетики МГУ имени М.В. Ломоносова

Программные системы и инструменты : Тематический сборник / Под ред. Р.Л. Смелянского. – Москва : Издательский отдел факультета ВМК МГУ имени М.В. Ломоносова (лицензия ИД № 05899 от 24.09. 2001 г.); МАКС Пресс, 2023. № 23. – 140 с.

П75

ISBN 978-5-89407-638-6 (ВМК МГУ имени М.В.Ломоносова)
ISBN 978-5-317-07118-9 (МАКС Пресс)
https://doi.org/10.29003/m3791.978-5-317-07118-9

Данный выпуск сборника составлен по материалам работ студентов и аспирантов, получивших рекомендации научного семинара кафедры Автоматизации систем вычислительных комплексов. Редколлегия сборника продолжает традицию его издания в память о первом руководителе кафедры АСВК Королёве Л.Н. В предлагаемом читателю тематическом сборнике публикуются статьи, посвященные проблемам современных компьютерных сетей, методам машинного обучения, задачам и алгоритмам дискретной оптимизации, анализу сетевого трафика.

Статьи сборника будут интересны студентам, аспирантам и специалистам в области разработки прикладных программных систем.

Ключевые слова: сети передачи данных, программно-конфигурируемые сети, Интернет вещей, машинное обучение, метод опорных векторов, DoS-атаки, ПКС-контроллер, виртуальная машина, блокчейн, анализ сетевого трафика, качество обслуживания, гетерогенные каналы, графовые базы данных, маршрутизация трафика, многоагентное обучение, балансировка нагрузки, сбор медицинской информации, многоагентные системы, генетические алгоритмы.

УДК 519.6+517.958
ББК 22.19

ISBN 978-5-89407-638-6
978-5-317-07118-9

Оглавление

СБОРНИК
«Программные системы и инструменты»

От редколлегии:

Тематический сборник "Программные системы и инструмент" был инициирован Львом Николаевичем Королевым в 2000 году, как площадка, где студенты и молодые ученые могли бы публиковать свои достижения. Все эти годы Лев Николаевич неустанно вел его, отбирал публикации, редактировал. Редколлегия сборника продолжает эту традицию в память об этом выдающимся человеке.

Этот выпуск сборника составлен по материалам работ студентов и аспирантов, получивших рекомендации научных семинаров кафедры Автоматизации Систем Вычислительных Комплексов, которую Л.Н. Королев создал и которой бессменно руководил.

Также в этом сборнике представлена одна из работ Льва Николаевича Королёва "Генетические алгоритмы с формальной точки зрения".

Редколлегия

Балашов В.В.

Организация обмена данными в реальном времени на базе программно-конфигурируемых сетей

Введение

Управляющая система реального времени (УС РВ), такая как бортовая информационно-управляющая система самолета или система управления автоматической производственной линией, может быть условно разделена на "ядро", в котором сосредоточены основные вычислительные ресурсы, и "периферию", представляющую собой совокупность контроллеров в составе датчиков и эффекторов. При этом сетевая среда в составе "ядра" УС РВ значительно отличается от сетевой среды, соединяющей "ядро" и периферийные устройства [1]. В составе "ядра" используются высокоскоростные каналы передачи данных, например, Fibre Channel [2] с пропускной способностью 1 Гбит/с и более, в то время как периферийные устройства подключены по линиям унаследованных стандартов, таких как ARINC 429 [3] и CAN [4]. Разнообразие используемых в УС РВ типов каналов усложняет процессы поддержки и модернизации УС РВ. Для решения этой проблемы целесообразно унифицировать сетевую среду в составе УС РВ (внутреннюю среду), сократив разнообразие типов каналов, но по-прежнему обеспечивая обмен данными в реальном времени. Кроме того, для УС РВ, взаимодействующих с периферийными ЦОД (для переноса части вычислений в облако) и другими УС РВ целесообразна унификация внутренней и внешней сетевой среды УС РВ, чтобы не добавлять еще один тип канала для внешней связи.

В качестве технологии для построения унифицированной сетевой среды в УС РВ целесообразно рассмотреть программно-конфигурируемые сети (ПКС) [5, 6] в связи с их широкими возможностями по управлению обменом данными как на уровне коммутаторов, так и централизованно — на уровне контроллера сети. В данной работе предложен подход к организации обмена данными в реальном времени на основе ПКС. Этот подход ориентирован на объединение в одной сети различных потоков данных с удовлетворением типичных для УС РВ требований качества сервиса.

1. Каналы передачи данных в составе сетевой среды УС РВ

Существует ряд признаков, по которым можно классифицировать каналы передачи данных. Среди них: физическая среда передачи данных, пропускная способность, область применения и т.п. Для классификации каналов в данной работе выберем два признака:

1. топология (структура) канала;

2. схема управления передачей данных.

С одной стороны, эти два признака достаточно общи, чтобы прослеживаться у всех каналов, используемых в УС РВ. С другой стороны, в совокупности они существенно влияют на схему передачи данных по каналу и связанные с этой схемой ограничения.

Будем рассматривать только цифровые каналы передачи данных. Проблема преобразования данных между аналоговым и цифровым форматами находится вне рамок данной работы.

Будем разделять каналы по выбранным признакам следующим образом.

По топологии: точка-точка; общая шина; кольцо; звезда (на основе концентратора или коммутатора); фабрика (совокупность "звёзд", основанных на коммутаторах).

По схеме управления передачей данных:

Единственный отправитель: только один абонент может отправлять данные; конфликты невозможны.

Централизованное управление: один из абонентов является контроллером канала, и все обмены данными инициируются только им; это позволяет избежать конфликтов при доступе к каналу.

Арбитраж: абоненты, желающие передать данные в канал, по определенному алгоритму определяют, кто из них первым получит доступ.

Коммутация пакетов: все абоненты подключены к коммутатору или совокупности коммутаторов; коммутаторы обеспечивают отсутствие конфликтов при передаче данных, но возможна перегрузка линии к одному абоненту при приеме данных от нескольких абонентов, что приводит к искажению временны*ы*х характеристик передачи данных.

Коммутация виртуальных каналов: все абоненты подключены к коммутатору или совокупности коммутаторов; для коммутаторов

	Топология				
Управл. передачей данных	Точка-точка	Общая шина	Кольцо	Звезда	Фабрика
Единственный отправитель	ARINC 429 ARINC 818				
Централизованное управление		MIL-STD-1553 CAN	FC-AE-1553	FC-AE-1553	FC-AE-1553
Арбитраж		CAN	FC-AL		
Коммутация пакетов				FC-AE-ASM	FC-AE-ASM
Виртуальные каналы				AFDX FC-AE-ASM-RT	AFDX FC-AE-ASM-RT

Таблица 1: Основные стандарты каналов УС РВ

задана система виртуальных каналов, на каждом из которых есть один абонент-отправитель и один или несколько абонентов-получателей; отсутствие конфликтов при передаче данных обеспечивается фиксированным разделением пропускной способности физической среды между виртуальными каналами.

В таблице 1 приведена классификация основных стандартов каналов УС РВ по признакам "топология" и "управление передачей данных". Детальную информацию об этих каналах можно получить в источниках [2, 3, 4, 7, 8, 9, 10, 11, 12, 13].

При выборе топологии для унифицированной сетевой среды УС РВ следует иметь в виду, что топология "точка-точка" плохо приспособлена для реализации сложных информационных связей с большим числом пар взаимодействующих устройств; топологии "общая шина" и "кольцо" обладают низкой устойчивостью к отказам отдельных абонентов, а также повреждениям физических каналов. Тем самым, наилучшим выбором являются топологии на основе коммутатора: "звезда" и более общая топология "фабрика".

Наилучшим вариантом схемы управления передачей данных для унифицированной сетевой среды является схема с виртуальными каналами, поскольку она изначально предназначена для мультиплексирования множества (десятков, сотен) потоков данных в единой физической среде.

Стандарты AFDX [13] и FC-AE-ASM-RT [12] описывают сетевую среду с топологией "фабрика", поддерживающую виртуальные каналы. При этом оба стандарта обладают ограниченными возможностями по реконфигурации сети (например, для парирования отказов). В коммутатор AFDX может быть загружена только одна таблица виртуальных каналов, что не позволяет реконфигурировать сеть (изменить источник, получателя, маршрут виртуального канала) за исключением

случая, когда в таблицу заложены виртуальные каналы для всех предусмотренных отказов — а это приводит к избыточному резервированию пропускной способности сети. Коммутатор FC-AE-ASM-RT может переключаться между несколькими заранее загруженными таблицами виртуальных каналов, что позволяет парировать одиночные отказы, но не достаточно для работы с кратными отказами, при которых сочетания отказавших устройств многочисленны.

Таким образом, унифицированная сетевая среда УС РВ должна быть построена по топологии "фабрика", поддерживать передачу данных в реальном времени через виртуальные каналы, а также обеспечивать реконфигурацию сети в ходе работы системы при множественных отказах.

2. Требования качества сервиса в сетевой среде УС РВ

2.1 Структура потоков данных в УС РВ

Отправителями и получателями данных в УС РВ являются задачи (программы), выполняющиеся на блоках УС РВ. В зависимости от расположения задачи-отправителя и задачи-получателя, данные передаются:

- через оперативную память блока УС РВ — если задача-отправитель и задача-получатель выполняются на одном блоке;
- через каналы передачи данных — если задача-отправитель и задача-получатель выполняются на разных блоках.

В системах с интегрированной модульной архитектурой (ИМА) [1] в состав блоков входят модули, которые могут не иметь общей памяти, в таком случае обмен данными между задачами с различных модулей происходит также через каналы информационного обмена.

При проектировании современных УС РВ совокупность потоков данных между задачами первоначально задается без привязки к физическим каналам передачи данных. Это позволяет в дальнейшем рассматривать различные варианты привязки задач к блокам УС РВ и вычислительным модулям в составе блоков, а также выполнять балансировку нагрузки на различные каналы передачи данных [14, 15].

Поскольку большинство вычислительных задач в УС РВ являются периодическими и формируют выходные данные в конце очередной итерации выполнения [16], потоки данных между задачами могут быть представлены в виде совокупности периодически передаваемых сообщений, каждое из которых характеризуется следующими атрибутами: размер; задача-отправитель; частота формирования (как правило, совпадает с частотой выполнения задачи-отправителя); одна или несколько задач-получателей. Будем считать, что одному потоку данных соответствует одно периодическое сообщение. Требования качества сервиса к передаче данных через сеть будем рассматривать на уровне требований к передаче сообщений.

При заданной привязке задач к блокам УС РВ и модулям в их составе, заданы также блок (модуль) - отправитель сообщения и блоки (модули) - получатели сообщения; на финальных этапах проектирования УС РВ также фиксируется распределение потоков данных по физическим каналам передачи данных (возможно, различное для разных режимов функционирования УС РВ). Эту информацию также относят к атрибутам сообщения и соответствующего ему потока данных. Отметим, что различные атрибуты потоков данных становятся определенными на различных этапах проектирования УС РВ.

Отдельного рассмотрения заслуживают потоки видеоинформации. Каждый такой поток представляет собой последовательность видеокадров — массивов с содержимым растрового изображения, снабженных служебной информацией и, возможно, сжатых. В данной работе будем рассматривать видеокадры как сообщения, которые должны передаваться с заданной частотой, как правило соответствующей частоте обновления изображения на блоке-получателе.

2.2 Требования реального времени к передаче данных

Своевременная доставка данных является критическим фактором для корректного функционирования УС РВ. Различные виды вычислений обладают различными уровнями требовательности к точности соблюдения моментов отправки и доставки данных, соответственно к различным потокам данных предъявляются разные по уровню жесткости требования к передаче данных в реальном времени (далее — требования реального времени).

Базовым требованием реального времени к передаче данных является требование *периодичности*: сообщение должно передаваться от задачи-отправителя к задаче-получателю не реже, чем один раз в каждый свой период (как правило, совпадающий с периодом задачи-отправителя).

При построении периодических цепочек (конвейеров) вычислительных задач, различные задачи в составе которых расположены на различных блоках УС РВ, актуальным является ограничение на длительность передачи сообщения от отправителя к получателю, или *максимальная длительность передачи* сообщения.

Задача-получатель может предъявлять повышенные требования к регулярности доставки сообщения. Одним из таких требований является верхнее ограничение на *джиттер*, или разброс, в зависимости от итерации, моментов получения сообщения. С учетом того, что моменты формирования сообщения задачей-отправителем могут быть нерегулярными ("плавать" в рамках периода задачи), можно говорить о джиттере для:

- времени формирования сообщения;
- длительности передачи сообщения;
- времени доставки сообщения.

Первый вид джиттера относится к характеристикам выполнения задачи. Второй (джиттер длительности передачи сообщения) относится собственно к характеристикам среды передачи данных. Джиттер времени доставки сообщения, т.е. момента, когда оно достигает задачи-получателя, определяется первым и вторым видами джиттера. Таким образом, к требованиям реального времени к передаче сообщения относится джиттер длительности передачи сообщения (в дальнейшем, для краткости, просто джиттер сообщения).

Для реализации требования периодичности, а также ограничений сверху на длительность и джиттер передачи сообщения, сетевая среда должна ограничивать влияние одних потоков данных на временные характеристики передачи других потоков данных, а также обеспечивать устойчивость передачи данных к некорректному функционированию отдельных устройств-отправителей, в т.ч. к формированию ими нештатно больших потоков данных за счет превышения размера выдаваемых сообщений и/или частоты их выдачи. В предложенном ниже подходе на основе ПКС, по аналогии с AFDX и FC-EA-ASM-RT,

это ограничение влияния реализуется с использованием механизма виртуальных каналов.

2.3 Требования, связанные с поддержкой реконфигурации

Под реконфигурацией УС РВ понимают совокупность следующих изменений:

- изменение состава выполняемых вычислительных задач;
- изменение привязки задач к модулям и блокам УС РВ (миграция задач);
- изменение состава задействованных аппаратных ресурсов УС РВ (каналов передачи данных, блоков, модулей в составе блоков).

Реконфигурация требуется в следующих основных случаях:

- при отказах сетевого оборудования и вычислительных модулей;
- при смене режима работы УС РВ.

Примером смены режима работы УС РВ может служить переключение бортовой УС РВ летательного аппарата из наземного режима в режим полета. Штатная реакция УС РВ на отказ отдельных компонентов оборудования также может рассматриваться как смена режима — переход в один из сбойных режимов.

В зависимости от области применения, к УС РВ предъявляются различные требования по поддержке реконфигурации. Для авиационных УС РВ эти требования достаточно жесткие, поскольку при выходе из строя части оборудования требуется обеспечение "постепенной деградации" с поддержкой наиболее критических функций и отказом от второстепенных функций. В то же время от системы автоматизации производства может не требоваться поддержка "постепенной деградации", поскольку выход из строя одной ступени конвейера приводит к остановке всего конвейера.

При реконфигурации УС РВ изменяется состав и характеристики потоков данных в УС РВ. За счет изменения состава задач изменяется и состав потоков данных — некоторые из потоков устраняются, другие — добавляются. При миграции задач изменяются блоки и модули, являющиеся отправителями и/или

получателями сообщений. Это требует реконфигурации сетевой среды — добавления/удаления виртуальных каналов, изменения их маршрутов.

Хотя набор сбойных режимов для отказа единичного компонента оборудования может быть рассчитан заранее и загружен в память УС РВ до начала ее функционирования (число сбойных режимов — порядка нескольких десятков, по числу основных компонентов аппаратуры УС РВ), число вариантов для двух и более отказов возрастает экспоненциально. Тем самым необходимо либо заранее рассчитывать "обобщенные" сбойные режимы, каждый из которых рассчитан на целую категорию множественных отказов, либо динамически рассчитывать реакцию на конкретную комбинацию сбоев в ходе работы УС РВ. Первый вариант приводит к неиспользованию в конкретном обобщенном режиме части имеющихся (не отказавших) ресурсов УС РВ; второй требует расширения функциональности УС РВ за счет поддержки расчета реакции на конкретные комбинации сбоев (возможно, на основе заранее заготовленных шаблонов).

Реконфигурация УС РВ, как и прочие действия УС РВ как системы реального времени, должна выполняться в рамках заданных временных ограничений. Особенно важным этот фактор является при реагировании на сбои УС РВ в связи с их непредсказуемостью и потенциально катастрофическими последствиями. Тем самым, основным требованием к реконфигурации СПД на основе ПКС является переход к целевой конфигурации в рамках заданных временных ограничений. Для примера, в современных бортовых УС РВ летательных аппаратов допустимое время реконфигурации при сбоях составляет несколько десятков миллисекунд.

Итого, для поддержки реконфигурируемости УС РВ среда передачи данных должна удовлетворять следующим основным требованиям:

- поддержка изменения в процессе функционирования УС РВ состава и характеристик потоков данных через СПД, с сохранением выполнения требований реального времени;
- поддержка как статического (заранее определенные таблицы), так и динамического (на основе вычислений, выполненных в ходе функционирования УС РВ) изменения состава и характеристик потоков данных;
- поддержка указанных изменений в реальном масштабе времени, т.е. в пределах заданных временных ограничений.

Под характеристиками потоков данных здесь понимаются в первую очередь привязка потоков к физическим каналам передачи данных, а также привязка задач-отправителей и задач-получателей к блокам (модулям) УС РВ.

3. Предлагаемый подход к организации обмена данными в реальном времени

В настоящей работе предлагается подход к организации в УС РВ обмена данными в реальном времени по ПКС-сети. Этот подход можно охарактеризовать следующим образом:

- для контроля трафика и разделения пропускной способности между потоками данных используется механизм виртуальных каналов;
- организация контроля трафика виртуальных каналов происходит по аналогии с сетями AFDX и FC-AE-ASM-RT, за счет скоординированной работы абонентов-отправителей и коммутаторов;
- сохраняется применимость существующих подходов к оценке задержек и джиттеров передачи сообщений, разработанных для сетей AFDX [17, 18];
- контроллер сети функционирует в проактивном (пассивном) режиме;
- поддерживается динамическое изменение набора виртуальных каналов (включая изменение их параметров и маршрутов) без прекращения передачи данных по не изменяемым виртуальным каналам.

Использование известной и практически отработанной в существующих бортовых сетях AFDX и FC-AE-ASM-RT схемы управления трафиком, с одной стороны, ориентировано на упрощение возможной модернизации существующих УС РВ и создания новых УС РВ на основе существующих, а с другой стороны, позволяет использовать известные подходы к обоснованию соблюдения требований реального времени к передаче данных через сеть.

Проактивный режим функционирования контроллера гарантирует отсутствие избыточного трафика между коммутаторами и контроллером, за исключением трафика,

необходимого для мониторинга и перенастройки сети. Это существенно как с точки зрения минимизации нагрузки на сеть, так и для обеспечения прогнозируемости времен передачи данных через сеть, поскольку не требуется модифицировать существующие методы оценки задержек и джиттера для учета воздействия служебного трафика.

Предложенная схема использует стек протоколов TCP/IP за исключением маршрутизации на основе IP-адресов. Передаваемое сообщение разделяется на множество UDP-пакетов, каждый из которых соответствует одному кадру канального уровня (т.е. Ethernet). Эти пакеты посылаются узлом-отправителем в соответствии со схемой, используемой в сетях AFDX [13]. Номер виртуального канала указывается в поле *vlan* заголовка пакета. Маршрутизация пакета коммутаторами производится на основе номера виртуального канала, как и в сетях AFDX и FC-AE-ASM-RT. Для осуществления маршрутизации в таблицы коммутаторов заносятся специальные правила, которые в зависимости от значения поля с номером виртуального канала отправляют пакет на соответствующий выходной порт коммутатора. Пакеты, размер которых превышает заданный лимит для виртуального канала, так же как и пакеты, пришедшие на порт, не соответствующий указанному в их заголовках виртуальному каналу, сбрасываются коммутатором.

Контроль трафика производится при помощи измерителей (meter) — технологии, введенной в протоколе OpenFlow 1.3 и поддерживаемой в ПКС-коммутаторах [19]. Измеритель ассоциируется с потоком данных; в рассматриваемом случае поток определяется как последовательность принятых пакетов с определенным номером виртуального канала, т.е. пакетов заданного виртуального канала. Для каждого измерителя на коммутаторе определен свой набор интервалов скоростей потока. В зависимости от принадлежности текущей измеренной скорости потока к одному из интервалов, к полученному пакету применяется заданное для этого интервала действие, например, дальнейшая передача или сброс.

Способ измерения скорости потока не определяется протоколом OpenFlow 1.3 и зависит от реализации механизма измерителей на конкретном коммутаторе. В рамках предлагаемой в данной работе схемы передачи данных, от коммутатора требуется поддержка измерения скорости потока на основе алгоритма текущего ведра, аналогичного используемому в AFDX. Это позволяет контролировать не только непревышение скоростью потока

пропускной способности виртуального канала, но и то, что джиттер передачи сообщений не превышает максимально допустимого. Примером коммутатора, поддерживающего требуемую схему расчета скорости потока, является программный коммутатор *ofsoftswitch13* [19].

Измерители настраиваются на коммутаторах при помощи протокола OpenFlow следующим образом. Для каждого коммутатора, для каждого виртуального канала, проходящего через этот коммутатор:

- задается измеритель виртуального канала при помощи сообщения *FlowMod*;
- измерителю при помощи сообщения *MeterMod* задается максимальное значение кредита и интенсивность потока (в качестве полей *burst_size* и *rate*, соответственно).

В соответствии с предлагаемым подходом, виртуальный канал в ПКС имеет тот же набор параметров, что и виртуальный канал сетей AFDX. Поскольку схема выдачи данных абонентом в физическую линию и схема контроля трафика на коммутаторе аналогичны используемым в AFDX, для предлагаемого подхода применимы разработанные для сетей AFDX схемы оценки задержек и джиттера передачи сообщений через сеть [17, 18], а также методы построения систем виртуальных каналов [14].

4. Программная реализация и апробация подхода

Предложенный подход к организации обмена данными в реальном времени через ПКС-сеть реализован в виде приложения для сетевого контроллера RUNOS [20]. Структура приложения показана на рисунке 1.

Служебная часть приложения реализует программирование сетевых коммутаторов с целью добавления / удаления / модификации виртуальных каналов. При изменении на коммутаторе правил, относящихся к конкретному виртуальному каналу, правила для других виртуальных каналов не меняются, что позволяет продолжать передачу данных через эти виртуальные каналы.

Управляющая часть: отвечает за построение начальных маршрутов виртуальных каналов или загрузку готовых маршрутов, рассчитанных заранее другими средствами, такими как

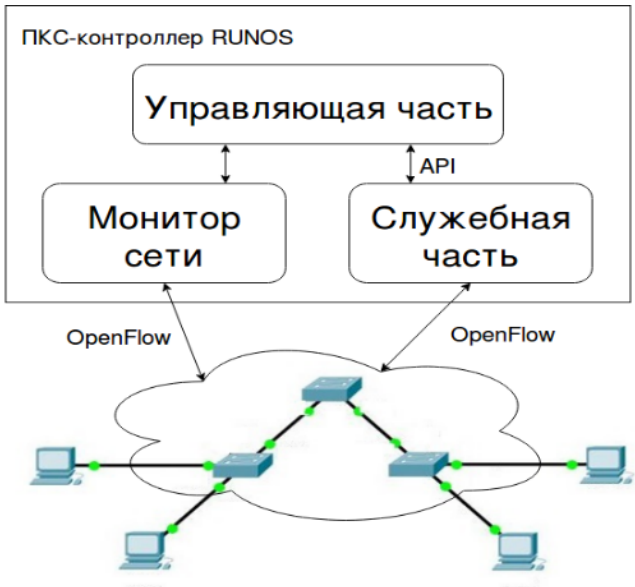

Рис. 1: Структура программного средства для сетевого контроллера

описанное в [14]; выполняет построение новых маршрутов виртуальных каналов при отказах компонентов сети или при миграции задач. Также управляющая часть выполняет расчет параметров виртуальных каналов для реализации требований качества сервиса.

Монитор осуществляет наблюдение за состоянием коммутаторов, линий связи, а также оконечных систем посредством периодической рассылки служебных сообщений через сеть. В случае выявления недоступности (отказа) компонента сети, монитор передает информацию управляющей части, которая выполняет реконфигурацию сети.

Приложение для контроллера реализовано на языке C++.

При апробации подхода использовалась виртуальная среда, построенная на основе эмулятора сети *mininet*, ПКС-контроллера RUNOS и программного коммутатора *ofsoftswitch13*.

Апробация подхода имела следующие цели:

- оценить задержки и джиттер передачи сообщений через сеть, подтвердить их соответствие расчетам по методикам, принятым для сетей AFDX;
- подтвердить, на основании полученных оценок, что

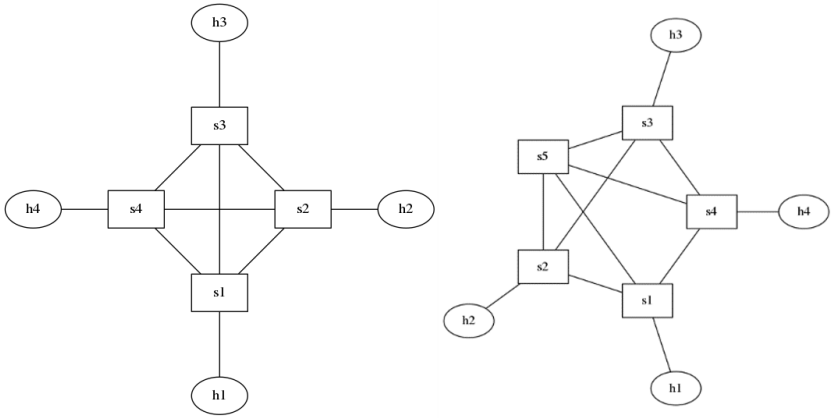

Рис. 2: Топологии сети для апробации подхода

предложенная схема позволяет осуществлять передачу различных потоков данных с гарантированной пропускной способностью для каждого из потоков;

- подтвердить, что попытки узла передавать данные по виртуальному каналу со скоростью, превышающей максимально допустимую пропускную способность, блокируются коммутатором (т.е. пакеты-нарушители сбрасываются);

- провести динамическую модификацию набора виртуальных каналов в соответствии с различными сценариями, типичными для отказов элементов сети УС РВ, а также подтвердить, что передача данных по виртуальным каналам, не затронутым модификацией, не прерывалась (отсутствует сброс пакетов этих каналов или искажение временных характеристик их передачи).

В качестве рабочей нагрузки на сеть использовались:

1. набор виртуальных каналов из работы по проектированию сетей AFDX [21];

2. набор виртуальных каналов для 100 потоков данных, создающий загрузку сетевых каналов порядка 80% (эти характеристики соответствуют современной бортовой УС РВ).

Исследовалась работа подхода как на симметричной (рис. 2 слева), так и на асимметричной (рис. 2 справа) топологиях сети, а

также на топологиях из [21]. Пропускная способность сети была установлена в 100 Мбит/с, чтобы соответствовать характеристикам сети из [21].

Проведенные эксперименты показали, что для рассмотренных потоков данных и топологий сети предложенная схема передачи данных в ПКС гарантирует обеспечение требуемой пропускной способности для каждого потока данных с задержкой и джиттером, удовлетворяющим заданным верхним ограничениям. В экспериментах, где некоторые абоненты формировали трафик, выходящий за ограничения для заданного виртуального канала (слишком большие пакеты и/или слишком малые интервалы времени между пакетами), коммутатор осуществлял сброс пакетов этого виртуального канала, тем самым предотвращая превышение ограничений на пропускную способность линии и/или искажение временных характеристик передачи сообщений через другие виртуальные каналы.

При проверке возможностей по динамической модификации набора виртуальных каналов были использованы сценарии, соответствующие реконфигурации сети при выходе из строя: коммутатора; линии связи, соединяющей два коммутатора; линии связи, соединяющей абонента с коммутатором; порта абонента или коммутатора. Проверялись в том числе кратные отказы, с последовательным выходом из строя до трех линий и одного коммутатора, но сохранением связности сети. Новые маршруты виртуальных каналов рассчитывались жадным алгоритмом в составе управляющего приложения на контроллере. Этот алгоритм основан на предложенном в [21] алгоритме построения виртуальных каналов в сети AFDX, за исключением фазы агрегации каналов. Алгоритм не является неотъемлемой частью предложенного подхода к построению сети и в данной работе не рассматривается.

В процессе реконфигурации не происходила потеря пакетов на виртуальных каналах, не затронутых реконфигурацией (т.е. сохранивших маршрут и параметры). Это связано с неизменностью относящихся к этим виртуальным каналам правил в сетевых коммутаторах. В случае отказа линии или порта, в процессе реконфигурации терялось до 5 пакетов; в случае отказа коммутатора — до 12 пакетов. Все потери пакетов происходили в виртуальных каналах, которые изменили свои маршруты. Потери связаны с последовательным перепрограммированием коммутаторов для их настройки на новый маршрут виртуального канала.

5. Применение подхода для связи с другими УС РВ и с периферийными ЦОД

Предложенный подход к организации обмена данными в реальном времени в ПКС-сети УС РВ может быть применен и для обмена данными между различными взаимодействующими УС РВ. Для информационного сопряжения нескольких УС РВ необходимо определить потоки данных, которые должны передаваться между ними, и выделить для этих потоков данных ресурс в ПКС-сетях сопрягаемых УС РВ. Применение подхода инвариантно к внешнему или внутреннему расположению сетевого абонента относительно УС РВ, также (по аналогии с сетями AFDX) не требуется синхронизация часов между взаимодействующими УС РВ. Защита УС РВ-получателя данных от превышения, со стороны УС РВ-отправителя, зарезервированной под виртуальные каналы пропускной способности обеспечивается средствами ПКС-коммутатора. Для сохранения временных характеристик потоков данных при передаче этих потоков через сеть, связующую различные УС РВ, эта сеть должна представлять собой ПКС, в которой функционирует контроллер с поддержкой предлагаемой схемы обмена данными.

Аналогичным образом может быть организована передача данных между УС РВ и периферийным ЦОД, на который перенесена часть вычислений, выполняемых в интересах УС РВ. В рамках ЦОД должен функционировать программный "агент", осуществляющий прием и передачу данных в виде периодических сообщений, помеченных номерами виртуальных каналов.

Заключение

В данной работе предложен подход к организации обмена данными в управляющих системах реального времени (УС РВ). Подход основан на применении программно-конфигурируемых сетей с поддержкой виртуальных каналов и позволяет использовать подходы к обоснованию соблюдения требований реального времени к передаче данных через сеть, разработанные для сетей AFDX. При этом предложенный подход обладает большими, чем в сетях AFDX и FC-AE-ASM-RT (также использующих виртуальные каналы), возможностями по реконфигурации сети в процессе ее функционирования.

Апробация подхода в виртуальной сетевой среде подтвердила

соблюдение требований реального времени в ходе передачи данных, в т.ч. в процессе реконфигурации сети — для тех виртуальных каналов, которые сохраняют маршруты и другие параметры.

Подход может быть применен не только для организации информационного обмена внутри УС РВ, но и для информационного сопряжения нескольких УС РВ, а также для обеспечения связи УС РВ с периферийным ЦОД.

Литература

1. Парамонов П.П., Жаринов И.О. *Интегрированные бортовые вычислительные системы: обзор современного состояния и анализ перспектив развития в авиационном приборостроении* // Научно-технический вестник информационных технологий, механики и оптики. – 2013. – № 2 (84). – С. 1–17.

2. Rodgers D., Driever P., Peterson D., Carlson C. *Inside a Modern Fibre Channel Architecture* [Электронный ресурс] URL: https://fibrechannel.org/wp-content/uploads/2021/09/Inside-Modern-Fibre-Channel-Architecture-Part-1-Final.pdf (дата обращения: 14.09.2023)

3. ARINC Specification 429. Part 1-17. – Airlines Electronic Engineering Committee. – 2004.

4. ISO 11898-1:2015. Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling. – ISO. – 2015.

5. Смелянский Р.Л. *Программно-конфигурируемые сети* // Открытые системы. – 2012. – № 9. – С. 15–26.

6. Software-Defined Networking: The New Norm for Networks // Open Networking Foundation. [Электронный ресурс] URL: https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/ (дата обращения: 14.09.2023)

7. Keller T., Alexander J. *Applications of ARINC 818 in Avionics Video Systems* // SAE International Journal of Aerospace. – 2010. – Vol. 2 – No. 1. – P. 95–102.

8. Хвощ С.Т., Амаду Х.Х. *Промышленные сети на основе стандарта MIL-STD-1553B* // Современные технологии автоматизации. – 1999. – №1. – С. 42–45.

9. ISO/IEC 14165-122:2005. Information technology – Fibre Channel – Part 122: Arbitrated Loop-2 (FC-AL-2). – ISO. – 2005.

10. ISO/IEC TR 14165-313:2013. Information technology – Fibre Channel – Part 313: Avionics Environment – Anonymous Synchronous Messaging (FC-AE-ASM). – ISO. – 2013.

11. ISO/IEC TR 14165-312:2009. Information technology – Fibre Channel – Part 312: Avionics environment upper layer protocol MIL-STD-1553B Notice 2 (FC-AE-1553). – ISO. – 2009.

12. Осипов Ю. С., Першин А. С., Пустовой Ю. В. *Способ передачи информации в реальном времени с использованием локальных сетей ограниченного размера на базе модификации протокола FC-AE-ASM.* – Пат. 2536659 РФ. – Бюл. №36. – 2013.

13. ARINC Specification 664. Aircraft Data Network, Part 7: Avionics Full Duplex Switched Ethernet (AFDX) Network. – Airlines Electronic Engineering Committee. – 2009.

14. Балашов В.В. *Семейство систем автоматизации проектирования бортовых вычислительных систем реального времени* // Программные продукты, системы и алгоритмы. – 2017. – № 4. – С. 1–19.

15. Balashov V., Antipina E. *Iterative co-scheduling of tasks and their data exchange through virtual link-based packet switched network* // Proc. 3 International Scientific and Technical Conference Modern Computer Network Technologies (MoNeTec-2020). – 2020.

16. Лазутин Ю.М., Чертов Г.А., Родиков А.В., Рогожкин В.А. *Методология построения потоков данных в сложных аппаратно-программных комплексах* // Программные продукты и системы. – 2011. – № 4. – С. 31–35.

17. Boyer M., Fraboul C. *Tightening End to End Delay Upper Bound for AFDX Network Calculus with Rate Latency FIFO Servers Using Network Calculus* // Proc. 2008 IEEE International Workshop on Factory Communication Systems. – 2008. – P. 11–20.

18. Bauer H., Scharbarg J.-L., Fraboul C. *Applying and Optimizing Trajectory Approach for Performance Evaluation of AFDX Avionics Network* // Proc. IEEE Conference on Emerging Technologies and Factory Automation (ETFA-2009). – 2009. – P. 1–8.

19. OpenFlow Switch Specification, Version 1.3.4 // Open Networking Foundation. – 2014.

20. Shalimov A., Nisovtsev S., Morkovnik D., Smeliansky R. *The Runos OpenFlow Controller* // Proc. 2015 Fourth European Workshop on Software Defined Networks (EWSDN). – 2015. – P. 103-104.

21. Вдовин П.М., Костенко В.А. *Организация передачи сообщений в сетях AFDX* // Программирование. – 2017. – № 1. – С. 5–20.

Бахмуров А.Г., Голубкова М.С.

Разработка Android приложения, имитирующего датчики интернета вещей

Введение

В последние годы интернет вещей (Internet of Things, IoT) стал неотъемлемой частью технологического прогресса. IoT представляет собой сеть устройств, соединенных между собой, которые могут обмениваться данными и функционировать автономно без человеческого участия. Одной из наиболее перспективных областей применения IoT является динамично развивающийся рынок интернета медицинских вещей (Internet of Medical Things, IoMT).

IoMT — это новый способ реализации медицинских услуг и внедрения инноваций в здравоохранении. Системы мониторинга, дистанционной диагностики, устройства для жизнеобеспечения и управление лекарствами — все это включено в понятие интернета медицинских вещей. С помощью IoMT предоставляются новые возможности для мониторинга здоровья пациентов, снижения затрат на медицинское обслуживание и повышения качества жизни пациентов.

Выпускные квалификационные работы студентов ВМК МГУ Фролова Александра[1] и Князева Егора[2] посвящены факультетскому проекту по развитию сервиса сбора и обработки медицинской телеметрии. Серверная часть проекта позволит медицинским работникам дистанционно наблюдать за состоянием пациентов и своевременно реагировать на серьёзные изменения в их показателях здоровья, а клиентская часть обеспечит сбор данных с носимых устройств, отображение информации о них пользователю и их передачу на сервер.

Применение IoMT включает разнообразные устройства, такие как умные часы, наушники, браслеты, датчики температуры, сердечного ритма и артериального давления, то есть различные приборы для дистанционной диагностики и лечения пациентов. Количество новых устройств на рынке стремительно растёт, например, в рамках обзора устройств были обнаружены такие интересные приборы, как серьга, которая отслеживает уровень сахара в крови и даёт обратную связь в реальном времени[5], или наушники, регистрирующие электрокардиограмму[4].

Сервис сбора и обработки медицинской телеметрии должен иметь возможность быстро адаптироваться под появление новых устройств. Данная статья описывает первые шаги в разработке удобного инструмента, позволяющего развивать сервис.

1. Постановка задачи

В статье описывается решение задачи, целями которой являлось:

- разработать приложение, имитирующее поведение датчика Интернета вещей в части передачи данных по интерфейсу Bluetooth LE, для исследования работоспособности клиентской части сервиса сбора и обработки медицинской телеметрии;
- создать возможности для выбора типа имитируемого датчика и настройки режима его работы;
- с помощью созданного имитатора датчика продемонстрировать возможность нагрузочного и функционального тестирования существующей клиентской части сервиса.

1.1 Потребности тестирования

Использование реальных датчиков может быть недостаточным при тестировании работоспособности приложений, взаимодействующих с устройствами IoMT.

Использование лишь реальных датчиков делает невозможным:

- имитацию датчиков, недоступных в настоящий момент;
- тестирование приложения при работе с несколькими датчиками, нужного количества которых может даже не быть в наличии;
- отслеживание возникновения ошибок на уровне приложений, т.к. мы не можем знать сколько пакетов фактически передали, например, Smart часы;
- контроль передачи всех пакетов для устройства в разных режимах работы, например в спящем режиме;
- имитацию некорректной работы датчика или же его новых возможностей, планируемых к реализации.

2. Рассмотрение существующих решений

Учитывая упомянутые ранее (в п. 1.1) требования к возможностям приложения, были рассмотрены следующие существующие решения:

HCI sniffer - это инструмент, позволяющий перехватывать пакеты данных между уровнями контроллера и хоста стека протоколов Bluetooth. Сниффер обеспечивает получение данных до их попадания на уровень приложения, но не предоставляет никаких возможностей для нагрузочного и функционального тестирования. Подобный инструмент существует в каждом смартфоне Android[6], а также на рынке есть множество аппаратных наблюдателей такого типа, их цена варьируется от 50 до 40 000 долларов[7].

BlueZ - это официальная реализация стека протоколов Bluetooth для Linux. Позволяет устанавливать соединение между ПК и смартфоном и управлять им с помощью командной строки. С его помощью можно, например, писать Python скрипты[8], управляющие соединением, что позволяет создать десктопное приложение, но использование компьютерных программ менее удобно для рядового пользователя, а главное - делает затруднительным параллельное тестирование, требующее несколько BLE устройств.

Nordic nRF Connect for Android[9] - это мобильное приложение с широкими возможностями. Оно позволяет настраивать собственную конфигурацию BLE сервера, рассылать широковещательные пакеты, обеспечивает одновременное сканирование, адвертайзинг[1] и поддержку сразу нескольких подключений. Однако, несмотря на длинный список возможностей, приложение всё же не предоставляет возможностей для автоматизированного функционального и более гибкого нагрузочного тестирования. Так же оно позволяет записывать трассу, но записывает пакеты целиком, что избыточно для наших целей, и добавит лишь неудобства.

Стенд полунатурного моделирования - это среда для наблюдения поведения программ во взаимодействии с моделями периферийных устройств в целях тестирования и оценки производительности этих программ. Для наших целей можно было построить подобную среду моделирования, основываясь на опыте, описанном в статье [10]. Стенд бы позволил легко создавать модели устройств и организовывать выполнение передачи данных по BLE. Однако готовых решений такого рода нет в открытом доступе, а

[1] определения см. в описании работы канального уровня BLE п. 3.3

самостоятельное построение такого стенда под наши нужды было бы избыточным решением.

Решения, удовлетворяющие описанным выше потребностям тестирования, на данный момент неизвестны, их нет в открытом доступе.

3. Обзор архитектуры BLE

Прежде чем сформулировать технические требования к разрабатываемому приложению, необходимо определить основные термины и кратко описать работу Bluetooth.

3.1 Bluetooth Classic и BLE

В настоящее время существует два типа устройств с поддержкой Bluetooth - Bluetooth Classic и Bluetooth Low Energy (BLE) — это два отличающихся протокола беспроводной связи:

- Bluetooth Classic спользуется для потоковой передачи данных, не оптимизирован для низкого энергопотребления, но поддерживает большую скорость передачи.
- Bluetooth Low Energy (BLE), т.е. Bluetooth с низким энергопотреблением. Используется в сенсорах, управлении устройствами и приложениях, не требующих передачи больших объемов данных, предназначен для применения в малопотребляющих устройствах с большими интервалами между передачей данных.

Таким образом, Bluetooth Classic позволяет посылать данные в больших объёмах и с большей частотой, однако это требует больших энергетических трат, поэтому используется, например, в беспроводных громкоговорителях, автомобильных информационно-развлекательных системах и наушниках. BLE чаще всего применяется в приложениях, чувствительных к энергопотреблению или в устройствах, передающих небольшие объемы данных с большими перерывами между передачами (например, разнообразные сенсоры параметров окружающей среды или управляющие устройства, такие как беспроводные выключатели). Поэтому в рамках данной работы было принято решение использовать BLE.

3.2 Описание стека протоколов BLE

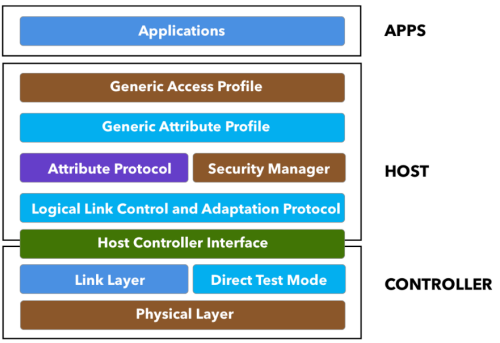

Рис. 1: Архитектура BLE

На Рис. 1 представлена архитектура BLE[13]. Остановимся на трёх уровнях: канальный (Link Layer), уровень протокола атрибутов (ATT) и общий профиль атрибутов (GATT).

3.3 Канальный уровень

На канальном уровне возможны следующие состояния устройства BLE:

- **Standby**: когда устройство не передает и не принимает никаких данных.
- **Advertising**: устройство посылает широковещательные пакеты для обнаружения другими устройствами.
- **Scanning**: ищутся устройства, посылающие широковещательные пакеты.
- **Initiating**: начинается процесс установки соединения с устройством, находящимся в состоянии advertising.
- **Connected**: одно устройство установило соединение с другим и регулярно обменивается с ним информацией.

Определив понятия сканирования и адвертайзинга, можно определить роли устройств:
Периферийное устройство – устройство, которое объявляет о своем присутствии путем адвертайзинга и принимает запросы на соединение от центральных устройств.

Центральное устройство – устройство, которое обнаруживает периферийные устройства и считывает передаваемую ими информацию. Оно также может устанавливать соединение с одним или несколькими устройствами одновременно.

3.4 Протокол атрибутов (ATT)

Протокол ATT определяет, в каком виде сервер представит свои данные клиенту и как эти данные будут структурированы.

Сервер - устройство, которое предоставляет данные, которые содержит или которыми управляет, а также получает входящие команды от связанного устройства и отправляет ответы и уведомления.

Клиент - устройство, которое взаимодействует с сервером с целью прочитать предоставляемые сервером данные и/или для того, чтобы контролировать его поведение. Это устройство, которое посылает команды и запросы и получает входящие уведомления.

Следует обратить внимание, что роли устройств на уровнях ATT (клиент и сервер) и GATT (центральное и периферийное) никак не взаимосвязаны, т.е. например центральное устройство может быть как клиентом, так и сервером.

Данные, предоставляемые сервером, сгруппированы в атрибуты. Атрибут это общий термин для любых типов данных, предоставляемых сервером. К атрибутам относятся сервисы, характеристики и дескрипторы.

3.5 Общий профиль атрибутов (GATT)

Профиль GATT определяет формат сервисов и их характеристик, а также процедуры, используемые для взаимодействия с этими атрибутами, такие как обнаружение сервисов, чтение и запись характеристик, уведомления.

Сервис это группа из одного или большего числа атрибутов, часть (или все) из которых являются характеристиками. Он предназначен для группировки связанных атрибутов, удовлетворяющих специфической функциональности сервера.

Характеристика всегда является частью сервиса и предоставляет часть тех данных, которые сервер хочет предоставить клиенту. Характеристика содержит набор элементов (её свойства и разрешения на доступ, значение и набор дескрипторов), которые облегчают работу с содержащимися в характеристике данными:

- Свойства: представлены набором битов, определяющих то, каким образом значение характеристики может использоваться (например, чтение, запись, запись без ответа, уведомление).

- Дескрипторы: используются для хранения информации, связанной со значением характеристики. Примеры использования: расширенные свойства, пользовательское описание, поля, используемые для подписки на уведомления и индикации.

Сервисы, характеристики и дескрипторы, являясь атрибутами, имеют тип атрибута (универсальный уникальный идентификатор, UUID) - это 16-битное (в случае стандартных атрибутов) или 128-битное число (в случае атрибутов, определенных разработчиком устройства). Для того чтобы клиент мог получить информацию о каком-либо атрибуте, он должен знать его UUID, поэтому для согласования набора атрибутов у имитируемого устройства с клиентским приложением, все необходимые UUID хранятся в конфигурационном файле.

3.6 Формат данных

Данные, отправляемые сервером имеют определённую структуру, она определяется либо стандартами Bluetooth SIG[15], либо производителем конкретного устройства IoT.

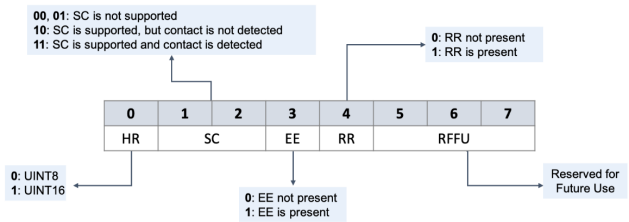

Рис. 2: Структура служебного байта

На Рис. 2 [16] для примера изображена структура служебного байта сервиса частоты сердцебиения. Здесь SC (Sensor Contact) - состояние контакта сенсора с кожей, EE (Energy Expended) - наличие в сервисе характеристики, отображающей расход электроэнергии, RR (RR-Intervals) - поддержка датчиком рассчёта

вариабельности сердечного ритма. На данный момент вся эта информация не представляет для нас интереса. В текущей реализации единице равен только самый первый бит, чтобы не ограничивать пользователя в размере передаваемых данных.

4. Технические требования к разрабатываемому приложению

На основе оценки достоинств и недостатков существующих решений, а также анализа потребностей тестирования, сформулированы следующие технические требования к разрабатываемому приложению:

- выбор или загрузка конфигурации имитируемого устройства;
- задание параметров передачи данных (интервал значений и частота обновления);
- поддержка адвертайзинга и соединения;
- возможности для работы с несколькими сервисами, включающими несколько характеристик, одновременно;
- передача данных с помощью BLE;
- запись отправляемых данных и временных меток отправки;
- выгрузка трассы для последующего анализа.

5. Полученные результаты

5.1 Пользовательский интерфейс

Начальный экран приложения позволяет пользователю выбрать тип имитируемого устройства. Список устройств был составлен на основе существующих на рынке устройств интернета медицинских вещей. На выбор предложены:

- Hexoskin[17] - смарт-рубашка с открытыми данными для мониторинга различных показателей здоровья человека (частота сердечных сокращений, частоты дыхания и т.д.) и других измерений активности, таких как подсчет шагов и частота вращения педалей.

- Ritmer[18] - смарт-браслет для мониторинга сердечной активности.
- Dexcom[19] - глюкометр с открытым исходным кодом.
- Smart весы - имитация поведения таких весов, как, например, Picooc[20] или Mi Body Composition Scale[21].
- Smart часы - имитация поведения таких часов, как Xiaomi Mi Smart Band[22].
- Универсальное устройство - возможность для пользователя составлять свой список передаваемых данных.

При выборе конкретного устройства (любого, кроме универсального) также выбирается и соответствующий конфигурационный файл, задающий список предоставляемых устройством данных. В настоящий момент такие файлы написаны для Hexoskin и Smart часов.

При переходе на промежуточный экран "Универсальное устройство" приложение устанавливает http-соединение с серверной частью сервиса сбора и обработки медицинской телеметрии и получает от сервера хранящиеся на нём конфигурационные файлы[11]. В соответствии с полученным списком на экране динамически[12] генерируются кнопки для выбора имитируемого устройства.

Таким образом для имитации устройства по индивидуальным запросам необходимо прежде всего написать конфигурационный файл, задающий список передаваемых данных. В настоящий момент приложение может получать только загруженные на сервер авторизированными пользователями файлы, однако в будущем можно будет предоставить возможность для загрузки конфигурационного файла, например, из памяти смартфона.

На втором экране в соответствии с конфигурационным файлом генерируется динамический список[12] всех предоставляемых данных (например, частота сердцебиения и/или частота дыхания), и для каждого показателя предоставляются поля для установки минимального и максимального значений и частоты их обновления.

5.2 Архитектура приложения

На основании описанных выше особенностей архитектуры BLE можно заключить, что разрабатываемое приложение должно быть, с точки зрения протокола BLE, периферийным устройством: оно должно рассылать широковещательные пакеты, пока не будет обнаружено центральным устройством, которое установит с ним соединение для считывания данных. Приложение также должно быть сервером: оно должно создавать GATT-таблицу (иерархия и отношения между различными атрибутами) для предоставляемых данных, а так же создавать эти данные и передавать клиенту. GATT-таблица создаётся на основании конфигурационного файла, задающего набор сервисов, характеристик в них и их UUID.

5.3 Генерация данных

После создания каждого сервиса в сопрограммах[14] для каждой характеристики запускаются функции, в которых асинхронно, с задержкой, определённой пользователем при настройке параметров, из промежутка значений, определённого пользователем, выбираются случайные числа. Число присваивается соответствующей характеристике, после чего подключенному устройству отправляется уведомление об обновлении этой характеристики. В данный момент в приложении реализована корректная генерация значений только для стандартного сервиса частоты сердцебиения.

5.4 Запись трассы

Для тестирования работы клиентского приложения необходимо сравнение полученных от датчика данных с данными, впоследствии поступившими на сервер, также интерес представляет временная задержка при передаче данных. Поэтому при подключении стороннего устройства к разрабатываемому приложению, все данные, отправляемые ему, записываются в текстовый файл вместе с названием характеристики и временем отправки. После окончания работы становится доступной кнопка SHARE LOGS, с помощью которой созданный файл можно отправить по почте или в любые мессенджеры.

6. Экспериментальное исследование

В рамках выпускной квалификационной работе студента ВМК МГУ Фролова Александра[1] проводились тесты над клиентской частью сервиса с использованием приложения, рассматриваемого в данной статье. Тестирование происходило по двум сценариям: тестирование на долговечность, которое позволяет проверить приложение на отсутствие сбоев и прерываний в течение продолжительного времени, и тестирование на масштабируемость, которое помогает определить, насколько приложение способно к работе с большим количеством датчиков. В результате экспериментов удалось установить, что при подключении большого числа устройств задержки ожидаемо растут, причем рост задержек составляет менее 10% медианного значения, откюда был сделан вывод об устойчивой работе клиентской части сервиса.

7. Перспективы развития

В дальнейшем можно реализовать:

- загрузку конфигурационного файла из памяти устройства;
- поддержку служебного байта для большего количества сервисов;
- сохранение в файл настроек, введённых пользователем, для повторного использования;
- поддержку передачи потоковых данных, например ЭКГ;
- получение пакетов от клиентского устройства (например, о входящих звонках);
- поддержка Bluetooth Classic.

Литература

1. Фролов А. В. *Разработка приложения сбора данных с нескольких устройств интернета вещей, подключенных одновременно, на примере медицинской телеметрии,* 2023

   `https://github.com/IoMT-LVK/papers/blob/main/client/4.%20Frolov_Diploma_Paper.pdf`

2. Князев Е. И. *Разработка серверной части приложения интернета вещей на основе принципов передачи репрезентативного состояния* 2023

   https://github.com/IoMT-LVK/papers/blob/main/server/4.%20Knyazev_Diploma_Paper.pdf

3. Голубкова М. С. *Разработка Android приложения, имитирующего датчики интернета вещей* 2023

   https://github.com/IoMT-LVK/papers/blob/main/client/5.Golubkova_Course_Paper.pdf

4. Heart.Zone A provider of innovative Electro-Cardio-Gram

   https://heart.zone/

5. Серьга, отслеживающая уровень сахара в крови и дающая обратную связь в реальном времени

   https://tectales.com/wearables-sensors/sense-glucose-earring-for-managing-diabetes.html

6. Medium *Bluetooth LE packet capture on Android* 2020

   https://medium.com/propeller-health-tech-blog/bluetooth-le-packet-capture-on-android-a2109439b2a1

7. Novel Bits *BLE Sniffer Basics + Comparison Guide* 2023

   https://novelbits.io/bluetooth-low-energy-ble-sniffer-tutorial/

8. Bluetooth® Technology Website *Bluetooth® Technology for Linux Developers* 2023

   https://www.bluetooth.com/bluetooth-resources/bluetooth-for-linux/

9. Nordic Semiconductor *nRF Connect for Mobile*

   https://www.nordicsemi.com/Products/Development-tools/nrf-connect-for-mobile

10. Бахмуров А.Г., Волканов Д.Ю., Смелянский Р.Л., Чемерицкий Е.В. *Интегрированная среда для анализа и разработки встроенных вычислительных систем реального времени* 2013
    https://istina.msu.ru/publications/article/4694706/

11. Фролов А. В. *Развитие клиентской части сервиса сбора и обработки медицинской телеметрии* 2022

    https://github.com/IoMT-LVK/papers/blob/main/client/3.%20Frolov_Couse_Paper.pdf

12. Android Developers *Create dynamic lists with RecyclerView* https://developer.android.com/develop/ui/views/layout/recyclerview

13. Afaneh, Mohammad for Novel Bits *Intro to Bluetooth low energy* 2018

14. Kotlin Help *Coroutines guide: Kotlin*

    https://kotlinlang.org/docs/coroutines-guide.html

15. Bluetooth® Technology Website *Specifications*

    https://www.bluetooth.com/specifications/specs/

16. Mariam Bahameish *Extracting Heart Rate Measurements from Bluetooth LE Packets* 2019

    https://mariam.qa/post/hr-ble/

17. Hexoskin Smart Shirts

    https://www.hexoskin.com/

18. Монитор сердечной активности Ritmer

    https://www.ritmer.ru/

19. Dexcom Continuous Glucose Monitoring

    https://www.dexcom.com/global

20. Умные весы Picooc

    https://picooc.ru/

21. Весы Mi Body Composition Scale

    https://ru.buy.mi.com/ru/item/3194100001

22. Xiaomi Mi Smart Band

    https://www.mi.com/ru/product/mi-smart-band-6/

Британков Е.С., Пашков В.Н.

Применение метода опорных векторов для обнаружения DoS-атак на контроллер в ПКС сетях

Введение

В программно-конфигурируемых сетях (ПКС, SDN) логически централизованное управление сетью, сетевым оборудованием и потоками данных осуществляется специальным программным обеспечением – сетевой операционной системой или контроллером с набором приложений [1]. Таким образом, контроллер, как центральный элемент управления в ПКС сетях, является единой точкой отказа. Отказ контроллера может быть вызван как естественными причинами, связанными с отказами сервера или его компонентов, ошибками в ПО сервера или ошибками в приложениях контроллера, так и носить преднамеренный характер в виде реализации атак на контроллер, нацеленных на выведение его из строя или блокировку. Поэтому для использования ПКС в реальных сетях в контуре управления должны быть предусмотрены соответствующие механизмы обеспечения отказоустойчивости контроллера. Проблема отказа контроллера, вызванного ошибками в ПО или отказами аппаратного обеспечения, обычно решается за счет внесения избыточности (резервирования контроллера, каналов управления и т.д.). Отказ контроллера или его блокировка, вызванная атаками на него, чаще всего решается за счет использования механизмов обнаружения атаки и ее источников, снижения ее активности и минимизации ее последствий для дальнейшего функционирования сети.

В работе рассматривается одна из самых актуальных проблем безопасности для контура управления в ПКС/OpenFlow сетях – DoS- и DDoS- атаки на ПКС контроллер. Под DoS атакой (атака типа "отказ в обслуживании") на контроллер в ПКС/OpenFlow сети обычно понимается совокупность действий злоумышленника, нацеленных на генерацию с некоторого узла сети (коммутатора) большого количества бесполезных (фиктивных) запросов к контроллеру с целью перегрузки его их обработкой и установлению в сети большого количества фиктивных маршрутов с соответствующей установкой множества фиктивных правил коммутации в таблицах потоков коммутаторов [2]. Атака может быть усилена за счет проведения такой атаки сразу с нескольких

узлов в сети, то есть реализации распределенной DoS атаки (DDoS). DoS- и DDoS- атаки на контроллер в программно-конфигурируемых сетях могут приводить к серьезным негативным последствиям для функционирования сети:

- увеличение времени обработки контроллером запросов от коммутаторов сети;
- увеличение времени создания новых маршрутов в сети;
- заполнение таблиц потоков OpenFlow коммутаторов фиктивными правилами;
- увеличение времени отклика пользовательских сетевых сервисов;
- полная или частичная недоступность сервисов для конечных пользователей.

Для практического внедрения ПКС в реальных сетях операторов связи, сетях ЦОД и предприятий крайне значимыми являются вопросы разработки методов и средств для обнаружения DoS- и DDoS- атак на контроллер, а также методов и средств противодействия им и смягчения и устранения последствий этих атак [3]. В данной работе предлагается подход на основе применения метода опорных векторов для обнаружения DoS-атаки на контроллер в ПКС/OpenFlow сети.

1. Постановка задачи

Пусть задана некоторая фиксированная топология ПКС/OpenFlow сети, в которой управление конфигурацией сетевых устройств (коммутаторов) осуществляется контроллером по протоколу OpenFlow [4]. Предположим, что один или несколько хостов в сети находятся под управлением злоумышленника или заражены вредоносным программным обеспечением. С этих хостов злоумышленник может вести DoS- или DDoS-атаку на контроллер, используя особенности установления новых маршрутов в ПКС сети на основе протокола управления OpenFlow.

Атака реализуется следующим образом: зараженный хост или множество хостов инициируют большое количество новых потоков данных в сети, изменяя случайным образом или по некоторому известному принципу основные поля заголовков пакетов (IP адреса отправителя и получателя, MAC адреса отправителя и получателя

и другие) с целью заставить коммутатор сети генерировать запросы (на установление новых фиктивных маршрутов в сети) к контроллеру в связи с отсутствием соответствующих правил в его таблице потоков. Успешность атаки может быть существенно выше, если:

- Скомпрометирован коммутатор сети и злоумышленнику известен перечень уже установленных правил на коммутаторе.
- Злоумышленнику известна топология сети и структура сети: злоумышленник может генерировать с хоста потоки данных, которые требуют построения длинных маршрутов в сети с соответствующей дополнительной вычислительной нагрузкой по их вычислению, генерации и установлению большего количества правил на коммутаторы на каждый запрос.

Поскольку вычислительные ресурсы контроллера ограничены физическими ресурсами сервера (CPU, RAM), на котором он запущен, то контроллер может не справиться с таким большим количеством запросов от коммутаторов. Большую часть вычислительной мощности контроллер будет затрачивать на обработку запросов, вызванных действиями злоумышленника, и не обрабатывать запросы на создание новых маршрутов в сети, вызванные появлением новых потоков данных от реальных пользователей (см. рис. 1). Таким образом, необходимо найти способ определять в сети зараженные хосты, с которых ведется DoS- или DDoS-атака на контроллер, чтобы в дальнейшем была возможность запустить механизмы по ее предотвращению и устранению последствий атак (удаление правил из таблиц коммутаторов для фиктивных потоков и др.). При этом время обнаружения вредоносных хостов должно быть достаточно небольшим, чтобы уменьшить влияние атаки на пользователей сети. Также поиск зараженных хостов должен допускать как можно меньше ошибок первого и второго рода.

В общем виде задачу обнаружения DoS (DDoS) – атаки на контроллер можно сформулировать следующим образом. Пусть в ПКС/OpenFlow сети под управлением контроллера C заданы:

1. $G(S, H, C, E)$ – граф программно-конфигурируемой сети, где S – множество коммутаторов, H – множество хостов, непосредственно подключенных к ПКС, C – контроллер сети, E – множество каналов связи.

2. $F = \{F_h | h \in H\}$ – множество всех потоков данных в сети (пользовательских и вредоносных).

3. $Y = \{M, S, U\}$ – множество классов хостов: вредоносный, подозрительный и пользовательский хост, $H = H_M \cup H_S \cup H_U$

Необходимо найти:

- множества H_M и H_S.

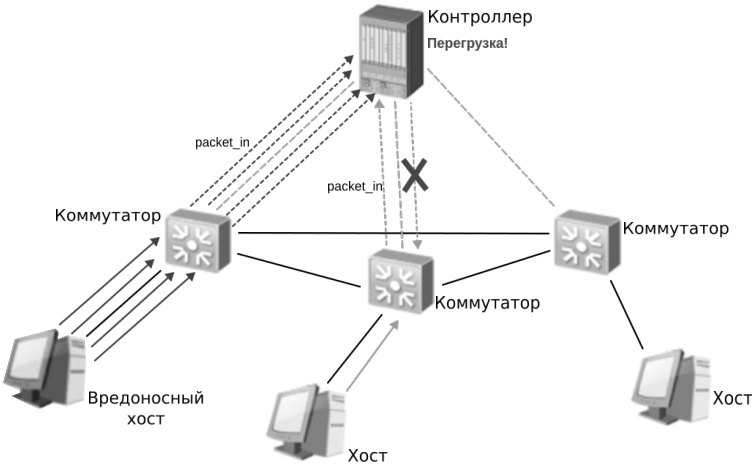

Рисунок 1. Пример DoS-атаки на контроллер в ПКС/OpenFlow сети

2. Обзор существующих методов обнаружения DDoS-атак на контроллер в ПКС

В данном разделе приводится обзор методов обнаружения именно DDoS-атак как класса, включающего DoS-атаки.

В статье [5] приводится описание метода, использующего автоэнкодер и нейронную сеть для обнаружения DDoS-атак на контроллер ПКС-сети. В данной работе используется анализ отдельных пакетов в сети. Нейронную сеть обучают на значениях признаков из датасета CICDDoS-2019, в качестве признаков используются размеры пакетов, их содержимое и различные метаданные, содержащиеся в датасете. Предлагаемый метод определяет, является ли пакет в сети частью DDoS-атаки. Для

оценки качества определялся показатель Accuracy (точность), достигший значения 0.99.

В статье [6] приводится описание метода обнаружения DDoS-атаки на контур передачи данных (заполнение каналов между коммутаторами вредоносным трафиком) с использованием метода опорных векторов. В данной работе анализируются потоки данных ПКС, а классифицируемыми объектами являются хосты. В качестве признаков используются количество потоков данных хоста (то есть созданных контроллером после получения запроса packet_in от данного хоста), количество IP, с которых приходили пакеты потоков хоста, скорость создания новых потоков данных, а также соотношение парных и непарных потоков. Модель обучается на датасете, собранном на стенде на основе Mininet. В рамках работы предложенный метод был реализован в качестве приложения для контроллера Ryu, а также экспериментально испытан (точность достигла значения 0.987).

В статье [7] приводится описание метода для обнаружения DDoS-атаки на контроллер ПКС при помощи различных методов машинного обучения (линейная регрессия, наивный байесовский классификатор, метод опорных векторов и др.). В данной работе также анализируются потоки данных ПКС, а классифицируемыми объектами являются хосты. В качестве признаков используются количество пакетов/байт в потоке, средний размер пакета потока и количество потоков к тому же хосту/к тому же порту хоста. Для обучения моделей авторами был собран датасет с использованием стенда на основе Mininet. Предложенные методы были реализованы в виде приложений для контроллера Floodlight, а также экспериментально исследованы. Максимальная точность 1.0 была достигнута при использовании метода k ближайших соседей, дерева решений и случайного леса.

В статье [8] приводится описание метода обнаружения DDoS-атак на контроллер с использованием метода опорных векторов. В данной работе анализируются значения признаков потоков, а классифицируются хосты в сети (на зараженные и пользовательские). Для классификации используются среднее количество пакетов/байт на поток, среднеквадратичное отклонение количества пакетов/байт в потоке и среднее кол-во IP, с которых приходили пакеты потоков хоста. Датасет для обучения модели был сгененрирован на стенде на основе Mininet. Предложенный метод был реализован в качестве приложения для контроллера OpenDaylight, а также экспериментально исследован: точность достигла 0.97. Также в рамках работы был рассмотрен вопрос

противодействия DDoS-атаке и блокировки вредоносных хостов.

По результатам обзора были выделены два подхода к обнаружению DDoS-атак – на основе анализа пакетов и на основе анализа потоков. Из них был выбран метод анализа потоков, поскольку он представляется менее затратным с точки зрения вычислительной мощности. В качестве модели машинного обучения был выбран метод опорных векторов, так как он специализируется на разделении данных на небольшое количество групп и хорошо показывает себя в рассмотренных работах для задачи обнаружения DoS атак. Также были выбраны признаки, которые используются в предлагаемом в данной работе подходе для классификации хостов.

3. Подход к обнаружению DoS-атак на контроллер на основе метода опорных векторов

В качестве признаков для определения фиктивных потоков данных на основе проведенного обзора были выбраны следующие метрики:

1. Количество потоков хоста (HFC) – количество потоков данных хоста в таблицах потоков коммутаторов сети (единица измерения – кол-во потоков).

2. Скорость создания потоков данных (SFE) – количество потоков данных хоста, созданных за последние N секунд (единица измерения – кол-во потоков). Вычисляется как $HFC - PrevHostFlowsCount + DelHostFlowsCount$, где $PrevHostFlowsCount$ – количество потоков данных хоста в таблицах потоков коммутаторов сети N секунд назад, $DelHostFlowsCount$ – количество потоков данных хоста, удаленных из таблиц потоков коммутаторов сети за последние N секунд.

3. Среднее количество пакетов на поток ($ANFP$) – среднее количество пакетов, прошедших через потоки данных хоста за последние N секунд (единица измерения – пакеты). Вычисляется по формуле

$$\frac{\sum\limits_{x_h \in X_h} (PacketCount - PrevPacketCount)}{HFC}$$

, где $PacketCount$ – количество всех пакетов, прошедших через поток x_h, $PrevPacketCount$ – количество пакетов, прошедших через этот поток N секунд назад и ранее.

4. Среднеквадратичное отклонение количества пакетов в потоке ($VNFP$) – среднеквадратичное отклонение количества пакетов, прошедших через потоки данных хоста за последние N секунд (безразмерная величина). Вычисляется по формуле:

$$\sqrt{\frac{\sum\limits_{x_h \in X_h} ((PacketCount - PrevPacketCount) - ANFP)^2}{HFC}}$$

В качестве используемой модели машинного обучения был использован метод опорных векторов. Основная идея этой модели состоит в построении разделяющей гиперплоскости между векторами объектов из разных классов с максимальным зазором, т.е. расстоянием между гиперплоскостью и ближайшим объектом к ней (предполагается, что чем больше зазор, тем меньше в среднем будет ошибка классификатора). Подробнее узнать о методе опорных векторов можно по ссылке [9].

Для получения класса хоста необходимо:

1. Собрать датасет из значений признаков вредоносных и пользовательских хостов.
2. Обучить на полученном датасете метод опорных векторов.
3. Для определения класса хоста в сети предлагается следующая последовательность действий:
   (a) Запросить данные о правилах и статистиках потоков у коммутаторов сети.
   (b) Извлечь информацию о потоках рассматриваемого хоста.
   (c) Вычислить значения HFC, SFE, $ANFP$ и $VNFP$.
4. Получить класс хоста как предсказание метода опорных векторов.

4. Реализация предложенного подхода для контроллера RUNOS 2.0

Реализация предложенного метода была выполнена на языке C++ в виде приложения для контроллера RUNOS 2.0 [10] с открытым исходным кодом. Для маршрутизации пакетов использовалось приложение Learning Switch. Необходимо отметить, что:

1. Во всех сообщениях FlowMod выставлялся флаг OFPFF_SEND_FLOW_REM, сигнализирующий о том, что коммутатору нужно известить контроллер об удалении заданного потока. Это необходимо для корректного вычисления SFE.

2. Также для всех новых потоков выставлялся cookie (идентификатор потока), в котором кодировался хост, получением пакета которого было инициировано создание потока (для отслеживания потоков хоста), а также случайно сгененрированный идентификатор самого потока (необходим для корректного вычисления $ANFP$ и $VMFP$).

Сбор датасета, на котором проводилось обучение предложенного подхода, производится приложением Dataset-Collector [11] для контроллера RUNOS 2.0. Приложение собирает признаки потоков (с помощью OpenFlow запроса flow_stats) хоста $h1$ с интервалом, заданным пользователем, вычисляет HFC, SFE, $ANFP$, $VNFP$ и записывает их в CSV-файл.

Для сбора датасета использовался стенд на основе Mininet с пятью линейно соединенными коммутаторами, поддерживающими протокол OpenFlow 1.3 и 5 хостами, присоединенными к каждому из них (суммарно 25 хостов). Для генерации пользовательского трафика на хосте $h1$ использовалась команда `hping3 -i 1.5 <IP-адрес хоста>` (т.е. генерация одного пакета раз в 1.5 с.), вызванная для каждого из других хостов. Для генерации вредоносного трафика использовалась команда `hping3 -i u6000 -rand-source <IP-адрес хоста>` (т.е. генерация около 167 пакетов/с. с указанием случайного IP-адреса отправителя для каждого пакета).

В качестве модели машинного обучения был выбран метод опорных векторов со стандартным масштабированием признаков. Обучение модели происходило с использованием модуля sklearn, в

частности, были использованы классы StandardScaler и SVC. Для автоматизации обучения моделей был создан скрипт получающий в качестве аргумента командной строки имя CSV-файла, содержащего датасет, а возвращающий файл с данными, необходимыми для вычисления класса хоста по его признакам – коэффициенты масштабирования, а также веса w и b. Для обучения методом проб и ошибок было выбрано значение гиперпараметра C, равное 20.

Для обнаружения DoS-атак на контроллер RUNOS 2.0 было разработано приложение DoS-Detector [12]. После инициализации приложения оно один раз в заданный пользователем промежуток времени посылает коммутатору запросы flow_stats, вычисляет для каждого хоста HFC, SFE, $ANFP$, $VNFP$ (аналогично приложению для сбора датасета) и, пользуясь вычисленными весами признаков, вычисляет класс хоста как $sign(w^T x + b)$. Если класс какого-либо хоста определяется как M (вредоносный) или S (подозрительный), приложение выводит сообщение об этом в консоль. При этом класс хоста определяется как S в случае, если приложение не может отнести хост к классам M или U.

5. Экспериментальное исследование

Для экспериментального исследования предложенного подхода для обнаружения DoS-атак на контроллер ПКС был создан экспериментальный стенд (рисунок 2). Стенд использует Mininet с различными топологиями сети в зависимости от теста (линейная, кольцо и звезда с 5 или 7 коммутаторами, поддерживающими протокол OpenFlow 1.3). К каждому из коммутаторов подключено по нескольку хостов (их количество также варьируется от теста к тесту).

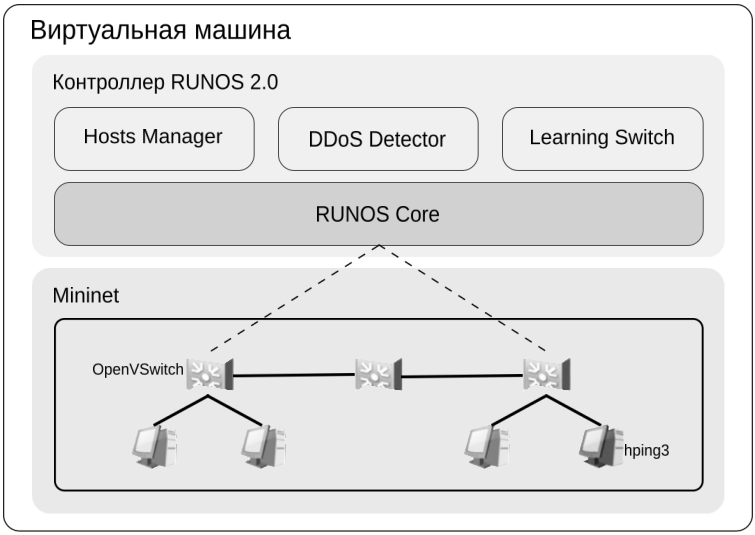

Рисунок 2. Схема экспериментального стенда

Используемая методика такова: каждый из хостов ведет обмен пакетами с другими хостами (всеми остальными в сети) на скорости обычного обмена данными (посылать суммарно около 200 пакетов в секунду). В произвольный момент времени один из хостов начинает посылать вредоносные пакеты с подмененными IP-адресами отправителя при помощи утилиты hping3 (около 300 пакетов в секунду). Ожидается, что приложение-детектор обнаружит изменение его поведения.

Варьируемые параметры:

1. Промежуток между вычислениями значений признаков потоков N (2/3/5 секунд).
2. Количество коммутаторов в сети (5/7).
3. Топология сети (линейная/кольцо/звезда).
4. Количество хостов, присоединенных к каждому коммутатору (3/5/7).

Измеряемые параметры:

1. Количество ложных срабатываний детектора (присвоение класса M или S хосту класса U) за время атаки DoS-атаки на контроллер (около 10 секунд).

2. Количество ложных срабатываний детектора за 10 секунд без атакующих контроллер хостов.

3. Время обнаружения вредоносных хостов (измеряется в промежутках, через которые вычисляются значения признаков потоков, так как иначе на параметр влияет, когда именно была начата атака – сразу после последнего запроса или перед следующим).

Экспериментально полученные результаты, касающиеся времени обнаружения вредоносного хоста, изображены на рисунках 3, 4 и 5. На отметках горизонтальной оси в скобках указано количество коммутаторов в топологии сети. Все измеряемые параметры усреднены по пяти экспериментам. При этом при проведении экспериментов не было обнаружено ошибок 1 и 2 рода. Примеры вывода приложения изображены на рисунках 6 и 7.

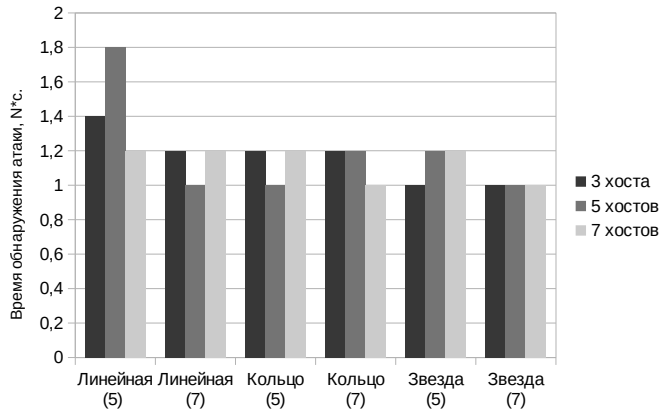

Рисунок 3. Результаты экспериментального исследования при $N =$ 2 сек.

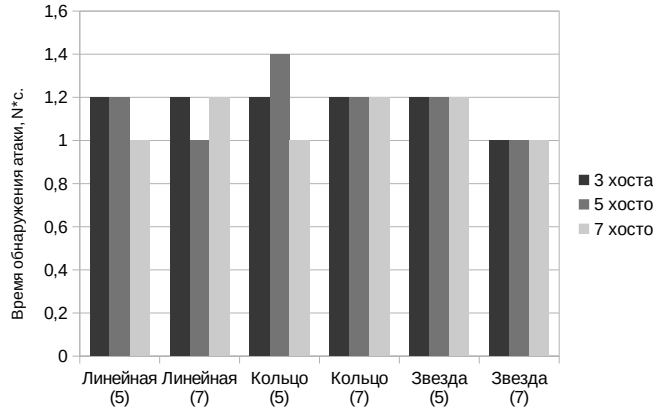

Рисунок 4. Результаты экспериментального исследования при $N =$ 3 сек.

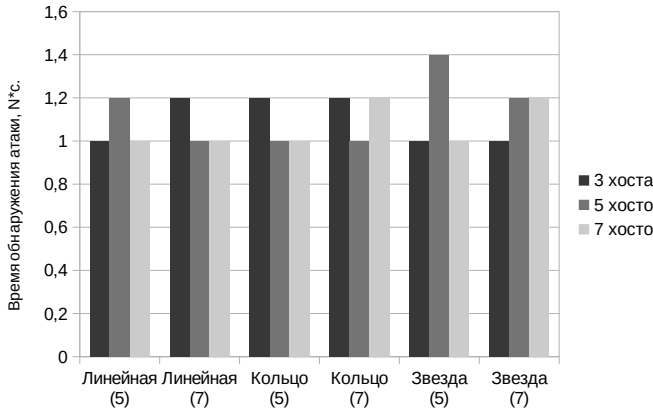

Рисунок 5. Результаты экспериментального исследования при $N =$ 5 сек.

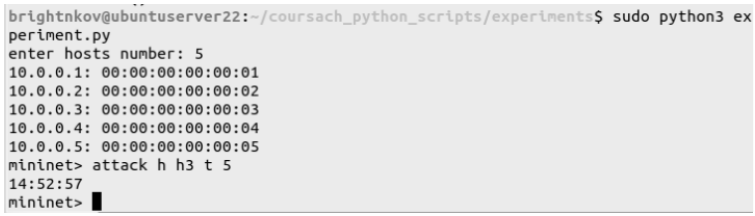

Рисунок 6. Пример создания атаки на контроллер в среде Mininet

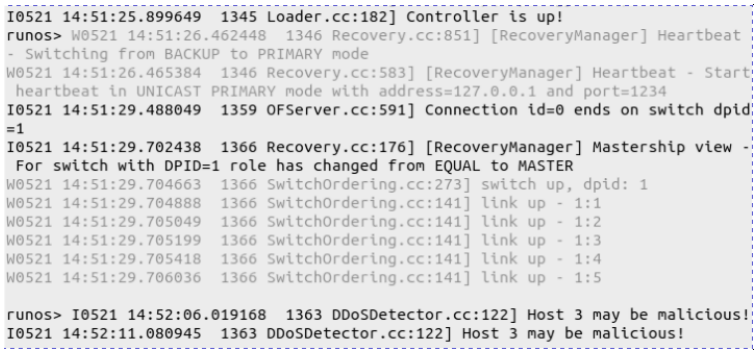

Рисунок 7. Пример сообщения приложения об обнаружении вредоносного хоста

На основе экспериментально полученных данных можно сформулировать следующие выводы:

1. У приложения, обнаруживающего DoS- атаки, не происходит ложных срабатываний, что означает, что в случае использования в реальной сети пользовательский трафик не будет блокироваться.

2. Приложение во всех проведенных экспериментах определяло DoS-атаку и хост, с которого она ведется, что также показывает ее эффективность.

3. Во всех проведенных экспериментах время на обнаружение атаки не превосходило $2 * N$ секунд, где N – количество секунд между вычислениями классов хостов.

Конкретное значение N должно определяться администраторами ПКС – так, при небольшом N нагрузка на контроллер будет больше, однако DoS-атака будет обнаружена раньше, а при больших N нагрузка на контроллер будет значительно меньше, но время обнаружения атаки увеличится.

В рамках дальнейших исследований планируется изучить поведение предложенного метода при менее существенных различиях в скорости отправки трафика вредоносными и пользовательскими хостами.

6. Заключение

В работе был разработан подход для обнаружения DoS-атак на контроллер ПКС на основе метода опорных векторов. Разработанный метод был реализован в виде приложения для контроллера RUNOS 2.0.

В рамках экспериментального исследования подход показал свою эффективность: в процессе работы не происходило ложных срабатываний, а все DoS-атаки и их хосты-источники были распознаны в сети за время, не превосходящее 10 секунд (при $N = 5$ с.).

В рамках развития работы планируется исследовать характеристики предложенного подхода на более сложных топологиях с большим количеством коммутаторов и хостов в сети, исследовать качество работы подхода для обнаружения DDoS-атак, осуществляемых с нескольких зараженных хостов сети, оценить скорость и стабильность.

Литература

1. Смелянский Р. *Программно-конфигурируемые сети* //Открытые системы. СУБД. – 2012. – №. 9. – С. 18-18.

2. Антипина А.В., Пашков В.Н. *Метод предотвращения DDoS атак на контроллер в программно-конфигурируемых сетях*// Программные системы и инструменты. Тематический сборник № 19. Под общей редакцией чл.-корр. РАН, профессора Р. Л. Смелянского. – 2019 – С. 6-17.

3. Pashkov Vasily, and Anna Antipina. *Protection of the Control Plane from DDoS Attacks in Software-Defined Networks* // In Proceedings of International Conference on Modern Network Technologies (MoNeTec). IEEE, 2022.

4. Open Networking Foundation. *OpenFlow Switch Specification. Version 1.3.0* https://www.opennetworking.org/wp-content/

uploads/2014/10/openflow-spec-v1.3.0.pdf (дата обращения: 03.10.2023)

5. Elsayed, M. S., Le-Khac, N. A., Dev, S., & Jurcut, A. D. (2020, August). *Ddosnet: A deep-learning model for detecting network attacks* // In Proceedings of 21st International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM) (pp. 391-396). IEEE, 2020.

6. Kumar Singh, Vishal. *DDOS attack detection and mitigation using statistical and machine learning methods in SDN.* PhD diss., Dublin, National College of Ireland, 2020.

7. Naman Sonthalia, E. V. Avinash Reddy, Harsh Pagaria, G. Vani Jayasri. *Using Machine Learning in Software Defined Networks to Recognize and Avoid DDOS Attacks* // IJRASET. - 2023. - №11. - С. 1145-1150.

8. Myo Myint Oo, Sinchai Kamolphiwong, Thossaporn Kamolphiwong, Sangsuree Vasupongayya. *Advanced Support Vector Machine- (ASVM-) Based Detection for Distributed Denial of Service (DDoS) Attack on Software Defined Networking (SDN)* // Journal of Computer Networks and Communications, 2019.

9. Kumar Singh, Vishal. *DDOS attack detection and mitigation using statistical and machine learning methods in SDN.* PhD diss., Dublin, National College of Ireland, 2020.

10. ARCCN, *SDN/OpenFlow Controller RUNOS 2.0* // URL: https://github.com/ARCCN/runos (дата обращения: 03.10.2023)

11. Британков Егор, *Dataset Collector* // URL: https://github.com/YegorCrazy/dataset_collector_runos (дата обращения: 03.10.2023)

12. Британков Егор, *DDoS Attacks Detector* // URL: https://github.com/YegorCrazy/ddos_detector_runos (дата обращения: 03.10.2023)

Загайнов Д.К., Курячий Г.В., Волканов Д.Ю.

Построение и анализ графа python3-зависимостей в репозитории Sisyphus

Введение

С ростом количества программного обеспечения, написанного на языке программирования **python3** [1], становится все сложнее отслеживать предоставляемые и требуемые программные зависимости. Задача определения и удовлетворения зависимостей актуальна как для разработчиков дистрибутивов, занимающихся в том числе пакетированием программного обеспечения, так и прикладных разработчиков. Так, например, в составе пакета **llvm15.0** [2] 183 различных python3-зависимости.

Также в объемных проектах может быть необходимо разделение набора модулей на поднаборы в соответствии с их зависимостями. Например, из пакетов могут выделять подпакеты с зависимостями на средства тестирования, сборки документации, работы с сетью, чтобы исключить неудовлетворяемые зависимости или снизить набор зависимостей за счет тех, что не влияют на основную, требуемую пользователю, работу пакета.

Представленнная работа имеет следующую структуру:

- В разделе 2 представлено понятие пакета в языке программирования **python3**, его зависимостей, их видов и экспортируемых символов.
- В разделе 3 приведена постановка задачи.
- В разделе 4 приведено описание практической части.
- В разделе 5 приведены результаты экспериментов.
- В разделе 6 приведено заключение по проделанной работе.

1. Описание предметной области

1.1 Понятие модулей в python3

Под модулем в **python3** понимается любой файл с расширением **.py** или тем расширением, которое соответствует разделяемой библиотеке в данной операционной системе.

В первом случае в качестве модуля подразумевается произвольный скрипт, написанный на языке **python3**. Во втором случае в качестве модуля может выступать динамическая библиотека. Так как в данной работе рассматривается репозиторий **Sisyphus**, семейства дистрибутивов **Linux**, **ALT Linux**, то предполагается расширение **.so** для динамических (разделяемых) библиотек.

В свою очередь модули формируют пакеты. Под пакетом в **python3** понимается каталог, в котором могут быть различные модули и другие подпакеты, но обязательно должен быть специальный модуль **__init__.py**.

1.2 Понятие зависимости в python3

Под зависимостью модуля в **python3** понимается его потребность в другом модуле. Иными словами, под зависимостью можно понимать импортирование символа как объекта некоторого пространства имен.

Потребность в символе может быть удовлетворена как самим пакетом, так и модулями, не входящими в пакет.

Сторонняя зависимость — потребность в символе, не экспортируемом самим пакетом.

Если зависимость рассматривать как отношение между модулями, то данное отношение будет обладать свойством транзитивности. То есть, если модуль **A** зависит от модуля **B**, а модуль **B** зависит от модуля **C**, то модуль **A** транзитивно зависит от модуля **C**.

Зависимости можно классифицировать по различным признакам. Например, допустима классификация по типу использования [3]:

- Эксплуатационные зависимости.
- Сборочные зависимости.
- Инструментальные зависимости.

Под эксплуатационной зависимостью понимается набор программ, необходимых при непосредственной эксплуатации ПО, под сборочными программы, необходимые для его сборки, и под инстурментальными те, которые используются при его разработке.

В данной работе будут рассмотрены лишь эксплуатационные зависимости, так сборочные определяются непосредственно в момент сборки ПО.

1.3 Предоставляемые зависимости в python3

python3-пакет может удовлетворять зависимости других пакетов путем предоставления им требуемых символов.

Модуль экспортирует свое имя и свое символьное пространство. Пакет же экспортирует свое имя и предоставляемые символы своего содержимого.

Также существует специальный модуль **__init__**, который импортируется совместно с самим пакетом, добавляя свое пространство имен в пространство имен пакета. Однако, проблема экспортирования внутреннего пространства имен модулей является отдельной задачей и рассмотрена в данной работе не будет.

2. Постановка задачи

2.1 Содержательная постановка задачи

Корневой модуль — модуль, который предоставляет основную функциональность, требуемую пользователем.

Избыточный модуль — модуль, на который существует сторонняя зависимость у одного из модулей исходного пакета, но который не является необходимым.

Тогда для данного **python3**-пакета, заданного набора корневых модулей, входящих в рассматриваемый пакет, и заданного набора избыточных модулей требуется найти:

- Зависимости и экспортируемые символы рассматриваемого **python3**-пакета.
- Ключевой пакет **K**, состоящий из корневых модулей и их зависимостей.
- Набор подпакетов **C**, включающий в себя:
  - Избыточные модули.

– Модули из исходного пакета, зависящие от них.

– Их зависимости.

Требования:

- Модуль не может входить более чем в 1 пакет.
- Если пакет **K** транзитивно или напрямую зависит от одного из избыточных модулей, то разбиение невозможно для заданного набора корневых и избыточных модулей.

Требования к решению:

- Запрещается запускать код компонентов **python3**-пакета.
- Инструмент определения зависимостей python3-пакета должен поддерживать все способы импортирования символов, определенные в п. 2.2

2.2 Формальная постановка задачи

Дано:

$V = F \cup U$, где **V** — вершины, **E** — ребра, **F** — данное множество всех модулей, входящих в пакет.

$G = (V, E)$, **G** — ориентированный граф, без кратных ребер и петель. Данный граф соответствует рассматриваемому пакету и его сторонним зависимостям, его вершины требуемым и предоставляемым модулям, а ребра отношению зависимости.

U — множество сторонних модулей, не входящих в рассматриваемый пакет.

$R \subseteq F$ — заданное множество корневых вершин.

$S = s_1, \ldots, s_\text{k} \subseteq U$ — множество избыточных модулей.

Дуга $(p_1, p_2) : p_1, p_2 \in V$ — зависимость $\mathbf{p_1}$ от $\mathbf{p_2}$

Требуется найти:

- Подграф $K \subseteq G$, который содержит заданные корневые вершины и прямое транзитивное замыкание для них.
- Набор подграфов **C**, каждый из которых содержит:
  - Один из заданных избыточных модулей.

  - Обратное транизитивное замыкание для соответствующего избыточного модуля.
  - Вершины, которые достижимы из набора подграфов $\mathbf{C_k}$, образованных тразнитивным замыканием **F** по зависимостям на $\mathbf{s_k}$.

- Вершины, достижимые более, чем из одного множества $\mathbf{\{C_k\}}$, выделяются в отдельный подграф.

3. Описание практической части

3.1 Инструмент для определения зависимостей

Для определения зависимостей **python3**-модуля был разработан **py3req.py**. Данный инструмент читает **AST**-дерево передаваемого модуля, в случае если это программа, написанная на **python3**. Проходя по этому дереву **py3req.py** находит все импорты, реализованные с помощью функций **import**, **from-import**, **importlib.import_module** и **__import__**.

Так как на данном этапе исследования идет работа лишь с пакетами и их модулями, то при использовании конструкций **from A import B** символ **B** опускается. Таким образом, образуется зависимость **A** вместо **A.B**.

Данный инструмент способен обрабатывать перечисленные ранее функции импортирования модулей, а также относительные импорты. Для исключения из списка зависимостей тех, которые удовлетворяются экспортируемыми символами самого пакета, **py3req.py** использует **py3prov.py**.

В том случае, когда модуль реализован в виде разделяемой библиотеки, то из заголовка **ELF**-файла извлекается архитектура операционный системы и образуется соответствующая зависимость на **ABI python3**. Также данный инструмент позволяет формировать файл **requirements.txt**, используемым в дальнейшем инструментом **pip** для установки зависимостей.

Иногда разработчику необходимо порождать зависимости лишь в каких-то исключительных ситуациях, которые не предполагаются стандартным сценарием, как то вызов функции или обработка исключения. Для таких случаев в **py3req.py** предусмотрен механизм игнорирования подобных импортов, но уведомление пользователя о них.

3.2 Инструмент определения экспортируемых символов

Для определения экспортируемых символов **python3**-пакетов был разработан инструмент **py3prov.py**. Он по именам передаваемых файлов строит иерархию модулей как для абсолютных, так и относительных импортов. Также данный инструмент проверяет возможность импортирования данных символов, например, на наличие недопустимых знаков в имени или доступность из области видимости, в случае использования специальных файлов **.pth**. [4]

4. Разработанный алгоритм выделения подпакета из пакета

1. Формируется подграф $\mathbf{K}$, соответствующий ключевому пакету. Для этого существляется поиск в глубину относительно корневых модулей для выделения набора корневых модулей и их зависимостей. [5]

2. Затем формируются подграфы $\mathbf{C_k}$, состоящие в начале из $\mathbf{s_k}$ избыточных модулей. Для избыточных модулей осуществляется обратный поиск в глубину с добавлением модулей, не вошедших еще ни в один из подграфов $\mathbf{C_k}$ или $\mathbf{K}$, в соответствующие подграфы, содержащих избыточные модули в качестве корня. В случае, если встречается модуль, который уже принадлежит какому-то подграфу, то он и его предки не рассматриваются.

3. Для образовавшихся подграфов $\mathbf{C_k}$, в том порядке, в котором они были переданы на вход, осуществляется поиск в глубину относительно их вершин:

   (a) Если встречается модуль, еще не вошедший ни в один из подграфов, то он добавляется в текущий.

   (b) Если встречается модуль, уже вошедший в один из $\mathbf{C_k}$, то данный модуль выносится в отдельный подграф $\mathbf{C_{k+1}}$ и относительно него осуществляется поиск в глубину для взятия всех его зависимостей, не попавших в $\mathbf{K}$.

5. Экспериментальное исследование

В качестве экспериментальных данных использовались пакеты репозитория **Sisyphus**, содержащие **python3**-модули. Для определения времени работы алгоритма выделения подпакета из пакета использовался модуль **timeit**, входящий в стандартную библиотеку.

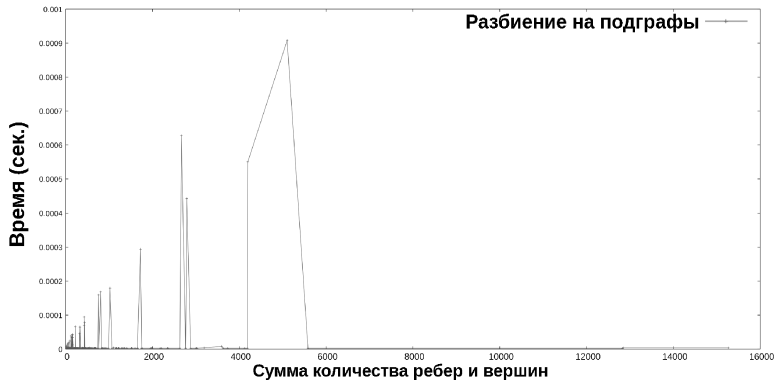

Рис. 1: Зависимость времени работы алгоритма выделения подпакета из пакета от размера пакета

Эксперименты производились на следующем аппаратном комплексе:

- CPU Intel Xeon CPU E3-1246 v3 @ 3.50GHz
- Объем оперативной памяти: 16 Гбайт

В таблице 1 рассмотрим работу инструментов для 3 пакетов с разным количеством модулей и зависимостей, но двумя корневыми вершинами.

Время работы, приведенное в таблице 1, позволяет приблизительно оценить, как меняется время работы алгоритма с увеличением количества вершин и ребер в графе. На рисунке 1 приведен график зависимости времени работы разработанного алгоритма для выделения подпакета из пакета от количества ребер и вершин в получаемом графе. Объем входных данных составил 350 пакетов. В качестве избыточных модулей брались модули **requests** и **numpy**, но для каждого пакета задавался всего 1 избыточный модуль.

Название пакета	Кол-во python3-модулей	Кол-во вершин	Кол-во ребер	Кол-во избыточных вершин	Время работы
reno	29	59	101	1	0.17 сек
matplotlib	157	294	1120	3	1.81 сек
silx	617	816	3479	2	3.57 сек

Таблица 1: Время работы алгоритма выделения подпакетов из пакета.

По рисунку 1 видно, что длительность выполнения алгоритма не превышает 0.001 с. Это позволяет запускать алгоритм сотни раз без создания существенных задержек. Всплески на графике частично могут быть объяснены вызовом и обработкой исключений со стороны программ. Исключения вызывались в случае попадания в цикл или зависимости одного из корневых модулей на избыточный.

6. Заключение

В данной работе были решены следующие задачи:

1. Разработан инструмент для определения зависимостей **python3**-модулей.
2. Разработан инструмент для определения экспортируемых символов **python3**-пакетов.
3. Разработан алгоритм и реализован инструмент для выделения из **python3**-пакета подпакетов с зависимостями на избыточные модули.
4. С помощью разработанных инструментов удалось проанализировть зависимости и экспортируемые символы пакетов репозитория **Sisyphus**, а затем построить и проанализировать на предмет возможности выделения избыточных зависимостей полученные графы **python3**-зависимостей.

В ходе проведенных исследований выяснилось, что, несмотря на актуальность описанной проблемы, пока еще не было предложено инструмента для определения всех экспортируемых

python3-модулями символов. Данная подзадача требует дальнейшего изучения.

Литература

1. Python3 programming language web-site. URL:`https://www.python.org/` (дата обращения 24.09.2023)
2. The LLVM Compiler Infrastracture. URL:`https://llvm.org/` (дата обращения 23.09.2023)
3. Курячий Г. В. *Зависимости при разработке на Python* URL: `https://uneex.org/LecturesCMC/PythonDevelopment2023/04_MergetoolCommandline` (дата обращения 21.09.2023)
4. Documentation for python3 module "site". URL: `https://docs.python.org/3/library/site.html` (дата обращения 17.09.2023)
5. Depth-first search algorithm. URL:`https://en.wikipedia.org/wiki/Depth-first_search` (дата обращения 10.10.2023)

Писковский В.О., Сагалевич В.Д.

Исследование зависимости характеристик работы виртуальной машины в режиме расширенного фильтра записи от способа хранения и протокола доступа к данным виртуального диска

Введение

Практическая деятельность обработки информации на персональном компьютере обычно требует предоставления возможности одновременной работы с внутренними, распределенными информационными ресурсами, и доступом в сеть Интернет. Необходимость решения этой задачи обуславливает использование в ряде случаев соответствующих организационно-технических методов защиты информации, в основе которых лежит идея изоляции доменов. Программное средство общего назначения с встроенными средствами защиты "Абонентский облачный терминал" (АОТ)[1] решает задачу путем совмещения на одной аппаратной единице, персональном компьютере, произвольного количества разнородных, изолированных, не зависимых друг от друга рабочих мест. Каждое место работы функционирует под контролем операционной системы, в своей программной и сетевой среде, что позволяет полностью изолировать его от сети.

Рабочая инфраструктура централизованной [облачной] вычислительной структуры (ЦВС)[1] предприятия может содержать репозитарий контейнеров с виртуальными машинами (ВМ), реализующими рабочие места сотрудников и прошедшими принятую на предприятии процедуру внедрения в постоянную эксплуатацию. Репозитарий эталонных ВМ и обновлений содержит эталонные ВМ, серверы с обновлениями ПО. Задачи, решаемые этим компонентом ЦВС: формирование проверенных, согласованных, безопасных эталонных рабочих мест в виде контейнеров с типовыми ВМ. Образы ВМ копируются на локальные диски АОТ. Однако одним из существенных недостатков такого решения является то, что для размещения на персональном компьютере образа виртуальной машины с

установленной на ней полноценной ОС, например, MS Windows 10, необходимо скопировать и передать по сети около 40 Гб, что выливается в длительную процедуру обновления и приводит к ”коллапсу” локальной сети.

Чтобы избежать такой ситуации, предлагается запускать ВМ в режиме расширенного фильтра записи. В этом режиме можно настроить образ ВМ так, чтобы он оставался доступным только для чтения, а все изменения, внесенные пользователем, сохранялись локально на специальном слое файловой системы. Решение поставленной задачи должно снизить время загрузки образа ВМ и нагрузку на сеть.

1. Постановка задачи

Целью работы было исследование времени запуска ВМ в режиме расширенного фильтра записи на АОТ в зависимости от способа хранения и протокола доступа к данным виртуального диска.

Введем обозначения:

- w - пропускная способность сети, Мбит/сек;
- $\tau 0$ - задержка отправления пакетов в сети, мс;
- Ncl – количество клиентов;
- Mos – тип ОС (Windows, Linux);
- Cs – критерий старта;
- P – протокол доступа к данным виртуального диска, P $\in \{SMB, NBD, NFS\}$.

Требуется:

- Экспериментальным методом найти зависимость времени запуска T=f_t(w, $\tau 0$, Ncl, Mos, Cs, P) и максимальную скорость чтения данных виртуального диска Rx=f_{Rx}(w, $\tau 0$, Ncl, Mos, Cs, P).

2. Обзор существующих протоколов доступа к данным

Существует множество различных протоколов доступа к данным виртуального диска. В статье приведен краткий обзор и сравнительный анализ применимости протоколов для решения задачи запуска образа ВМ на АОТ в режиме расширенного фильтра записи со следующими критериями:

1. **Оптимизирующие механизмы**: наличие у протокола механизмов, снижающих нагрузку на сеть, то есть возможность уменьшить количество сообщений между клиентом и сервером.

2. **Масштабируемость**: поддерживает ли протокол горизонтальное масштабирование.

3. **Чувствительность**: снижается ли производительность протокола при изменении задержки отправки пакетов в сети.

4. **Устойчивость**: реакция протокола на разрыв соединения между клиентом и сервером.

Используемые метрики - Пропускная способность сети.

SMB

- **Описание протокола и область применения**

  SMB (Server Message Block)[2] — сетевой протокол, обеспечивающий общий доступ к файлам и устройствам в компьютерной сети, будь то внешний жесткий диск или принтер. Этот протокол, основанный на клиент-серверной технологии является довольно популярным инструментом для удалённого доступа к данным. Клиент и сервер взаимодействуют с помощью TCP/IP.

- **Масштабируемость**

  Протокол был разработан с упором на масштабируемость. Он может работать с большим количеством клиентов и высокой интенсивностью сетевого трафика. Однако сущетсвуют ограничения при одновременной работе множества клиентов с одним файлом. В таком случае SMB может заблокировать файл для предотвращения конфликта. Это может к дополнительной нагрузке на сеть.

- **Оптимизирующие механизмы**

  В протоколе реализовано множество оптимизирующих механизмов, позволяющих уменьшить количество пакетов в сети и снизить общую нагрузку на нее. Например, оппортунистическая блокировка, позволяет кэшировать некоторое количество пакетов и объединять их в один. Также улучшения производительности удается достичь благодаря переносу операций передачи данных на сетевую карту, тем самым разгружая процессор.

- **Чувствительность**

  Эффективность протокола снижается с увеличением задержки, так как время ожидания подтверждения доставки пакета также увеличивается.

- **Устойчивость**

  В случае разрыва соединения между клиентом и сервером, протокол позволяет восстановить прерванную сессию.

NBD

- **Описание протокола и область применения**

  NBD (Network Block Device)[3] - это сетевой протокол, который позволяет клиенту получать доступ к удаленным блочным устройствам по сети. Работает по модели клиент-сервер, где клиент и сервер взаимодействуют с помощью TCP/IP.

- **Масштабируемость**

  Протокол обладает хорошей способностью масштабирования. Масштабирование можно осуществлять горизонтально, добавляя дополнительные серверы для равномерного распределения нагрузки. Также доступен вариант использования балансировщика нагрузки для равномерного распределения клиентов между серверами. Кроме того, протокол позволяет кэшировать на стороне сервера данные, которые часто запрашиваются клиентами.

- **Оптимизирующие механизмы**

  Нагрузка на сеть во многом зависит от установленных настроек безопасности. У протокола есть проблемы с устойчивостью, поэтому при большом количестве соединений,

их разрывы могут значительно нагрузить сеть. Кроме того, некоторая дополнительная нагрузка из-за отсутствия возможности пропуска участков файлов, заполненных нулями. Однако он обладает механизмами кэширования, предварительной загрузки данных и параллельной обработки нескольких запросов клиента.

- **Чувствительность**

  Механизмы кэширования позволяют снизить влияние задержки на эффективность работы протокола.

- **Устойчивость**

  Протокол не имеет встроенных механизмов отказоустойчивости. Общение между клиентом и сервером может закончиться только по желанию клиента, то есть сервер не может самостоятельно оборвать соединение. Если происходит некорректный разрыв соединения с NBD устройством, необходимо выполнить полную перезагрузку сервера для сброса текущей сессии, иначе клиент не сможет повторно установить соединение с сервером. Поэтому если разрыв соединения происходит во время передачи данных, то при повторном подключении процесс передачи начнется с самого начала.

NFSv4

- **Описание протокола и область применения**

  NFSv4 (Network File System version 4)[4] — протокол, созданный для работы с распределенной файловой системой. Позволяет клиенту получать доступ к удаленной файловой системе (экспортированным каталогам) по сети. Основан на технологии клиент-сервер. Использует TCP/IP.

- **Масштабируемость**

  Протокол сконструирован с упором на быструю передачу данных и хорошую масштабируемость при параллельной работе нескольких клиентов с одним файлом. Когда несколько клиентов читают данные пропускная способность делится между ними не равномерно. Протокол поддерживает горизонтальное расширение, то есть увеличение количества серверов с данными для равномерного распределения нагрузки. Кроме того, этот протокол включает в себя расширение pNFS[5], специально разработанное для обеспечения параллельной работы множества клиентов.

- **Оптимизирующие механизмы**

  Нагрузка на сеть зависит от его конфигураций и числа клиентов, использующих его. Протокол включает в себя механизмы, направленные на снижение нагрузки на сеть, такие как возможность отправлять множество запросов к серверу одновременно и предварительный выбор данных для загрузки. Однако для эффективной работы этих механизмов необходима высокая пропускная способность сети.

- **Чувствительность**

  Увеличение задержки в сети сильно сказывается на эффективности работы протокола.

- **Устойчивость**

  В случае существования активной сессии клиента до перезагрузки сервера, существует возможность восстановления этой сессии после перезагрузки.

3. Программа и методика исследования

3.1 Программа исследования

- Измерить время запуска ВМ, критерием которого является загрузка сетевого интерфейса и графической оболочки, при использовании протоколов:
  - SMB3
  - NFSv4
  - NBD
- Измерить скорость чтения данных виртуального диска при условии доступа к нему с помощью протоколов:
  - SMB3
  - NFSv4
  - NBD

3.2 Методика исследования

Схема стенда для исследования

Стенд состоит из сервера и автоматизированного рабочего места (АРМ) сотрудника, которые соединены через маршрутизатор с помощью Gigabit Ethernet. В качестве сервера используется АОТ, построенный на базе ОС Debian 8.11 Jessie, со специальным ПО для администрирования АРМ сотрудника (Рис.1). Сервер обладает следующими характеристиками:

- SSD диск SanDisk SDSSDHP128G, скорость чтения 530 МБ/сек.
- 4-ядерный процессор Intel Core i5-4570 CPU 3.20GHz, L3: 6 МБ кэша.
- Сетевая карта Intel Ethernet Connection I217-V, 256 КБ кэша.

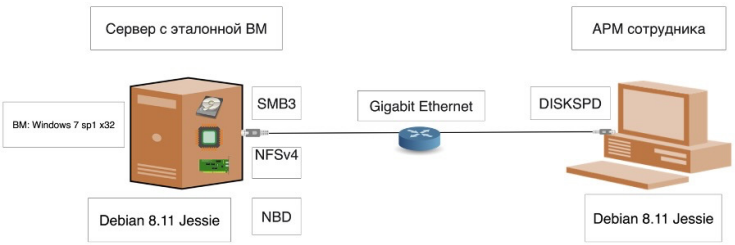

Рис. 1: Схема стенда

На виртуальный диск предварительно установлен образ ВМ Windows 7 sp1 x32, размером 15 ГБ. Образ содержит приложение Diskspd[6] вместе со скриптами для его запуска. Образ хранится в формате Raw с целью повышения эффективности[7]. Также на сервере настроены протоколы и соответствующие настройки доступа к файлу с ВМ. АРМ сотрудника сконфигурирован также на АОТ со специальным ПО для запуска ВМ и настроенной клиентской частью выбранных протоколов. Характеристики клиента аналогичны характеристикам сервера.

Алгоритм тестирования

- На клиенте и сервере настраивается выбранный протокол:
  - SMB3
  - NFSv4
  - NBD
- На сервер установлена утилита Linux Traffic Control. С её помощью настраивается задержка и скорость в сети. Задержка 0 означает, что дополнительная задержка не вносится, и она пренебрежимо мала по сравнению с остальными измеримыми характеристиками, как, например, если образ ВМ находится на самом локальном диске абонентского терминала и работает в режиме расширенного фильтра записи с целью обеспечения безопасности.
- Клиент подключается к серверу и производит автоматизированный запуск ВМ.
- После запуска ВМ, с помощью планировщика задач, запускаются тесты производительности.
- Тесты проходят в режимах:
  - последовательного чтения данных блоками по 1 МБ
  - случайного чтения данных по случайно выбираемым смещениям блоками по 4 КБ
  - чтения данных по случайно выбираемым смещениям блоками по 4 КБ с одновременным выполнением 32 операций чтения

3.3 Результаты экспериментов

В процессе экспериментов для каждого типа тестов получены данные, позволяющие определить целесообразность применения протокола в зависимости от параметров сети. Основными показателями, которые учитывались при анализе результатов эксперимента было время загрузки ВМ и скорость чтения данных с диска при разных значениях задержки и скорости в сети.

- Задержка - задержка в сети.
- Сеть - скорость в сети.

- Протокол - тестируемый протокол.
- T - Время загрузки - время загрузки ВМ.
- Rx - Скорость чтения - скорость чтения данных с диска.

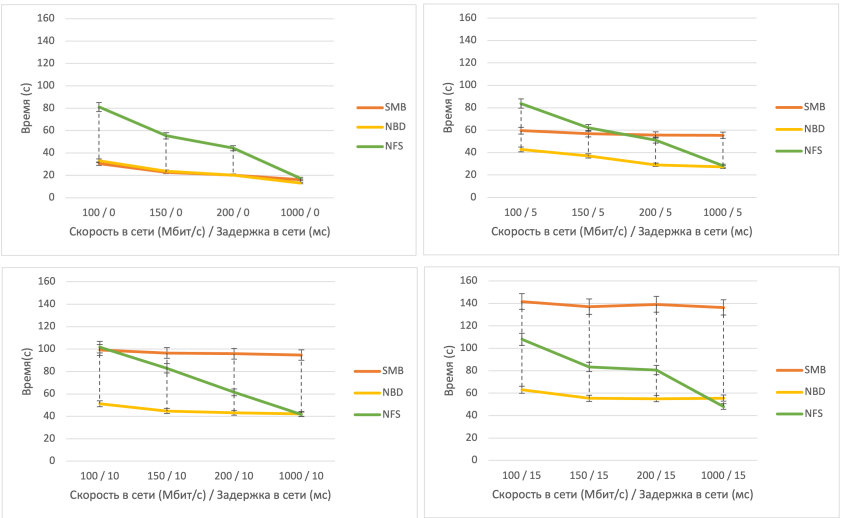

Рис. 2: Графики зависимости времени загрузки ВМ от скорости и задержки в сети

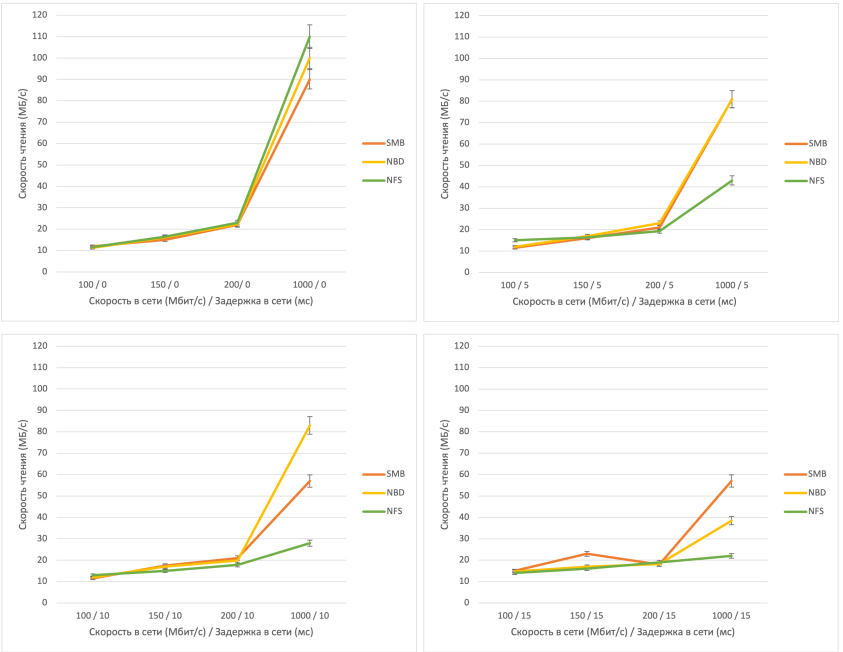

Рис. 3: Графики зависимости скорости чтения от скорости и задержки в сети при тестировании в режиме случайного чтения

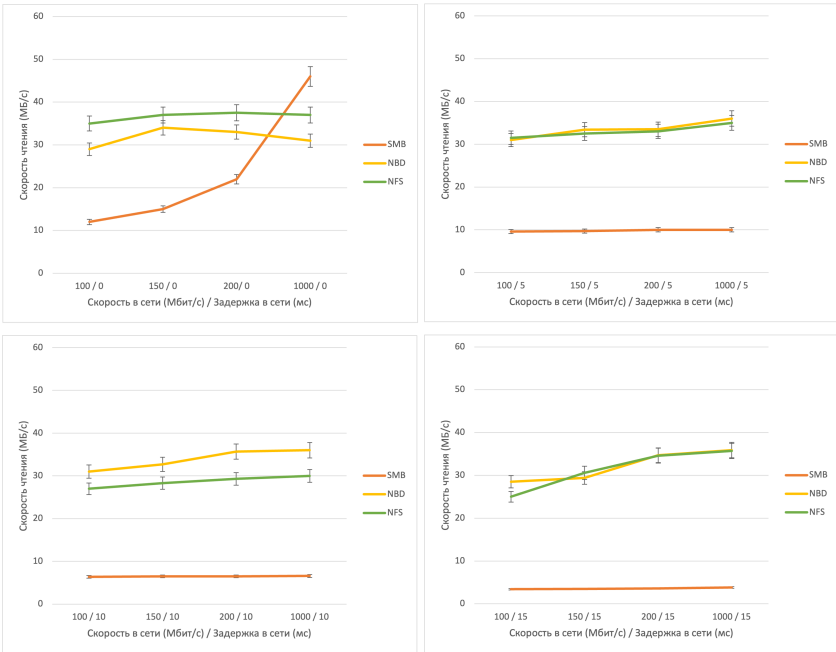

Рис. 4: Графики зависимости скорости чтения от скорости и задержки в сети при тестировании в режиме последовательного чтения

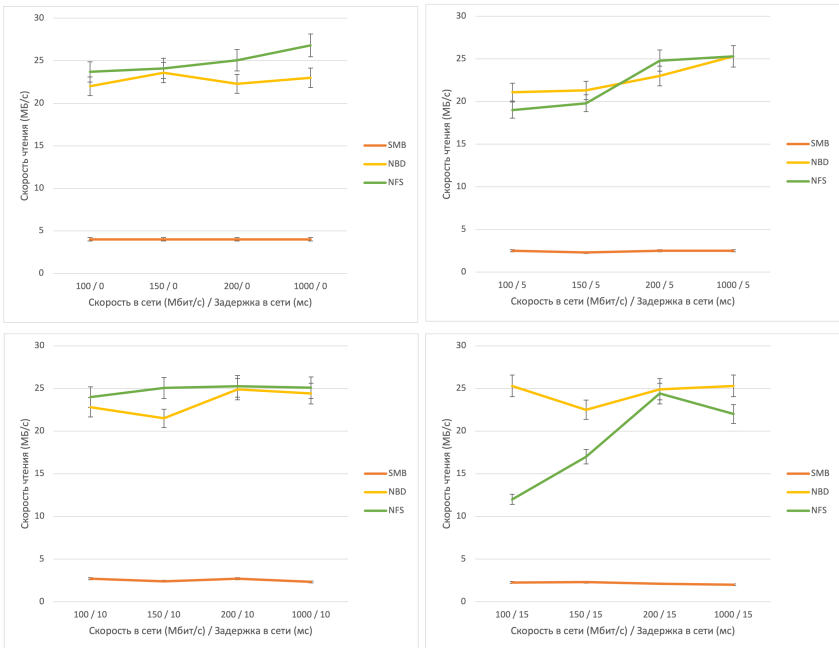

Рис. 5: Графики зависимости скорости чтения от скорости и задержки в сети при тестировании в режиме случайного чтения с одновременным выполнением 32 операций чтения

По результатам экспериментов протокол NBD в среднем показал наилучшие результаты. Время загрузки (Рис. 2) ВМ при его использовании в большинстве случаев оказывается наименьшим. Это связано с тем, что в протоколе NBD реализованы эффективные механизмы кэширования данных, которые позволяют уменьшить количество запросов и увеличить скорость доступа к данным.

Протокол SMB показал хорошие результаты во время измерения времени загрузки ВМ при условии маленькой задержки. В других случаях время загрузки больше, чем у NFS и NBD, так как протокол чувствителен к изменению задержки в сети. Также его производительность в тестах скорости чтения (Рис. 3, 4, 5) оказалась очень низкой. NFS в среднем показал результаты лучше, чем протокол SMB. Время загрузки ВМ при низкой задержке гораздо больше, чем у NBD и SMB. При увеличении задержки время загрузки ВМ с использованием NBD становится меньше, чем при использовании SMB.

4. Заключение

Оптимальным выбором для работы с ВМ в режиме расширенного фильтра записи является протокол NBD. Благодаря реализованным оптимизирующим механизмам протокол эффективен при разных значениях скорости и задержек в сети. В случае низкой задержки в сети лучше выбирать протокол SMB, так как он позволит сократить время загрузки ВМ.

Необходимо отметить, что влияние задержки на эффективность работы протоколов будет подробнее рассмотрено в новом исследовании.

Литература

1. Грушо А.А., Николаев А.В., Писковский В.О., Сенчило В.В., Тимонина Е.Е. *Подход к обеспечению информационной безопасности при использовании частных облачных вычислительных сред. Абонентский облачный терминал.*, MoNeTec-2020. – 2020

2. *[MS-SMB2]: Server Message Block (SMB) Protocol Versions 2 and 3*, Microsoft, 2020, URL: `https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-smb2/5606ad47-5ee0-437a-817e-70c366052962`, дата обращения: 2.11.2022

3. *The NBD protocol description*, URL: `github.com/NetworkBlockDevice/nbd`, дата обращения: 25.02.2023

4. *RFC 7530*, Internet Engineering Task Force (IETF), URL: `www.rfc-editor.org/rfc/rfc7530.html`, дата обращения: 2.10.2022

5. *An Overview of NFSv4*, Storage Networking Industry Association, 2012, URL: `https://www.snia.org/sites/default/files/SNIA_An_Overview_of_NFSv4-3_0.pdf`, дата обращения: 26.10.2022

6. *Diskspd*, Microsoft, URL: `github.com/microsoft/diskspd`, дата обращения: 25.02.2023

7. Minho Lee, Young Ik Eom *Efficient Data Cluster Management Scheme for Qcow2-based Virtual Disk in Home Cloud Server.*, IEEE International Conference on Consumer Electronics (ICCE), 2018.

Писковский В.О., Шибаев П.П.

Блокчейн–хранилище для контроллера ПКС Runos: концепция и архитектура

Введение

Программно-конфигурируемые сети (ПКС, Software Defined Networks — SDN) — это быстро развивающаяся технология, которая в последние годы получила широкое распространение благодаря своей способности повышать гибкость и эффективность сети. Одним из ключевых компонентов конфигурации SDN является сетевой контроллер, который отвечает за управление и направление трафика внутри сети. Многие приложения, созданные для контроллеров SDN, полагаются на централизованные базы данных или распределенные хранилища данных для хранения данных. Однако использование технологии блокчейн в качестве механизма хранения для приложений SDN ещё так широко не изучалось.

В этой статье мы предлагаем исследовать возможность использования базы данных блокчейна в качестве механизма хранения для SDN-приложения под названием L2 learning switch, которое предназначено для использования с сетевым контроллером RUNOS [1]. Цель исследования — продемонстрировать надежность и масштабируемость использования базы данных блокчейна для хранения данных приложения поддержки работы сетевых ресурсов в контуре данных.

Работа организована следующим образом. Даётся краткий обзор предметной области, соответствующей работы в области SDN и технологии блокчейн. Предлагается описание экспериментального стенда и методики испытаний. Представлены результаты экспериментальных исследований. Описываются возможные дальнейшие шаги исследований.

1. Постановка задачи

1.1 Контекст задачи

Использование ПКС приобрело популярность в последние годы как средство повышения гибкости и эффективности сети. Одним из ключевых компонентов настройки SDN является сетевой контроллер, который отвечает за управление и направление трафика.

RUNOS — это современный распределенный контроллер SDN/OpenFlow с открытым исходным кодом, который предоставляет множество преимуществ в управлении коммутаторами и потоками в программно-конфигурируемых сетях. Он предоставляет открытый API, который позволяет сторонним разработчикам создавать приложения для расширения его функциональности. RUNOS также унифицирует код и интерфейсы, обеспечивая поддержку различных подсистем в соответствии с принятыми соглашениями и требованиями. Его динамическое подключение приложений (PnP, Plug and Play) облегчает внедрение и управление различными функциональными компонентами. Кроме того, RUNOS позволяет строить логические распределенные отказоустойчивые центры управления, обеспечивая взаимозаменяемость, отказоустойчивость, ремонтопригодность и снижение издержек на поддержку, замену и модификацию сетевых решений.

Сегодня многие приложения, создаваемые для SDN-контроллеров полагаются либо на централизованные базы данных, либо на резидентные (”in-memory”) хранилища, например, Redis [2]. Основным недостатками архитектуры на основе Redis являются наличие единой точки отказа и отсутствие специальных механизмов защиты от подмены данных в хранилище. Альтернативным вариантом представляется использование блокчейна. В этой работе предлагается показать, как можно интегрировать приложение для сетевого контроллера RUNOS L2-learning switch для поддержки работы сетевых ресурсов в контуре данных с блокчейн-хранилищем для хранения данных приложения. Решения по типу Mongo DB и Dqlite cluster обладают теми же недостатками, что и Redis в случае сравнения с блокчейн-хранилищами.

1.2 О способах децентрализованного хранения данных

Существуют два основных типа архитектур информационных систем: централизованные и распределенные. Каждый из них обладает уникальными характеристиками и преимуществами, которые определяют их применение в различных областях. Централизованные системы характеризуются высокой скоростью работы благодаря единой точке контроля. Они обеспечивают доступность базы данных и полный контроль над данными. Однако их масштабируемость ограничена, и для увеличения производительности часто требуется обновление центрального узла, а также дополнительные лицензии и поддержка. С другой стороны, распределенные системы обладают масштабируемостью и надежностью благодаря встроенным механизмам резервирования и доступности, гарантируя безопасность и отказоустойчивость при доступности не менее 2/3 своих ресурсов.

Технология распределенного реестра (сокращенно — РР, или Distributed Ledger Technology — DLT) — это совокупность концепций и технологий, которые позволяют создавать и управлять базой данных, которая распределена между несколькими компьютерами или узлами в сети. Посредством общего механизма консенсуса обеспечивается целостность и защищенность регистрации и хранения транзакций (формализуется через т.н. ”проблему византийских генералов”) [3].

Блокчейн — разновидность DLT, включающая цепочку криптографически связанных, хронологически упорядоченных блоков, содержащих транзакции.

Блокчейн-сеть — одноранговая сеть связанных между собой вычислительных устройств, на которых присутствует экземпляр программного обеспечения блокчейна. Такие вычислительные устройства зачастую называют узлами или нодами (от англиского node — узел) сети.

Permissioned (открытые, не требующие прав доступа, от англ. permission — ”разрешение”) блокчейны предоставляют контролируемый доступ в свою сеть, что идеально подходит для корпоративных сред, где конфиденциальность и безопасность данных являются приоритетом. Они могут использоваться, например, в авиакомпаниях, финансовых учреждениях и торговых площадках, где важен строгий контроль доступа.

Permissionless (закрытые, требующие прав доступа) блокчейны, такие как Ethereum, Bitcoin, Cosmos Hub [4] и другие, предоставляют открытый и универсальный подход, не ограничивая

доступ к участникам. Они подходят для сферы, где необходима максимальная универсальность и доступность, например, при работе платформ, предоставляющих услуги создания и функционирования различных распределенных реестров.

Однако Permissionless блокчейны могут столкнуться с проблемами с масштабируемостью. Сложность в развертывании таких систем также выше из-за сложности коммуникаций между участниками.

Выбор между этими двумя типами блокчейнов зависит от конкретных требований системы и помогает адаптировать блокчейн-решения под потребности сетевой инфраструктуры.

Распределенные реестры (блокчейны) можно разделить на несколько уровней, каждый из которых выполняет определенные функции и обеспечивает характеристики системы.

Слой L0 (взаимодействие L1-блокчейнов) обеспечивает взаимодействие между различными блокчейнами L1, обеспечивая их согласованную работу. В качестве примеров можно привести Cosmos и Polkadot. Слой L1 (поддержка транзакций) — основной слой, который упорядочивает (сериализует) записи, образует из них транзакции, записываемые в блокчейн, например, Bitcoin и Ethereum. Слой L2 (масштабирование) решает проблемы масштабируемости блокчейнов L1 и интегрирует их для обеспечения более высокой производительности, как то: Polygon и Optimism (L2 Ethereum-совместимые решения для удешевления и ускорения транзакций). Слой L3 (поддержка приложений) содержит децентрализованные приложения и протоколы, позволяя разработчикам создавать разнообразные блокчейн-приложения, например, Uniswap реализованная в виде смарт-контракта на блокчейне Ethereum.

2. Архитектура решения

В данной работе представлен опыт интеграции предлагаемого решения для хранения данных на основе блокчейна для RUNOS с приложением для Runos. Блокчейн создан с помощью Cosmos SDK [4] — библиотеки для разработки специализированных блокчейнов. Сетевое взаимодействие, обмен транзакциями и безопасность осуществляются с помощью платформы на основе алгоритма консенсуса Tendermint Core [5]. Архитектура экспериментального стенда представлена на Рис. 1. Возможности Cosmos SDK и Tendermint были исследованы на реальной инфраструктуре рабочей группой Interchain Foundation [6]. Из этого исследования

следует, что:

- нагрузка для 128 валидаторов (узел-валидатор или просто валидатор — это узел блокчейна, отвечающий за проверку и подтверждение транзакций) составила по 12 клиентов на валидатор (1536 клиентов);
- регистрация блока происходила в среднем за $2,53$ с (96% попадает в промежуток 2.3 с – $2,7$ с);
- пропускная способность составляет порядка 400 тр/с;
- отказоустойчивость гарантируется при отключении 1/3 валидаторов, при этом время создания блоков на работающих валидаторах не изменяется.

В рамках данного исследования блокчейн-приложение [7] развернуто в 4 экземплярах на отдельных машинах. Также присутствует экземпляр сетевого контроллера RUNOS, на котором запущено приложение L2 Learning Switch, написанное на языке C++. Для того, чтобы обеспечить совместимость приложений, на языке Go написан REST API сервер [7], к которому обращается L2 Learning Switch. В свою очередь, сервер направляет запросы к gRPC-узлу блокчейна. В блокчейне хранятся кортежи вида $(dp\_id, eth\_addr, port)$. Произведены тестовые запуски, которые продемонстрировали совместимость и корректную работу компонентов стенда.

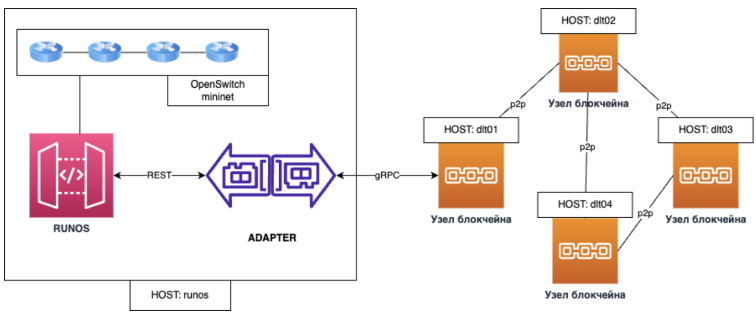

Рисунок 1. Архитектура экспериментального стенда

3. Экспериментальное исследование

Результаты проведенных в рамках данной работы экспериментов показали, что экспериментальный стенд для хранения данных на основе блокчейна уже в случае 3 и 4 узлов уступает в производительности БД Redis. С увеличением числа узлов происходит увеличение среднего времени выполнения операций (Рис. 2). Доверительные интервалы представлены в Табл. 1. Это можно объяснить накладными расходами для достижения консенсуса между узлами блокчейна и сложностью структуры данных, которую он поддерживает.

Случай	**Чтение (мс)**	**Запись (мс)**
Redis	$0,0738 \pm 0,00011$	$0,07965 \pm 0,00014$
Блокчейн, 3 узла	$0,135 \pm 0,00012$	$0,1891 \pm 0,0002$
Блокчейн, 4 узла	$0,1486 \pm 0,0002$	$0,22665 \pm 0.0003$

Таблица 1: Время выполнения операций, доверительные интервалы

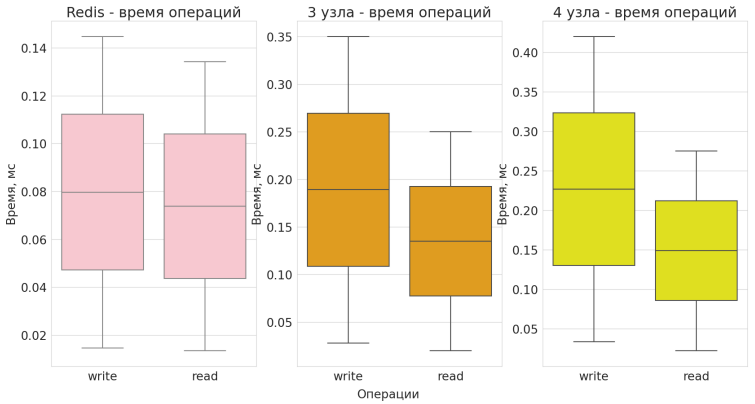

Рисунок 2. Результаты экспериментов

4. Заключение

В данной работе проведено сравнение производительности решения для хранения данных на блокчейне и традиционной NoSQL базы данных Redis. Результаты показали, что с увеличением числа узлов возникают накладные расходы на достижение консенсуса между ними.

Исходя из проведенных исследований, выбор между централизованными и распределенными системами хранения данных, а также между различными типами блокчейнов, следует делать в зависимости от конкретных требований и потребностей в сетевой инфраструктуре. Дальнейшие шаги исследований могут предполагать проведение экспериментальных исследований по нагрузке блокчейн-сети, а также создание статистической модели для подбора оптимальных параметров распределенного реестра для хранения данных в блокчейне, а также алгоритма консенсуса Tendermint.

Литература

1. Официальный сайт проекта Runos `https://runsdn.com/index.php/products/runos`

2. Пашков В.Н. Разработка высокодоступной платформы управления для программно-конфигурируемых сетей // Материалы XIX Международной конференции по вычислительной механике и современным прикладным программным системам (ВМСППС'2015) (Алушта, 2015). — М.: Издательство МАИ, 2015. — С. 169 – 171

3. Sunyaev A. Distributed Ledger Technology // Internet Computing. — 2020. — P. 265–299.

4. Официальный сайт проекта Cosmos SDK `https://docs.cosmos.network/`

5. Kwon J. Tendermint: Consensus without Mining `https://tendermint.com/static/docs/tendermint.pdf`

6. Cason, D., Fynn, E., Milosevic, N., Milosevic, Z., Buchman, E., Pedone, F. The design, architecture and performance of the Tendermint Blockchain Network // 40th International

Symposium on Reliable Distributed Systems (SRDS) — 2021. — P. 23–33.

7. Исходные коды приложений

   `https://github.com/shibaeff/runos_chain`

Рязанов А. М., Волканов Д. Ю., Цыганов Н.И.

Методы сбора и хранения сетевого трафика для решения задачи прогнозирования качества гетерогенных каналов в сетях передачи данных

Введение

Качество обслуживания (КО) - технология предоставления различным классам трафика различных приоритетов в обслуживании [1]. В работе рассматриваются следующие 4 показателя КО:

- пропускная способность (bandwidth);
- задержка при передаче пакета (latency);
- джиттер (jitter);
- доля потерь пакетов (loss ratio).

В этой статье рассматривается спектр подзадач, сопряжённых с формирование обучающих наборов, которые могут быть использованы для решения задач анализа и прогнозирования КО гетерогенных каналов в СПД.

В разделе 1 этой статьи приводится постановка задачи прогнозирования качества гетерогенных каналов в СПД. В разделе 2 описываются параметры КО для гетерогенного канала, в разделе 3 приводится выбор методов сбора и хранения сетевого трафика. Раздел 4 содержит описание архитектуры средств сбора, хранения и анализа сетевого трафика.

1. Постановка задачи прогнозирования качества гетерогенных каналов в сетях передачи данных

Неформально задачу прогнозирования качества гетерогенных каналов в СПД можно сформулировать следующим образом:

Дано:

- набор характеристик исследуемого гетерогенного канала:
    1. максимальная пропускная способность;
    2. максимальный процент потерянных пакетов;
    3. номинальный диапазон колебания задержек.
- трасса сетевого трафика, собранная на этом канале на временном промежутке $[0; T]$, содержащая информацию о заголовках протокольных единиц данных от канального (L2) до транспортного (L4) уровня. Также может содержать информацию о заголовках прикладного уровня (L7) для упрощения идентификации трафика и извлечения сведений о характеристиках КО.
- набор показателей КО:
    - пропускная способность (bandwidth);
    - задержка при передаче пакета (latency);
    - джиттер (jitter);
    - доля потерь пакетов (loss ratio).

Необходимо:

- определить значения показателей КО на промежутке $[0; T]$;
- спрогнозировать показатели КО на промежутке времени $(T; T^*]$.

Данная задача разбивается на следующие подзадачи:

1. разработка методов подсчёта числовых значений показателей КО гетерогенного канала;
2. разработка методов сбора сетевого трафика;
3. разработка методов хранения сетевого трафика;
4. разработка методов прогнозирования показателей КО гетерогенного канала.

В данной статье рассматриваются подзадачи 1,2,3.

2. Параметры КО для гетерогенного канала

Рассмотрим 4 показателя КО подробней и рассмотрим способы их подсчёта [2].

Пропускная способность – Bandwidth

Пропускная способность (bandwidth) – это количественная характеристика, отражающая возможности передачи данных по конкретному средству подключения. В СПД под пропускной способностью понимается объем данных, который можно передать из одной точки в другую за определенное время.

Задержка – Type-P-One-way-Delay

Измерение односторонней задержки вместо двусторонней задержки вызвано следующими факторами:

- в гетерогенных каналах путь от источника к пункту назначения может отличаться от пути от пункта назначения обратно к источнику («асимметричные пути»), так что для прямого и обратного пути используются разные последовательности промежуточных сетевых узлов. Независимое измерение каждого пути отражает различие в производительности между двумя путями;

- даже если два пути симметричны, они могут иметь различающиеся задержки из-за асимметричной организации механизма очередизации;

- производительность приложения может зависеть в основном от производительности в одном направлении. Например, передача файлов с использованием TCP может больше зависеть от производительности направления потока данных, а не от направления передачи подтверждений;

- в сетях с поддержкой КО обеспечение в одном направлении может радикально отличаться от предоставления в обратном направлении, и, следовательно, гарантии КО различаются. Независимое измерение путей позволяет проверить обе гарантии.

Джиттер – Jitter

Джиттер - это разница в задержках пакетов в одном и том же потоке.

Если период до того, как пакет, достигший устройства, будет отправлен устройством, отличается от одного пакета к другому в потоке, возникают колебания и ухудшается качество обслуживания. Для некоторых служб (голос, видео) наличие высокого джиттера является недопустимой ситуацией, так как происходят прерывания в реальном времени.

Коэффициент потерь – Loss rate

Потеря пакетов происходит, когда один или несколько пакетов данных, передающихся по СПД, не достигают назначения. Потеря пакетов может быть вызвана либо ошибками при передаче данных, либо перегрузкой сети. Потеря пакетов измеряется как процент потерянных пакетов по отношению к отправленным пакетам:

$PLR = \frac{N^{tx} - N^{rx}}{N^{tx}}$,

где N_{tx} и N_{rx} — общее количество переданных и полученных протокольных единиц данных (PDU) соответственно. Эту оценку можно легко выполнить путем извлечения всех размеров пакетов в реальном времени, которые передаются и принимаются соответственно.

3. Методы сбора и хранения сетевого трафика

В работе [3] были рассмотрены три основных метода сбора трафика, которые имеют различные требования к объему памяти:

- сбор всех пакетов;
- сбор сетевого потока;
- сбор расширенного потока.

В первом методе производятся сбор, обработка и хранение копии каждого пакета трафика для последующего анализа. Эти данные содержат полную информацию о заголовках пакетов и передаваемой в пакетах информацией. При использовании второго метода размер собранной информации уменьшается за счёт того, что сетевой поток представляет из себя набор сетевых пакетов, для которых выполняются ряд условий (таких, например, что пакеты

посылаются примерно в одно время, имеют один и тот же адрес источника и номер порта, имеют один и тот же адрес назначения и номер порта, используют один и тот же протокол). Сбор расширенного потока включает в себя сбор всех пакетов и сетевого потока. При этом к информации потока добавляется информация, взятая непосредственно из заголовков пакетов или из передаваемой в пакетах информации. Вместе с тем расширенный поток также может содержать дополнительную информацию о каком-то внешнем источнике, например, о географическом расположении IP-адресов источника и назначения. Для задач прогнозирования КО гетерогенного канала в СПД достаточно собирать информацию только о заголовках сетевых пакетов и применять хэш-функцию к пользовательским данным. При сборе данных устройство, на котором осуществляется сбор трафика также проводится его анонимизация, включающая анонимизацию конфиденциальных данных и урезание данных, не требуемых для дальнейшего использования.

При сборе трафика каждый поступающий на устройство пакет должен проходить обработку, включающую следующие этапы:

1. фильтрация;
2. анонимизация;
3. сжатие;
4. сохранение.

Для хранения сетевого трафика могут быть использованы различные способы хранения данных, такие, как файлы (например, лог-файлы); базы данных и их комбинации [3]. Каждый из этих способов имеет собственные аспекты. Большинством приложений сбора и анализа данных поддерживается формат pcap [4], поэтому он был выбран в качестве формата хранения.

4. Архитектура средств сбора, хранения и анализа сетевого трафика

Возможность захвата пакетов и сохранения их для последующего анализа позволяет разделить задачи сбора информации и ее анализа.

Поток, согласно RFC3917 [5], определяется как набор IP-пакетов, проходящих через точку наблюдения в сети в течение определенного интервала времени. Все пакеты, принадлежащие

определенному потоку, имеют набор общих свойств. Каждое свойство определяется как результат применения функции к значениям:

- поля заголовков канального уровня (например, метки MPLS);
- поля заголовка сетевого уровня пакета (например, IP-адрес назначения);
- поля заголовка транспортного уровня (например, номер порта назначения);
- опционально, поля заголовка приложения (например, поля заголовка RTP [6]);
- характеристики, полученные в результате обработки пакета (например, сетевой интерфейс, используемый при наблюдении).

Для организации сбора и хранения сетевого трафика была разработана архитектура средств сбора, хранения и анализа сетевого трафика TrafficKeeper, указанная на рисунке 1.

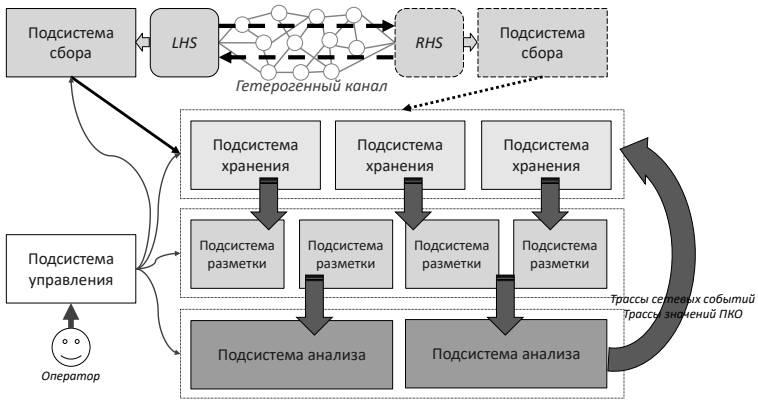

Рис. 1: Общее представление архитектуры средств сбора, хранения и анализа сетевого трафика

В состав средств сбора, хранения и анализа сетевого трафика TrafficKeeper входят следующие подсистемы:

- подсистема управления;
- подсистема сбора;

- подсистема хранения;
- подсистема разметки;
- подсистема прогнозирования.

При получении команды инициализации сбора трафика от подсистемы управления проводится подготовка подсистемы сбора к сбору трафика, одним из действий которой является получение от подсистемы хранения сокета точки принятия трафика. После этого при наступлении времени начала сбора происходит запуск сбора и отправка собираемого трафика в подсистемы хранения. Также в процессе сбора в подсистему хранения отправляется информация о текущем состоянии сбора. При завершении сбора в в подсистему хранения отправляется отчет о проведении сбора.

При получении команды инициализации сбора трафика от узла управления проводится подготовка подсистемы хранения к принятию трафика, одним из действий которой является отправка в соответствующие подсистемы сбора сокетов точек принятия трафика. При поступлении очередной порции трафика от подсистемы сбора происходит сохранение данной порции. Также в процессе сбора происходит получение от подсистемы сбора информации о текущем состоянии сбора и ее сохранение. При завершении сбора происходит получение от подсистемы сбора отчета о проведении сбора.

В результате сбора трафика в подсистеме хранения сохраняется следующая информация:

- идентификатор трафика;
- описание трафика;
- тип возникновения трафика (собранный или сгенерированный);
- количество узлов трафика;
- идентификаторы узлов сбора;
- пакеты, входящие в трафик;
- период сбора трафика (содержит времена начала и конца сбора трафика);
- конфигурация фильтрации трафика;
- конфигурация анонимизации трафика;

- отчет о проведении сбора трафика.

Разметка трафика инициируется оператором и предполагает

1. передачу трассы сетевого трафика (ТСТ) из подсистемы хранения в подсистему разметки;
2. разметку ТСТ, заключающуюся в вычислении параметров КО;
3. передачу разметки ТСТ из узла разметки в узел хранения и сопоставления разметки собранной ТСТ.

Прогнозирование значений параметров КО также инициализируется оператором и предполагает:

1. передачу ТСТ и её разметки из подсистемы хранения в подсистему прогнозирования;
2. вычисление прогнозируемых значений параметров КО;
3. передачу результатов прогнозирования ТСТ в подсистему хранения.

5. Заключение

В данной работе рассматривается задача прогнозирования качества гетерогенных каналов в сетях передачи данных. Рассматриваются количественные параметры КО, методы сбора и хранения сетевого трафика, а также предлагается архитектура средств сбора, хранения и анализа сетевого трафика. В дальнейшем планируется формализовать задачу прогнозирования значений показателей КО, исследовать методы её решения на основе вероятностно-статистических подходов и методов машинного обучения, а также полностью реализовать средство сбора, хранения и анализа сетевого трафика TrafficKeeper.

Литература

1. Колотовкин И. С. *Нейронные сети в задачах распределения трафика компьютерных сетей* //Нейрокомпьютеры и их применение. – 2020. – с. 330-332.

2. Huawei Technologies Co., Ltd. *QoS Technology White Paper-6W10*, Май 2013 https://etcdf.tc.df.gov.br/?a=documento&f=downloadPDF&iddocumento=1598002

3. Шыхалиев Р.Г., О методах сбора и хранения сетевого трафика компьютерных сетей // Big data: imkanlarl, multidissiplinar problemleri ve perspektivleri - 2016 - с. 67-69.

4. Harris G., Richardson M. PCAP Capture File Format [HTML] (https://datatracker.ietf.org/doc/id/draft-gharris-opsawg-pcap-00.html) (дата обращения: 01.09.2023).

5. J. Quittek, T. Zseby, B. Claise, S. Zander, *Requirements for IP Flow Information Export (IPFIX)*, RFC3917, Октябрь 2004 https://www.rfc-editor.org/info/rfc3917 (дата обращения: 01.09.2023)

6. Schulzrinne H. et al. RFC3550: RTP: A transport protocol for real-time applications. – 2003 (дата обращения: 01.09.2023).

Тюшев М. В., Смелянский Р. Л.

Об использовании графовых баз данных для маршрутизации в программно-конфигурируемых сетях

Введение

Программно-конфигурируемые сети (ПКС)[1] —это сети с централизованным программным управлением. Их популярность связана с тем, что они позволяют гибко и эффективно реагировать на изменения требований со стороны приложений к инфраструктуре вычислений. Именно приложения, точнее их требования, являются основной движущей силой развития инфраструктуры вычислений. Данные аналитической компании MarketsandMarkets свидетельствуют, что в 2019 году объём мирового рынка программно-конфигурируемых сетей и дата-центров достиг $51,7 млрд[2]. Эксперты утверждают, что рынок является растущим и останется таковым. Прогнозируется, что доходы от продаж программно-конфигурируемых решений на глобальном уровне будут увеличиваться на 25,5% ежегодно, и к 2024 году достигнут отметки в $160,8 млрд [2]. Эксперты IDC также подтверждают растущий спрос на программно-конфигурируемые технологии [2].

Процесс внедрения в практику ПКС сетей естественно заставляет переосмысливать решения, которые были разработаны для традиционных сетей. Одной из проблем является проблема маршрутизации. Традиционные алгоритмы сетевой маршрутизации модифицируются и применяются к ПКС сетям для решения задачи маршрутизации трафика. Однако одним из критических параметров таких алгоритмов является сложность, которая определяет время сходимости сети —понятие, означающее период времени, необходимый для приведения таблиц маршрутизации на сетевых устройствах в актуальное состояние. В ПКС сети на этот параметр существенное влияние оказывают количество сетевых устройств в контуре данных и их удаленность от контура управления. В традиционных TCP/IP сетях данная проблема маршрутизации решается путём разбиения сети на локальные зоны (area), размер локальной зоны в топологии автономной системы выбирают так, чтобы ее можно было бы представить структурой данных с операциями поиска пути в графе приемлемой сложности. Для ПКС сетей такой подход невыгоден. Одна из причин –

сложность контура управления – для каждой области свой контур управления.

Одним из возможных решений задачи маршрутизации трафика в ПКС сетях могут стать графовые базы данных (ГБД) [3]. ГБД —это специализированная платформа для построения графов и управления ими. Данные в графовых базах данных хранятся в виде вершин и ребер, которые представляют объекты и связи между ними. Узлы содержат информацию о каждом объекте, такую как его идентификатор, тип и произвольный набор атрибутов. Ребра представляют связи между узлами и содержат идентификатор, информацию о типе связи, ее направлении и произвольный набор атрибутов. Графовые базы данных также предоставляют эффективно реализованные графовые алгоритмы, например поиск маршрутов.

1. Постановка задачи

Пусть $N =< G, f >$ —транспортная сеть с топологией, представленной графом $G =< V, E >$, где

- V —множество вершин,
- E —множество ребер, на котором задана функция разметки f
- $f : e \to d(e), e \in E, d(e) \in \mathbb{R}^+$ —вес ребра.

Пусть n—количество вершин в графе, m —количество ребер, $p = \frac{m}{n(n-1)}$ —плотность матрицы смежности графа.

Пусть задан поток отказов $R(t_i) = (r_1(t_i), \ldots, r_\xi(t_i))^T$, $\xi \leq n$ в дискретный момент времени $t_i = t_{i-1} + \Delta t$, где Δt —заданный интервал, $r_j(t_i)$ —вероятность отказа j-ого ребра в момент времени t_i.

Пусть $C(N, R, p)$ —функция стоимости алгоритма маршрутизации по Дейкстре. $G(N, R, p)$ —функция стоимости алгоритма маршрутизации, использующего графовую базу данных. Функции стоимости определяются как время, затраченное на расчёт маршрутов.

Пусть T —время сходимости сети (время, требуемое на приведение таблиц в состояние, отражающее актуальное состояние сети). Таким образом, задача заключается в выборе средства маршрутизации при заданных (N, R, p) по критерию:

$$T = \min_{(N,R,p)} \{C(N, R, p), G(N, R, p)\}$$

2. Обзор существующих решений

ГБД представляют собой специальный тип баз данных, ориентированных на хранение и обработку графовых структур данных. ГБД —это перспективный инструмент для анализа и обработки информации, структура которой может быть представлена графом.

Цель настоящего обзора заключается в выборе реализации ГБД для проведения экспериментального исследования, основываясь на следующих критериях: открытость СУБД, наличие реализаций графовых алгоритмов, масштабируемость, надежность и сложность использования. В данной работе рассматриваются несколько самых популярных реализаций графовых баз данных, а именно: Neo4j[3], ArangoDB[4], MongoDB[5] и Titan[6].

Neo4j — это графовая система управления базами данных с открытым исходным кодом, реализованная на Java. Она является ведущей графовой СУБД в мире. С помощью Neo4j можно легко создавать и обрабатывать графы, а также использовать высокоуровневые инструменты, которые предоставляет система. Neo4j обладает простым и интуитивно понятным языком запросов Cypher. Также важно отметить, что СУБД Neo4j включает Graph Data Science (GDS) — это библиотека предоставляющая набор инструментов для анализа графовых данных в Neo4j. GDS разработана специально для Neo4j и обеспечивает мощные возможности для анализа графов, включая такие операции, как поиск маршрутов, анализ центральности, поиск сообществ, анализ схожести и другие. GDS предоставляет высокоуровневый API для выполнения запросов и операций над графом, что делает работу с графом более удобной и эффективной. Эта библиотека позволяет быстро и легко проводить сложные анализы на масштабных графах, что делает ее важной составляющей в экосистеме Neo4j.

ArangoDB —мультимодельная база данных с открытым исходным кодом. Она представляет собой комбинацию графовой базы данных, базы данных документов[7] и хранилище с доступом по ключу[8] в одной системе. ArangoDB обеспечивает высокую производительность и масштабируемость, а также поддерживает различные языки программирования. ArangoDB обладает универсальным языком запросов AQL, который поддерживает не только графовые запросы, но и запросы к документным и ключ-значение хранилищам. Это делает его более гибким в использовании, особенно в случаях, когда база данных содержит не только графовые данные.

MongoDB —мультимодельная база данных, с открытым исходным кодом. MongoDB — документоориентированная[7] СУБД, однако в последнее время ее функциональность была расширена до графовых структур данных. MongoDB обеспечивает высокую производительность и масштабируемость. Для хранения графовых данных в MongoDB можно использовать набор документов, которые представляют вершины графа, и другой набор документов, который представляет связи между вершинами. Однако, следует отметить, что MongoDB не имеет инструментов для работы с графовыми данными, например таких как Cypher или AQL. В следствие чего работа с более сложными запросами к графам требует больше усилий и ресурсов для реализации операций, связанных с графовой обработкой.

Titan —это масштабируемая ГБД, оптимизированная для хранения графов и запросов, содержащих сотни миллиардов вершин и ребер, распределенных по кластеру из нескольких машин. Titan — это транзакционная база данных, которая может поддерживать тысячи одновременных пользователей, выполняющих сложные обходы графов в реальном времени. Titan предоставляет возможность автоматического распределения и репликации данных для повышения производительности и отказоустойчивости. Titan поддерживает графовый язык запросов Gremlin, а также приложения Gremlin —инструменты, созданные на основе API-интерфейсов, предоставляющие пользователям общие алгоритмы работы с графами.

Из рассмотренных реализаций ГБД, Neo4j и ArangoDB и Titan являются узкоспециализированными системами управления графами. Они используют оптимизированные алгоритмы и структуры данных для хранения и обработки графов, поэтому Neo4j, ArangoDB и Titan имеют высокую производительность при работе с данными, представленными в виде графовой структуры, что является основным преимуществом для целей маршрутизации в сети. Эти реализации также предоставляют удобные языки запросов и имеют высокую надежность. Они могут быть эффективным инструментом для решения задачи маршрутизации в программно-конфигурируемых сетях. Однако главным преимуществом Neo4j является наличие GDS, что делает работу с графами более удобной и эффективной, поэтому Neo4j была выбрана в качестве исследуемой ГБД.

3. Методика и инструментарий

Для проведения сравнительного анализа эффективности использования ГБД и традиционных подходов маршрутизации на ПКС контроллере по критерию минимизации времени сходимости сети в зависимости от её топологии и потока отказов, была разработана следующая методика эксперимента.

При проведении экспериментального исследования были сделаны следующие допущения:

1. $f \equiv 1$ —используется метрика "хопов" (число переходов между сетевыми устройствами).

2. Поток отказов R имеет равномерное распределение и в каждый момент времени t_i отказывает только одно ребро.

Исследование проводится методом статистических испытаний на стенде, который включает имитационную модель сети (контура данных) Mininet[9], сетевую операционную систему RunOS[1] и ее приложения "маршрутизация" и приложение маршрутизации, использующее ГБД Neo4j.

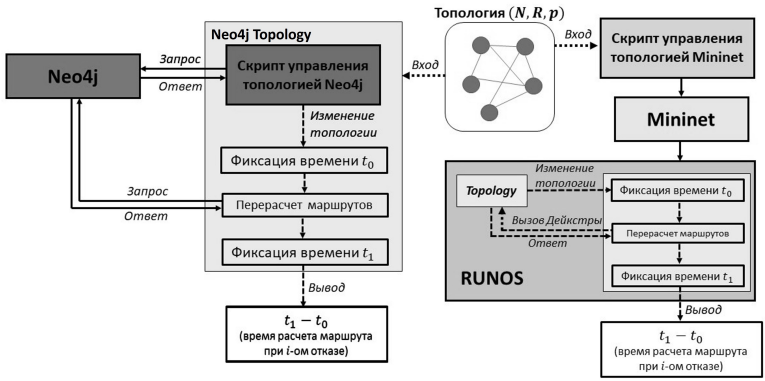

Рис. 1: Экспериментальный стенд

3.1 Технические характеристики стенда

- ОС: Ubuntu 22.04
- ЦП: Ryzen 5 5600X 6 × 3.7 ГГц
- ОЗУ: 32 ГБ 3200 ГГц

3.2 Методика проведения экспериментального исследования

Для каждой входной топологии (N, R, p)

$$R(t_i) = (r_1(t_i), \ldots, r_\xi(t_i))^T$$

$$r_{j_i}(t_i) = 1,\ r_k(t_i) = 0\ \forall k \neq j_i,\ 1 \leq j_i,\ k \leq \xi$$

Причем

$$j_i \neq j_k\ \forall i \neq k,\ 1 \leq i,\ k \leq \xi$$

Далее приведена последовательность действий, предпринимаемых при проведении эксперимента, для RunOS и Neo4j. Действия выполняются последовательно.

RunOS

1. Запуск скрипта управления топологией Mininet ($i = 1$).
2. Удаление ребра, соответствующего $R(t_i)$ (имитация отказа).
3. Фиксация времени τ_0, запуск функции перерасчета маршрута.
4. Получение перерасчитаного маршрута, фиксация времени τ_1, $(i = i + 1)$, Если $i < \xi$ переход к п. (2).

** После выполнения п. (4) происходит запись результата $\tau_1 - \tau_0$ —время перерасчета маршрута при i-ом отказе.*

Neo4j

1. Запуск скрипта управления топологией Neo4j ($i = 1$).
2. Удаление ребра, соответствующего $R(t_i)$ (имитация отказа).
3. Фиксация времени τ_0, запрос на перерасчет маршрута к СУБД.
4. Получение перерасчитаного маршрута, фиксация времени τ_1, $(i = i + 1)$, Если $i < \xi$ переход к п. (2).

** После выполнения п. (4) происходит запись результата $\tau_1 - \tau_0$ —время перерасчета маршрута при i-ом отказе.*

4. Результаты

Далее приведены результаты экспериментального исследования. Обозначения:

- N — количество вершин в топологии
- p — плотность топологии
- Вертикальная ось —время перерасчета маршрута
- Горизонтальная ось —момент времени

4.1 Реальные топологии

Представленные далее топологии, заимствованы с TopologyZoo.

Рис. 2: Топология GARP: $N = 54,\ p = 0.04$

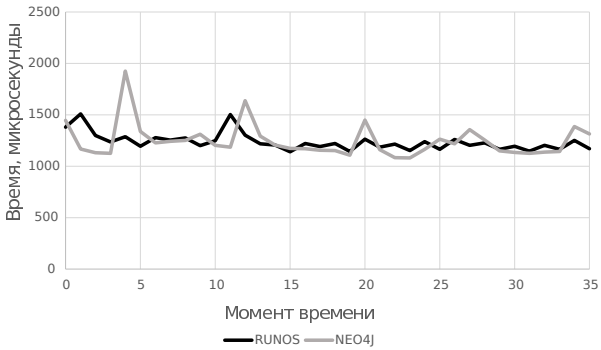

Рис. 3: Зависимость времени расчета маршрута в зависимости от дискретного момента времени для топологии GARP

Рис. 4: Топология Colt Telecom: $N = 153,\ p = 0.016$

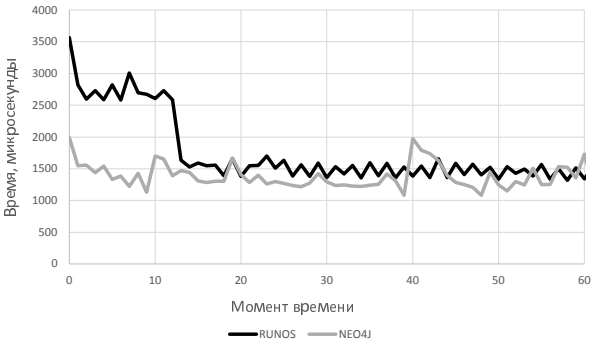

Рис. 5: Зависимость времени расчета маршрута в зависимости от дискретного момента времени для топологии Colt Telecom

4.2 Сгенерированные топологии

Топологии генерируются с помощью NetworkX — это Python пакет для создания, управления и изучения структуры, динамики и функций сложных графов.

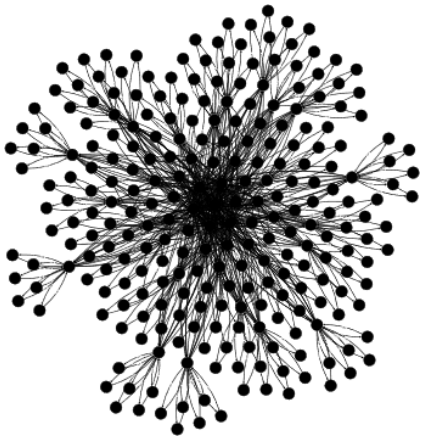

Рис. 6: Топология Fat Tree: $N = 258,\ p = 0.013$

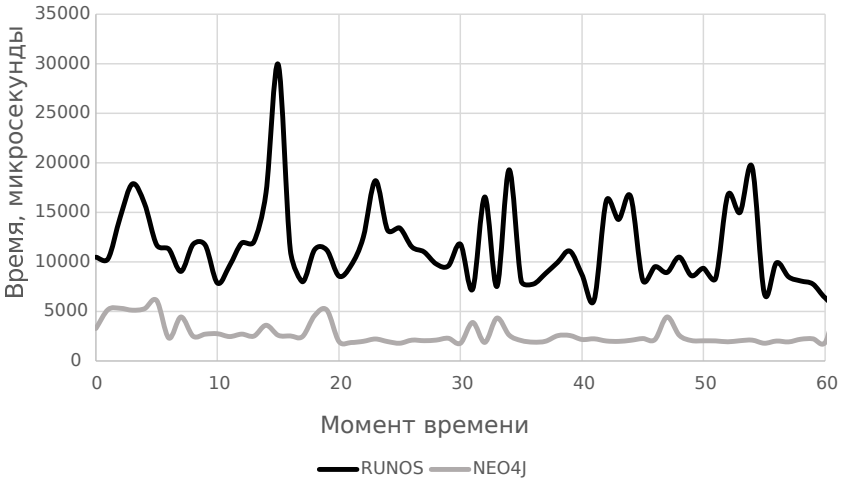

Рис. 7: Зависимость времени расчета маршрута в зависимости от дискретного момента времени для топологии Fat Tree

Из представленных результатов, можно заметить, что при

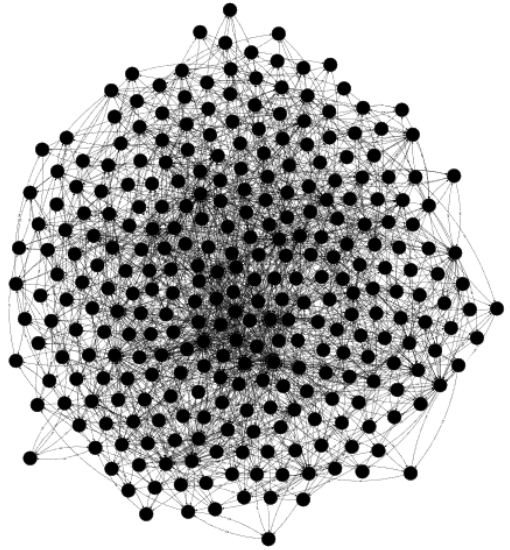

Рис. 8: Топология D500: $N = 500,\ p = 0.011$

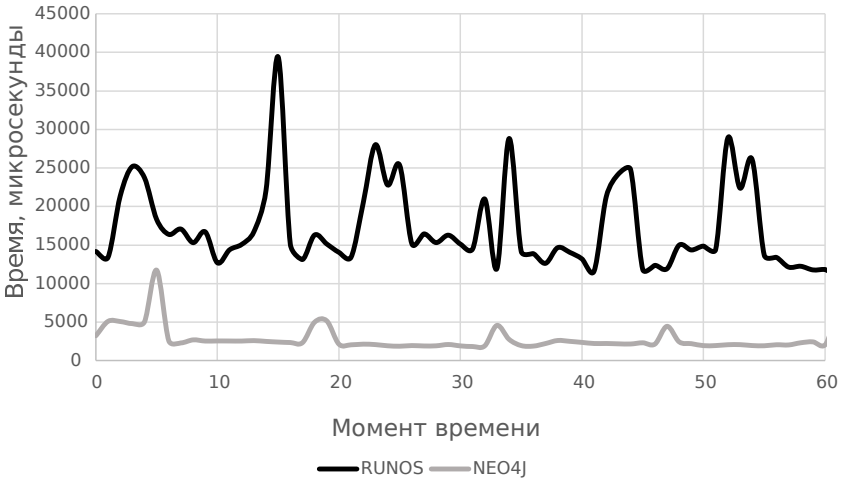

Рис. 9: Зависимость времени расчета маршрута в зависимости от дискретного момента времени для топологии D500

увеличении количества узлов топологии графовая СУБД Neo4j, в отличие от RunOS, не теряет производительности.

Далее представлено сравнение времени перерасчета маршрутов в

логарифмической шкале в Neo4j и RunOS для топологий от 100 до 500 узлов с плотностями от 0.02 до 0.1.

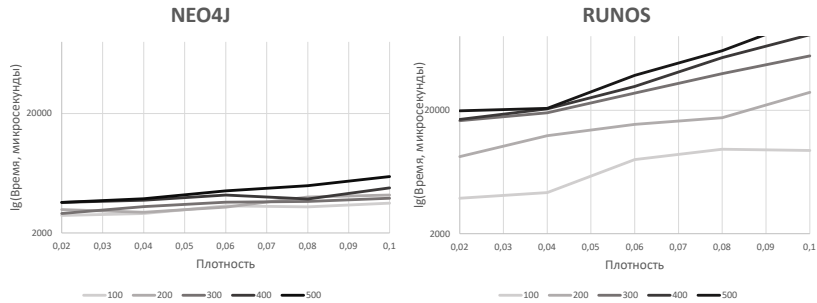

Рис. 10: Зависимость времени расчета маршрута в зависимости от дискретного момента времени для топологии D500

Из представленных на Рис. 10 результатов можно видеть, что при увеличении плотности топологии графовая СУБД Neo4j, в отличие от RunOS, не теряет производительности.

Далее представлены результаты измерения времени перерасчета маршрутов в Neo4j для топологий, содержащих от 100 до 1000 узлов с плотностями 0.1 до 1.

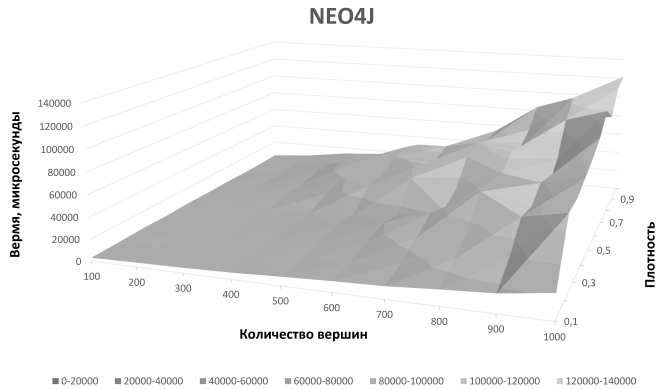

Рис. 11: Зависимость среднего времени расчета маршрута от плотности и количества вершин в топологии

Из представленных результатов видно, что Neo4j сохраняет производительность при масштабировании топологии.

5. Заключение

В результате проведенной работы был выполнен сравнительный анализ различных ГБД для маршрутизации в ПКС сетях. Из предложенных баз данных —Neo4j, ArangoDB, MongoDB, Titan —была выбрана Neo4j по ряду критериев, таких как открытость СУБД, наличие реализаций графовых алгоритмов, масштабируемость, производительность запросов, надежность и сложность использования.

Для проведения экспериментального исследования был разработан стенд, который включал в себя имитационную модель сети (контура данных) Mininet, сетевую операционную систему RunOS и приложение "маршрутизация", а также приложение маршрутизации, использующее ГБД Neo4j.

В результате экспериментального исследования было выявлено, что ГБД эффективны для решения задачи маршрутизации. Также было показано, что ГБД лучше всего справляются с плотными топологиями. При работе с небольшими топологиями использование ГБД не оправдано.

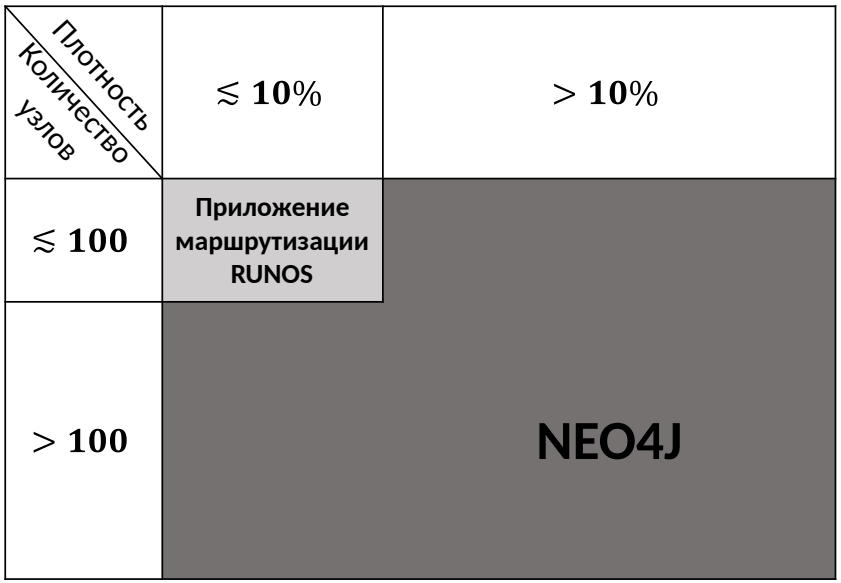

Рис. 12: Выбор средства маршрутизации в зависимости от размера и плотности топологии

Таким образом, можно сделать вывод, что графовые базы данных

хорошо подходят для решения задачи маршрутизации в ПКС при больших топологиях и могут быть эффективным инструментом для управления сетью.

Однако вопрос о возможном снижении производительности маршрутизации за счет затрат на информационное взаимодействие RunOS и Neo4j остается открытым для дальнейших исследований.

Литература

1. *Смелянский Р. Л., Антоненко В. А.* Концепции программного управления и виртуализации сетевых сервисов в современных сетях передачи данных. / Антоненко В. А. Смелянский Р. Л. — Москва Курс 2019.

2. *TAdviser.* Программно-определяемые сети Software-Defined Network, SDN. — TAdviser —российский интернет-портал и аналитическое агентство. URL: `https://www.tadviser.ru/a/158716`. Дата обращения: 05.05.2023.

3. *Robinson, Ian.* Graph databases: new opportunities for connected data / Ian Robinson, Jim Webber, Emil Eifrem. — O'Reilly Media, Inc. 2015.

4. *Fernandes, Diogo.* Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB. / Diogo Fernandes, Jorge Bernardino // Data. — 2018. — Pp. 373–380.

5. *Parmar, Raj R.* MongoDB as an efficient graph database: An application of document oriented NOSQL database / Raj R Parmar, Sudipta Roy // Data Intensive Computing Applications for Big Data. — IOS Press, 2018. — Pp. 331–358.

6. *Mishra, Vivek.* Titan graph databases with cassandra / Vivek Mishra, Vivek Mishra // Beginning Apache Cassandra Development. — 2014. — Pp. 123–151.

7. *Nayak, Ameya.* Type of NOSQL databases and its comparison with relational databases / Ameya Nayak, Anil Poriya, Dikshay Poojary // International Journal of Applied Information Systems. — 2013. — Vol. 5, no. 4. — Pp. 16–19.

8. *Nguyen, Thanh Trung.* Zing database: high-performance key-value store for large-scale storage service / Thanh Trung Nguyen, Minh Hieu Nguyen // Vietnam Journal of Computer Science. — 2015. — Vol. 2, no. 1. — Pp. 13–23.

9. *Kaur, Karamjeet.* Mininet as software defined networking testing platform / Karamjeet Kaur, Japinder Singh, Navtej Singh Ghumman // International conference on communication, computing & systems (ICCCS). — 2014. — Pp. 139–42.

Цветкова В. П., Степанов Е. П.

Исследование методов эффективного общения в многоагентном обучении для задачи балансировки сетевого трафика

Введение

В современном мире наблюдается постоянный рост объема сетевого трафика. Отчет Cisco [1] показывает, что объем трафика, передаваемого через Интернет, с 2015 по 2020 год возрос с 0.16 петабит/с до 0.5 петабит/с. Это означает, что нагрузка на каналы передачи данных увеличивается и даже может превысить их пропускную способность, тем самым вызвав перегрузку. Перегрузкой канала называется состояние, когда суммарная интенсивность поступления потоков на вход канала превышает его пропускную способность. Одним из методов для борьбы с перегрузкой является распределение потоков данных по нескольким маршрутам, который далее будем называть балансировкой сетевого трафика. В работе рассматривается подход к балансировке, основанный на многоагентных методах машинного обучения с подкреплением.

Многоагентное обучение с подкреплением позволяет разрабатывать алгоритмы для координации действий агентов в изменяющейся среде, где каждый агент стремится максимизировать свою награду. Такие алгоритмы могут использоваться для решения задачи балансировки сетевого трафика, поскольку взаимодействие со средой позволяет адаптироваться к изменяющимся условиям и выбирать действие, которое принесет наибольшую награду в текущем состоянии среды.

Существует три подхода к управлению многоагентными системами: централизованный, децентрализованный и распределенный. Централизованные системы [2] — это системы, в которых есть единая точка управления, называемая центральным контроллером. Все агенты системы выполняют команды, выданные центральным контроллером. В децентрализованных системах (fully decentralized [3]) каждый агент принимает свое собственное решение, основываясь только на своей истории выполнения действий и локальных состояниях. Конечное поведение системы — это совокупность решений отдельных агентов. Распределенная

система — это система, в которой процесс принятия решения распределен между агентами, и у каждого агента есть возможность обмениваться информацией с другими агентами для достижения общей цели.

В этой статье рассматривается распределенный подход в задаче балансировки трафика, цель которой заключается в обеспечении равномерной загрузки каналов сети. Для достижения этой цели агенты обмениваются данными о своем состоянии с другими агентами перед принятием решения о распределении сетевого трафика по каналам. Обмен данными и их обработка для принятия решения создают накладные расходы, поэтому важно свести к минимуму взаимодействие агентов при обеспечении оптимальной балансировки трафика. Будем называть общение *эффективным* в таких методах, которые позволяют уменьшить обмен информацией между агентами. Эффективное общение может существенно повлиять на производительность системы, так как помогает уменьшать накладные расходы [4].

Первый раздел этой статьи посвящен описанию задачи балансировки трафика и алгоритму ее решения. Во втором разделе рассматриваются существующие методы эффективного общения и их применимость к описанной задаче. В третьем разделе представлены цели экспериментального исследования, методика и результаты, а четвертый раздел является заключением к работе.

1. Балансировка сетевого трафика

1.1 Описание задачи

Топология сети ТР задана с помощью ориентированного графа G = (V, E), где V – множество вершин, которые соответствуют устройствам, перенаправляющим трафик в сети, E – множество ребер, которые являются каналами в сети. В графе каждому ребру $e(u, v) \in E$ из вершины u в вершину v сопоставлены два параметра $c_{u,v}$ — пропускная способность канала, $b_{u,v}$ — текущая нагрузка на канал.

Задача балансировки трафика состоит в минимизации отклонения загрузки каналов от средней загрузки каналов в сети, то есть в минимизации функции Ф:

$$\Phi = \frac{1}{N} \sum_{u,v} \left(\frac{b_{u,v}}{c_{u,v}} - \mu\right)^2,$$

где $\mu = \frac{1}{N} \sum_{u,v} \frac{b_{u,v}}{c_{u,v}}$ — средняя загрузка каналов в сети, N — количество каналов. Эту функцию будем называть *целевой*.

1.2 Алгоритм решения

Для описания функционирования сети агентов будем использовать как и в статье [5] марковский процесс принятия решения: (S, A, r, P, γ), где S — пространство состояний среды, A — пространство действий, r — пространство наград после выполнения действий, P — функция перехода в следующее состояние, $\gamma \in (0,1]$ — коэффициент дисконтирования.

В качестве алгоритма обучения с подкреплением используется алгоритм градиентной политики "Субъект-Критик" (Actor-Critic), основанный на алгоритме Proximal Policy Optimization (PPO) [6]. Алгоритм "Субъект-Критик" предполагает, что все агенты имеют свою роль: субъект или критик. Субъекты учатся выбирать действия на основе текущего состояния среды, а централизованный критик оценивает выбранное действие и состояние среды. Критик предоставляет для каждого субъекта обратную связь после оценки пар состояние-действие, которая осуществляется на основе вознаграждений, полученных от среды. Каждый субъект обновляет свою политику на основе информации критика, где под политикой понимается распределение вероятностей по доступным действиям в данном состоянии.

PPO — алгоритм обучения с подкреплением, основная идея которого заключается в обучении агента находить оптимальную политику в задачах обучения с подкреплением. Оптимальная политика определяет, какие действия агент должен выполнять в различных состояниях среды для максимизации награды.

Для общения между агентами используется модель Message Passing Neural Network (MPNN) [7], которая является частью класса нейронных сетей Graph Neural Networks (GNN) [8] и основана на итеративном алгоритме передачи сообщений, который преобразует передаваемую между агентами информацию. В предложенном в статье [9] методе каждый агент контролирует один канал, и агенты обмениваются текущей занятой пропускной способностью канала. Каналы являются *соседними*, если у них есть общая вершина в топологии сети. Будем называть агентов *соседями*, если каналы, которые они контролируют, являются *соседними*. Каждый агент обменивается данными со своими соседями.

Требуется исследовать методы эффективного общения в применении к модели MPNN и реализовать наиболее подходящий для задачи балансировки сетевого трафика таким образом, чтобы уменьшение количества передаваемых данных не приводило к уменьшению скорости обучения алгоритма.

2. Обзор методов эффективного общения

Методы эффективного общения сравнивались по следующим критериям:

- Зависимость метода от структуры сообщения. Все рассмотренные многоагентные методы с эффективным общением можно классифицировать на две группы:
  1. Работающие с уменьшением количества итераций обмена сообщениями;
  2. Направленные на минимизацию объема самих сообщений.
  Соответственно, первая группа не зависит от структуры передаваемых данных, а вторая — зависит, так как необходимо работать с объемом сообщений, а значит непосредственно со значениями сообщений.

- Вероятность потери значащих сообщений. Если на участке сети не произошло изменений, то агент будет передавать ту же информацию, что и в предыдущий раз. Такие сообщения будем называть незначащими, так как они не несут новой информации, а все остальные сообщения являются значащими. Потерю значащих сообщений необходимо свести к минимуму, чтобы у каждого агента было достаточно информации для принятия верных решений по балансировке трафика в сети.

- Неуменьшение скорости обучения. Необходимо увеличить или оставить той же скорость обучения алгоритма по сравнению с той скоростью, что была до применения метода эффективного общения.

- Уменьшение объема передаваемых данных по сравнению с объемом, который передавался до применения метода эффективного общения.

В методе LAPG [10] агенты обмениваются данными только в том случае, если накопилось достаточно изменений относительно прошлого раза. Приведенные в статье эксперименты показывают хорошие результаты: алгоритм с применением LAPG достигает того же значения усредненной награды, что и алгоритм без LAPG, при этом количество обменов между агентами в 3 раза меньше. Метод может быть применен к задаче балансировки трафика.

Авторы статьи [11] предлагают метод CEAC, в котором агенты передают только часть своего вектора политик соседям. Такой

Метод\ Критерий	Структура собщения	Потеря значащих сообщений	Увеличение скорости обучения	Уменьшение объема передаваемых данных
LAPG [10]	Не зависит	Нет	Отсутствует	В 3 раза
CEAC [11]	Зависит	Да	Нет экспериментов	Нет экспериментов
Message-Dropout [12]	Не зависит	Да	В 1.4 раза	В 2 раза
VAC [13]	Зависит	Нет	В 1.3 раза	В 2 раза
CE HMG [14]	Не зависит	Да	Уменьшение	В 2 раза

Таблица 1: Результаты обзора

подход может быть применен в задаче балансировки трафика для сокращения векторов передаваемых сообщений при условии, что сокращение будет не за счет значащей части сообщения. Определение значащей части сообщения аналогично определению значащего сообщения.

Message-Dropout [12] — это метод нейронной сети, который основывается на алгоритме Dropout. Согласно этому алгоритму на этапе обучения нейронной сети передаваемые значения между слоями отбрасываются с заданной вероятностью p. Экспериментальная часть основана на задаче "Преследование-уклонение", в которой агенты-преследователи стараются догнать агентов-уклоняющихся. В каждый интервал времени каждый агент выбирает, в какую сторону (север, юг, запад, восток) двигаться. Соответственно, если агент-преследователь в текущий интервал времени получил информацию о том, где находится агент-уклоняющийся, а в следующий момент времени не получил, то положение агента-уклоняющегося изменилось только на один шаг в одну из сторон. В задаче балансировки трафика занятая пропускная способность канала в предыдущий интервал времени может сильно отличаться от следующего интервала времени. Например, пропускная способность канала была занята полностью на предыдущем интервале и освободилась на следующем. Если в

таком случае не передать информацию другим агентам, то канал может простаивать. Поэтому при таком отсеивании есть высокая вероятность потери значащих сообщений, и этот метод не подходит для задачи балансировки трафика.

Статья [13] посвящена методу VAC, где агент общается с другими агентами, только когда его локальное решение неоднозначно. Степень неоднозначности измеряется с помощью нахождения разницы между двумя самыми большими значениями, которые являются оценками качества действий. При получении запроса на общение агент отвечает в том случае, если его ответ считается однозначным. Этот метод зависит от структуры сообщения, так как основан на обмене информацией о действиях агентов, поэтому не подходит для задачи балансировки трафика, где агенты обмениваются занятой пропускной способностью канала.

Метод CE HMG [14] основан на решении задачи о двуруком бандите, где агент выбирает, выполнять ли обмен сообщениями, и если да, то выбирает, какому соседу передавать сообщение. После экспериментального исследования авторы приходят к выводу, что метод показывает результаты лучше, чем алгоритм Независимого Обучения, когда агенты вовсе не передают сообщения друг другу, но проигрывает алгоритму, где каждый агент общается с двумя и более агентами. Так как в задаче балансировки сетевого трафика каждый агент контролирует один канал, то соседей, с которыми он будет обмениваться информацией, как правило, минимум два. Поэтому из-за высокой вероятности потери значащих сообщений и уменьшения скорости обучения данный метод не может быть применен к задаче балансировки трафика.

По результатам проведенного обзора для проведения экспериментального исследования были выбраны методы LAPG [10] и CEAC [11].

3. Экспериментальное исследование

3.1 Цель экспериментального исследования

Цель: проверить сходимость многоагентного алгоритма и исследовать, как зависит скорость сходимости алгоритма и значение, к которому сходится функция Ф, от объема передаваемых сообщений и от количества переданных сообщений.

3.2 Методика исследования метода CEAC для уменьшения объема сообщений

Агенты могут сжимать передаваемые данные при помощи нейронных сетей с целью уменьшения объема сообщения. Сжатие сообщений нейронными сетями возможно благодаря тому, что они могут извлекать ключевые характеристики, за счет этого сохраняя существенную информацию.

Пусть $\vec{\alpha}^n = (\alpha_1, \ldots, \alpha_{32n})$ текущее сообщение для передачи, α_i = {0, 1}. Тогда сжатие происходит с помощью нейронной сети β, которая определена следующим образом: β: $\vec{\alpha}^n \to \vec{\alpha}^k$, где k < n, k > 1.

Метод сходится в том случае, если среднее значение функции Ф на протяжении DE эпизодов отличается от значения *Opt* не более, чем на ε. Значение *Opt* может быть как оптимальным, так и субоптимальным. Тогда скорость сходимости определим как $\frac{1}{R}$, где R — это минимальный номер эпизода среди таких, что на следующих DE эпизодах метод сходится:

$$R = \min\left(a \in N : \left|\frac{1}{DE}\sum_{i=a}^{a+DE}\Phi_i - \ Opt\right| < \varepsilon\right),$$

где a – номер эпизода, N – количество эпизодов в эксперименте. Значение *Opt* и параметр для сравнения ε заданы.

В предложенном в работе [5] методе агенты обмениваются сообщениями, состоящими из 16 чисел с плавающей точкой. Считаем, что размер каждого числа равен 4 байтам. Для сокращения объема сообщений в настоящей работе были проведены эксперименты с использованием метода CEAC, в каждом из которых по очереди выбиралось значение k из множества K = {2, 3, ..., 16}. Каждое значение k ∈ K — это количество чисел с плавающих точкой в сообщении, которое будем называть длиной сообщения. Было проведено исследование зависимости скорости сходимости алгоритма от значения длины сообщения k, при этом оптимальное значение *Opt* было одинаковым для всех эксперименотов, то есть функция Ф сходилась к одному и тому же значению с точностью до ε. Эксперименты проводились на топологии ромб и топологии из 16 вершин, представленных на Рис. 1 и Рис. 2 соответственно. Каждый запуск проводился на 2000 итераций алгоритма решения задачи балансировки сетевого трафика.

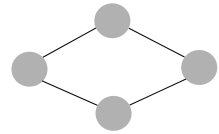

Рис. 1: Топология ромб

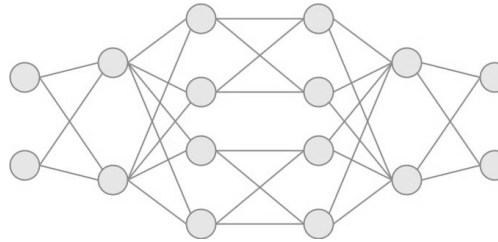

Рис. 2: Топология из 16 вершин

3.3 Методика исследования метода LAPG для уменьшения количества сообщений

Агенты могут пропускать передачу своего состояния, если не произошло больших изменений по сравнению с предыдущим состоянием. Пусть $\vec{\alpha}_1$, $\vec{\alpha}_2$ — первое и второе сообщения для передачи соответственно. Тогда сообщение $\vec{\alpha}_j$, j $>$ 2, передается другим агентам только в том случае, если произошло достаточно изменений, то есть евклидова норма разности векторов удовлетворяет следующему условию:

$$||(\vec{\alpha}_{j-1} - \vec{\alpha}_j)|| > \delta,$$

где δ — заданный параметр.

Для сокращения количества обменов на итерации, были проведены эксперименты с использованием метода LAPG. Перед принятием решения в алгоритме происходит заданное количество итераций обмена сообщениями между агентами. Была проведена серия экспериментов, учитывая разные варианты выбора параметра δ. В первом случае в качестве значения параметра δ было решено выбрать разность между сообщениями на второй и третьей итерации. То есть сообщения на первых трех итерациях передавались всегда, а последующие передавались только в том случае, если произошло достаточно изменений, которые

оценивались с помощью параметра δ. Во втором варианте параметр δ был подобран таким образом, что передача сообщений происходила только на первой итерации.

Было проведено исследование, в котором сравнивалось значение, к которому сходится функция Ф при использовании метода LAPG, со значением без использования этого метода в алгоритме решения задачи балансировки сетевого трафика. Также сравнивалась скорость сходимости алгоритма с использованием метода LAPG и без него. Эксперименты проводились на топологии из 16 вершин, представленной на Рис. 2. Каждый запуск проводился на 2000 итераций алгоритма.

3.4 Результаты экспериментов для уменьшения объема сообщений

На Рис. 3 и Рис. 4 представлены результаты проведенных экспериментов для двух топологий с разной длиной сообщения при использовании метода CEAC.

Из графика на Рис. 3 заметно, что результаты при объеме сообщений в 64 байта (размер k = 16) примерно те же, что и при объеме в 8 байт (размер k = 2). То есть при уменьшении объема сообщения с 64 до 8 байт скорость сходимости алгоритма почти не изменилась. Это значит, что без потери качества передаваемый объем сократился в 8 раз.

По графику, представленному на Рис. 4, можно сделать вывод о том, что результаты при объеме сообщений в 64 байта (размер k = 16) приблизительно те же, что и при объеме в 12 байт (размер k = 3). Значит при уменьшении объема сообщения с 64 до 12 байт скорость сходимости алгоритма сохранила свое значение, то есть без потери качества передаваемый объем сократился более, чем в 5 раз. Эксперименты также показали, что еще большее уменьшение объема приводит к увеличению R, а значит, уменьшению скорости сходимости.

Из графиков можно сделать вывод о том, что исходный алгоритм без применения методов эффективного общения не был оптимизирован по объему передаваемых сообщений. Метод CEAC позволяет это сделать.

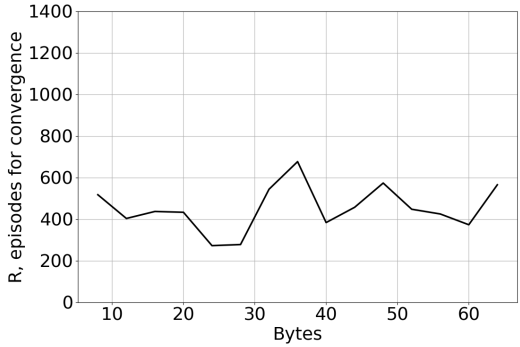

Рис. 3: Зависимость R от длины передаваемых сообщений для топологии ромб

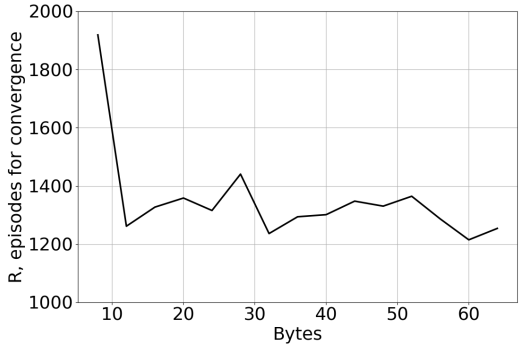

Рис. 4: Зависимость R от длины передаваемых сообщений для топологии из 16 узлов

3.5 Результаты экспериментов для уменьшения количества сообщений

На Рис. 5 и Рис. 6 представлены результаты проведенных экспериментов с разными значениями параметра δ. На Рис. 5 значение δ равно разности между сообщениями на второй и третьей итерации, а на Рис. 6 δ принимает значение, при котором общение агентов происходило только на первой итерации перед принятием алгоритмом решения.

Из двух графиков на Рис. 5 и Рис. 6 видно, что значение, к которому сходится функция Ф, осталось тем же при использовании

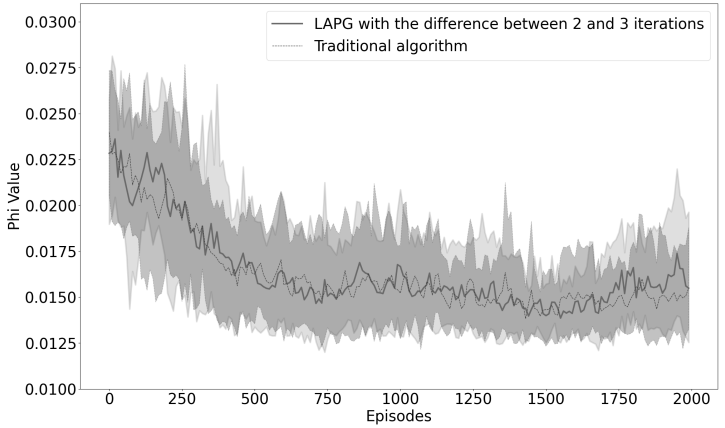

Рис. 5: Зависимость значения Ф-функции от номера эпизода в случае, когда значение δ равно разнице между сообщениями на второй и третьей итерации

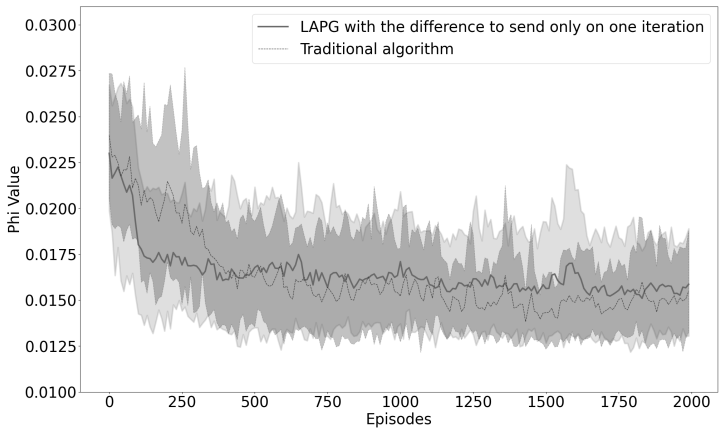

Рис. 6: Зависимость значения Ф-функции от номера эпизода в случае, когда δ принимает значение, при котором общение агентов происходило только на первой итерации

метода LAPG, что и было в алгоритме без использования метода LAPG. Также из графиков заметим, что на Рис. 5 скорость сходимости алгоритма осталась примерно той же с использованием метода LAPG, что и была без него, а на Рис. 6 скорость сходимости алгоритма увеличилась при использовании метода LAPG.

Количество передаваемых сообщений уменьшилось на Рис. 5 в 3 раза, а на Рис. 6 — в 5 раз. Метод LAPG является стабильным, так как разброс значений при его использовании сравним с разбросом без использования LAPG.

4. Заключение

Эффективное общение для многоагентных методов позволяет существенно уменьшить нагрузку на каналы передачи данных. Экспериментальное сравнение методов CEAC и LAPG с традиционными многоагентными методами машинного обучения показало следующие результаты:

1. Метод CEAC уменьшает передаваемый объем в 5 раз при сходимости функции Ф к одному и тому же значению, поддерживая ту же скорость сходимости.

2. Метод LAPG уменьшает количество передаваемых сообщений в 5 раз, сохраняя тем же значение сходимости функции Ф и увеличивая скорость сходимости алгоритма.

Дальнейший интерес для исследования представляет:

- Исследовать скорость сходимости при использовании алгоритма CEAC для других топологий сети агентов.

- Исследовать влияние значения параметра δ для алгоритма LAPG и выяснить эффективность алгоритма.

- Исследовать эффективность одновременного применения методов CEAC и LAPG.

Литература

1. Cisco Annual Reports: `https://www.cisco.com/c/en/us/about/annual-reports.html` [Дата доступа: 13.11.2023]

2. Milad Moradi. A centralized reinforcement learning method for multi-agent job scheduling in Grid // arXiv (preprint): `https://arxiv.org/abs/1609.03157`, 2016 [Дата доступа: 13.11.2023]

3. MA2QL: A Minimalist Approach to Fully Decentralized Multi-Agent Reinforcement Learning / Kefan Su, Siyuan Zhou, Jiechuan Jiang, Chuang Gan, Xiangjun Wang, Zongqing Lu // arXiv (preprint): https://arxiv.org/abs/2209.08244, 2022 [Дата доступа: 13.11.2023]

4. LAG: Lazily Aggregated Gradient for Communication-Efficient Distributed Learning / Tianyi Chen, Georgios B. Giannakis, Tao Sun, Wotao Yin // Part of Advances in Neural Information Processing Systems 31 (NeurIPS 2018)

5. On Fair Traffic allocation and Efficient Utilization of Network Resources based on MARL / Stepanov E.P., Smeliansky R.L., Plakunov A.V., Borisov A.V., Xia Zhud, Jianing Peid, Zhen Ya // ResearchGate (preprint): https://www.researchgate.net/publication/371166584_On_Fair_Traffic_allocation_and_Efficient_Utilization_of_Network_Resources_based_on_MARL, 2023 [Дата доступа: 13.11.2023]

6. Proximal Policy Optimization Algorithms / John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, Oleg Klimov // arXiv (preprint): https://arxiv.org/abs/1707.06347, 2017 [Дата доступа: 13.11.2023]

7. Neural Message Passing for Quantum Chemistry / Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, George E. Dahl // ICML'17, V. 70, 2017, P. 1263–1272

8. Graph Neural Networks: A Review of Methods and Applications / Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, Maosong Sun // AI Open, V. 1, 2020, P. 57-81

9. Is Machine Learning Ready for Traffic Engineering Optimization? / Guillermo Bernárdez, José Suárez-Varela, Albert López, Bo Wu, Shihan Xiao, Xiangle Cheng, Pere Barlet-Ros, Albert Cabellos-Aparicio // IEEE ICNP, 2021

10. Communication-Efficient Policy Gradient Methods for Distributed Reinforcement Learning / Tianyi Chen, Kaiqing Zhang, Georgios B. Giannakis, Tamer Başar // IEEE Transactions on Control of Network Systems, V. 9, Is. 2, 2022, P. 917 - 929

11. A Communication-Efficient Multi-Agent Actor-Critic Algorithm for Distributed Reinforcement Learning / Yixuan Lin, Kaiqing

Zhang, Zhuoran Yang, Zhaoran Wang, Tamer Başar, Romeil Sandhu, Ji Liu // 2019 IEEE 58th Conference on Decision and Control (CDC), P. 5562 - 5567

12. Woojun Kim, Myungsik Cho, Youngchul Sung. Message-Dropout: An Efficient Training Method for Multi-Agent Deep Reinforcement Learning // The 33rd AAAI Conference on Artificial Intelligence (AAAI) 2019, P. 6079-6086

13. Sai Qian Zhang, Qi Zhang, Jieyu Lin. Efficient Communication in Multi-Agent Reinforcement Learning via Variance Based Control // Advances in Neural Information Processing Systems 32 (NeurIPS 2019)

14. Dingyang Chen, Yile Li, Qi Zhang O. Communication-Efficient Actor-Critic Methods for Homogeneous Markov Games // International Conference on Learning Representations, 2022

Королёв Л.Н.

Генетические алгоритмы с формальной точки зрения

Введение

Вычислительный процесс поиска оптимального решения некоторой задачи, именуемый "генетическим" алгоритмом, имитирует механизмы естественного отбора. Грубая модель механизма естественного отбора основана на принципе выживания в популяции особей, наиболее приспособленных к среде обитания. Выжившие особи генерируют новую популяцию, особи которой лучше приспособлены к выживанию, чем особи предыдущих популяций. При неизменных характеристиках внешней среды обитания непрерывный процесс улучшения степени приспособленности особей к выживанию идёт по нарастающей, и с течением времени могут появиться особи, степень выживаемости которых достигает максимума.

Для того, чтобы процесс получения идеальной особи работал, необходимо, чтобы работали механизмы изменчивости, селекции и скрещивания, чтобы от хорошо приспособленных "родителей" с большой вероятностью появлялись ещё лучше приспособленные "дети".

В генетических алгоритмах сохранена "биологическая" терминология [3]. Вводится понятие "функции выживаемости" (fitness function), которая характеризует степень выживаемости конкретной особи. Вводится понятие операции "скрещивания", которая имитирует появление новой особи с характеристиками, унаследованными от родителей, участвующих в этой операции. Вводится понятие операции мутации, генерирующую новую особь с изменёнными генетическими характеристиками, которая имитирует механизмы изменчивости. Вводится понятие операции селекции, определяющей особи с плохими генетическими характеристиками, которые не должны повториться в новой популяции, что похоже на процесс вымирания особей с плохой выживаемостью.

Любой объект живой и неживой природы может быть описан перечислением присущих ему характеристик. Какой бы по смыслу не была характеристика, её можно задать либо числом, либо словом (фразой) над каким-либо алфавитом. Как известно, любое слово, составленное из букв упорядоченного алфавита, можно интерпретировать как число, и любое число, состоящее из

конечного числа цифр, можно интерпретировать как слово над алфавитом знаков цифр.

Иногда удобно считать, что перечень характеристик любого рассматриваемого объекта представляет собой упорядоченную последовательность слов над алфавитами, сопряжёнными с соответствующими характеристиками.

Пусть, например, некоторый объект описывается тремя характеристиками: весом, размером и качеством. При этом вес задаётся вещественным числом, размер – терминами "большой", "средний", "малый", а качество словами: "высокое", "низкое". В этом примере первая характеристика есть слово над алфавитом цифр от 0 до 9 и знака десятичной запятой, вторая и третья характеристики являются словами над алфавитом русских букв. Заметим для точности, что сами слова можно считать буквами, если иных значений эти характеристики не принимают.

В классических генетических алгоритмах последовательность характеристик называют хромосомой или геномом особи. В задачах распознавания такую последовательность называют образом. Представление характеристик особи в виде последовательности чисел позволяет перейти к привычной для математики терминологии и более точно сформулировать те проблемы, которые возникают в теории и практике применения генетических алгоритмов при решении целого ряда задач.

Основной класс задач, решаемых с применением генетических алгоритмов (ГА), – это задачи отыскания экстремума функции от многих переменных. В отличие от классической постановки той же задачи, каждая переменная может принимать значения только из области значений, определённой индивидуально для данной переменной, не обязательно совпадающей с областями значений других переменных.

В предыдущем примере для первой переменной областью значений является ограниченное множество действительных чисел, для второй переменной эта область может быть представлена тремя целыми числами, для третьей переменной эта область представима только двумя значениями – 0,1.

Процедуры вычисления значения функции учитывают структуру таких разнородных областей задания переменных, основанную на семантике признаков, выраженных числами.

Коль скоро любые признаки можно представить структурированной последовательностью числовых значений можно такую последовательность понимать как задание координат точки в n-мерном пространстве.

Множество точек, на котором определена функция F, может иметь достаточно сложную структуру, оно может быть односвязным и многосвязным, континуальным, дискретным и смешанным (дискретно-континуальным).

Для многих применений генетических алгоритмов относительно функции F заранее не делается никаких предположений о её свойствах, ни о её гладкости, ни о её непрерывности.

В каждой конкретной области применения ГА существуют вполне определённые правила построения допустимых значений характеристик рассматриваемых объектов. Например, вес крыла самолёта не может быть больше веса всего самолёта, или характеристика температуры некоторого объекта не может превосходить известной величины.

Это означает, что для каждой конкретной постановки задачи существует своя формальная грамматика, позволяющая порождать множество допустимых для данного случая описаний объектов. Сложность правил грамматики зависит от принятых правил кодирования характеристик объектов числами.

Если количество допустимых значений каждой характеристики объекта конечно, то можно отобразить многомерное множество характеристик на числовую ось и тем самым свести решение основной задачи к одномерному случаю – поиску экстремума функции одного переменного.

В большинстве случаев применения ГА так и поступают, сводя последовательность характеристик объекта к бинарной последовательности из нулей и единиц.

Большинство теоретических работ по анализу ГА основано на изучении поведения функции выживаемости F на бинарных последовательностях.

Однако, сведение многомерного к одномерному случаю, в частности, к бинарной последовательности, не всегда оказывается полезным, так как при этом может потеряться смысл исходной постановки задачи.

При применении компьютеров для численного решения любой задачи мы неизбежно переходим к дискретной модели представления реальных объектов и отношений между ними. Это позволяет, не нарушая "компьютерной" общности, перейти к рассмотрению следующей формальной модели генетических алгоритмов и механизма их работы.

1. Формальная модель генетических алгоритмов

Будем считать, что размерность дискретного пространства, в котором будет действовать ГА, предопределена и не будет меняться в процессе его работы; существует грамматика или свод правил, позволяющие генерировать допустимые координаты точек этого пространства, на множестве которых определена функция F, глобальный экстремум которой следует определить; над точками с допустимыми координатами вводится одноместная операция, отображающая исходную допустимую точку в другую с изменённой одной координатой (операция мутации), и вводятся многоместные операции, отображающие несколько исходных допустимых точек в одну или несколько новых допустимых точек (операции скрещивания); вводится понятие популяции, как ограниченного по числу набора допустимых точек.

Механизм работы ГА представляет собой итерационный процесс, на каждом шаге которого происходит вычисление новой популяции на основе применения операций мутации и скрещивания к точкам (особям) предыдущей популяции. Стратегия применения этих операций основана на алгоритме анализа значений функции выживаемости F, вычисленной для всех точек исходной популяции. При этом в популяции фиксируются наилучшие значения функции выживаемости F, в предположении, что из наилучших значений путём операций скрещивания можно получить в новой популяции точку, доставляющую приближение к искомому экстремальному F.

Операция мутации применяется главным образом для того, чтобы избежать зацикливания итераций в районе локального экстремума. Выбор стратегии построения итерационного процесса является главной задачей разработчика конкретного ГА, и определяется его изобретательностью, знанием предметной области и опытом, т.е. принадлежит к области искусства. Основными параметрами для выбора стратегии являются: вероятность применения мутации, вероятность и правила применения скрещивания, размер популяции, критерий окончания итерационного процесса.

Строгие доказательства сходимости ГА для большинства плохо формализованных задач поиска глобального экстремума чаще всего отсутствуют и иногда просто невозможны, но вычислительные эксперименты подтверждают целесообразность применения этого механизма к решению многих реальных задач.

Под плохо формализованными задачами следует понимать те, в

которых свойства функции выживаемости не могут быть заранее определены и точно специфицированы, когда малые изменения допустимых координат точки могут повлечь непредсказуемые изменения значений функции. Многие задачи из области экономики, проектирования сложных технических систем относятся к такому классу задач.

2. Формальная постановка задачи

Для иллюстрации некоторых проблем применения ГА рассмотрим формальную постановку следующей задачи. Пусть нам задан алгоритм вычисления функции F, определённой на всех двоичных строках одинаковой длинны n. Тем самым функция определена на всех вершинах n-мерного куба, на всех 2^n точках пространства.

Пусть о функции Fизвестно только то, что она принимает максимальное значение только в одной точке нашего дискретного n-мерного пространства.

Требуется найти эту точку.

Очевидно, задача имеет решение, и полным перебором эту точку можно найти. Попробуем применить механизм ГА к решению данной задачи, и будем поступать следующим образом.

Выберем размер популяции, равный $m << n$. Случайным образом сформируем двоичные строки начальной популяции: $S_1, S_2, S_3, \dots S_m$. Вычислим $F(S_i)$для $i = 1 \dots m$ и упорядочим строки в порядке убывания результатов вычислений, т.е. будем считать, что $F(S_i) > F(S_{i+1})$. Применим к нескольким "верхним" парам соответствующим образом упорядоченной последовательности строк начальной генерации операции скрещивания.

Операцию скрещивания определим, например, так: строки скрещиваемых пар разобьём на две части приблизительно одинаковой длины и сгенерируем две новых строки, компонуя выделенные части в другом порядке (см. Рис. 1).

Из оставшейся части "нижних" строк с плохими значениями функции выживаемости F выберем случайным образом несколько строк и применим к ним операцию мутации. Эта операция будет состоять в замене наугад выбранной компоненты двоичной строки на противоположное значение (1 на 0, 0 на 1).

Из вновь полученных строк сформируем новую популяцию того же размера m, отбросив строки с наиболее плохими значениями функцией выживаемости, выполнив так называемую селекцию.

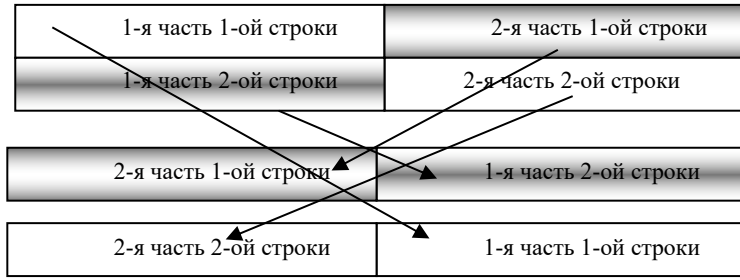

Рис. 1: Операция скрещивания

Со вновь полученной популяцией из m строк проделаем те же операции и т.д. Завершим работу генетического алгоритма, выполнив N итераций. Естественно, при этом мы запомним наибольшее значение $F(S)$, и строку, которая дала нам это значение, объявим решением задачи.

Приведённый выше алгоритм можно считать стандартным для большинства ГА. Задавшись вероятностями выполнения операций скрещивания, мутаций, параметрами m, N, n можно подсчитать вероятность того, что найденное решение соответствует действительности. Скорее всего эта вероятность p будет равна $\frac{mN}{2^n}$ и будет такой же, как в простейшем алгоритме вычисления значений $F(S_i)$ в случайно выбранных mN точках. Величина $= \frac{mN}{2^n}$, как правило, близка к 0. Из этого следует, что процедуры ГА бессмысленно применять в тех случаях, когда отсутствуют какие-либо гипотезы, характеризующие поведение функции $F(S_i)$.

3. Обзор методов

3.1 Метод схем

В работах по теоретическому анализу ГА [1,2,3] часто используют следующую гипотезу, характеризующую поведение $F(S_i)$:

Считается, что $F(S_i)$ представляет собой "чёрный ящик", устройство которого нам неизвестно, и проверить, как он работает можно только, анализируя его ответную реакцию на сравнительно небольшом числе примеров вычислений значений $F(S_i)$ в точках из ограниченного числа итераций, генерирующих популяции.

Предполагается, что появление больших значений $F(S_i)$ связано с наличием в последовательности характеристик некоторых определённых комбинаций этих характеристик. В нашем примере

некоторой комбинацией нулей и единиц.

Генетический алгоритм обязан "отгадать" эти комбинации. Идея отгадывания состоит в том, что путём сравнения кодов точек, доставляющих наибольшие значения $F(S_i)$, пытаются выделить общие части этих кодов и затем повторять их при генерации новой популяции.

Пусть, например, наибольшие значения функции $F(S_i)$ в некоторой популяции достигается на стоках:

$$\begin{aligned} S_1 &= 00110111\mathbf{0101}0010001\mathbf{0101}0\mathbf{00101}0\mathbf{1}0 \\ S_2 &= 10110000\mathbf{0101}1101001\mathbf{0001}\mathbf{0101}\mathbf{00}11 \\ S_3 &= 01001001\mathbf{0101}0100101\mathbf{0001}\mathbf{0101}\mathbf{01}000 \end{aligned} \tag{1}$$

Жирным шрифтом в этих строках отмечены совпадающие комбинации знаков. Гипотеза состоит в том что "скорее всего" хорошие значения $F(S_i)$ будут достигаться и на других строках, у которых эти комбинации на тех же самых местах сохранятся.

Это означает, что в "двоичных" генетических алгоритмах максимальное значение $F(S_i)$ следует искать в соответствующем подпространстве общего пространства из 2^n точек. Это должно уменьшить перебор, и сделать его направленным. Процедура построения популяций, с использованием этой гипотезы, получила название механизма "схем" или теории схем в генетических алгоритмах [1,2].

По своему существу, механизм схем является попыткой построения логической формулы по не полно заданной таблице истинности. От одной итерации к другой искомая логическая формула уточняется. Это можно так же трактовать, как проблему построения автомата по результатам его ответов на ограниченном множестве внешних сигналов.

Если не применять специальных алгоритмов типа мутации, механизм схем в чистом виде может привести к нахождению только локального максимума (экстремума). Алгоритмы, позволяющие "уходить" из локального экстремума, основаны на выборе параметров вероятности мутаций "плохих" с точки зрения схем строк. Для этого при генерации новой популяции в неё вводятся так же случайно выбранные, возможно и плохие, точки.

3.2 Метод подстановок

Другой подход, опирающийся на ту же гипотезу существования некоторой формулы взаимосвязи характеристик объекта (строки S_i), дающих искомый экстремум функции, состоит в следующем.

Сведём задачу к одномерному случаю. Это всегда можно сделать в дискретном случае, создав алгоритм нумерации всех возможных 2^n точек пространства. Каждой точке исходного n-мерного пространства тем самым будет поставлена в соответствие точка на числовой оси. Отметим при этом, что вычисления значений $F(S_i)$ есть очевидный способ отображения строк S_i в точки на числовой оси.

С помощью ГА пытаются решить задачу, как, вычисляя значения $F(S_i)$ в ограниченном числе точек, определить вид строки S_i, доставляющей искомый глобальный экстремум.

В каждой итерации ГА происходит отображение строк S_i сформированной популяции в точки числовой оси. Мы можем по полученным значениям $F(S_i)$, вводить меру близости строк S_k и S_m, как модуль разности $F(S_k) - F(S_m)$, и пытаться затем "угадать" какой должна быть структура строки, доставляющей экстремум функции $F(S)$.

На следующем шаге ГА, сформировавшем очередную популяцию, гипотеза о структуре "лучшей строки" может уточняться. Проблема состоит в выборе подходящей гипотезы построения "лучшей строки". Именно к этому сводится успех применения ГА к решению конкретной задачи.

Если строка характеристик объекта представляет собой набор целых чисел и каждая характеристика может принимать только конечное число значений, то строку можно рассматривать просто, как двоичную последовательность, а поиск структуры "лучшей строки" можно интерпретировать, как поиск алгоритма нумерации строк, удовлетворяющий критерию сохранения близости строк, определённой выше.

Формально проблема поиска нумерации строк означает поиск такой перестановки 2^n целых чисел, которая превращает $F(i)$ в монотонную (например, возрастающую) функцию целочисленного аргумента. Применение ГА в этом аспекте требует вероятностного анализа степени приближения к решению основной задачи.

Простейший способ отображения точек, заданных последовательностью целочисленных координат, состоит в подсчёте взвешенной их суммы. Перестановку порядковых номеров точек можно рассматривать как изменение первоначально заданных весов.

Рассмотрим следующий алгоритм. Пусть нам задана точка P с координатами $x_1, x_2, x_3, \ldots, x_n$. Номером этой точки на числовой оси $N(P)$ мы будем считать величину:

$$N(P) = \sum_i x_i w_i \tag{2}$$

Где w_i, некоторым образом выбранные веса, удовлетворяющие, например, тому условию, чтобы у различных точек были бы различные номера (что не всегда обязательно).

Упорядочивая точки по величине значения функции выживаемости в популяциях генетического алгоритма, будем стараться подобрать веса w_i так, чтобы были выполнены следующие условия: Если $F(P_k) > F(P_m)$, то $N(P_k) > N(P_m)$

$$(F(P_k) > F(P_m) \to N(P_k) > N(P_m)), \tag{3}$$

или более сильное условие:

$$|F(P_k) - F(P_m)| \leq \lambda(k-m)|N(P_k) - N(P_m)| \tag{4}$$

где λ константа, не зависящая от номера точки на оси (λ - дискретный аналог константы Липшица).

Вычисления весовых коэффициентов w_i может быть выполнены с использованием аппарата нейронных или байесовских сетей. В этом усматривается прямая связь генетических алгоритмов и нейронных сетей.

4. Выводы

За изложенной выше схемой применения ГА и общими словами скрывается серьёзный творческий процесс создания конкретного ГА.

На основе исследования предметной области, для которой решается многопараметрическая задача оптимизации, прежде всего, следует определить алгоритм вычисления функции, экстремальное значение которой нам следует отыскать. Соответственно, следует определить перечень параметров и их кодировку, от которых во многом зависит эффективность (вычислительная сложность) алгоритма вычисления функции F. Кодируя признаки объекта (особи) числовыми значениями, мы, тем самым, формируем пространство и топологию множеств допустимых точек в нем.

Затем, нам следует разумно определить размер популяции, которую мы будем генерировать в каждом цикле работы ГА.

Далее, следует определить алгоритмы формирования новых особей в новой популяции. Здесь может быть придумано много новых или модификаций стандартных методов мутации и скрещивания.

Операцию мутации можно рассматривать, как переход к новой случайно выбранной точке пространства как-то связанной с исходной. Скрещивание можно интерпретировать, как операцию, аналогичную шагу градиентного спуска в направлении искомого минимума. Слова "можно" следует трактовать, как то, что интерпретировать можно и по другому – все зависит от конкретной задачи и изобретательности исследователя-разработчика.

Литература

1. V. Srinvas, L.M. Pathaik. *Genetic Algorithms, a survey.* J. Computer. June 1994, IEEE press.

2. J.L. Ribeiro and oth. *Genetic Algorithm. Programming Environments.* J. Computer. June 1994, IEEE press.

3. Stephanie Forrest. *Genetic Algorithm. Principles of Natural Selection Applied to Computation.* J. Sciense, v. 261 August 1993.

Аннотации

Балашов В. В. Организация обмена данными в реальном времени на базе программно-конфигурируемых сетей // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В работе предложен подход к организации обмена данными в управляющих системах реального времени. Подход основан на применении программно-конфигурируемых сетей с поддержкой виртуальных каналов и позволяет использовать подходы к обоснованию соблюдения требований реального времени к передаче данных через сеть, разработанные для сетей AFDX. При этом предложенный подход обладает большими, чем в сетях AFDX и FC-AE-ASM-RT (также использующих виртуальные каналы), возможностями по реконфигурации сети в процессе ее функционирования.

Апробация подхода в виртуальной сетевой среде подтвердила соблюдение требований реального времени в ходе передачи данных, в т.ч. в процессе реконфигурации сети — для тех виртуальных каналов, которые сохраняют маршруты и другие параметры.

Ил.: 2 рис., Библиогр.: 21.

Бахмуров А. Г., Голубкова М. С. Разработка Android приложения, имитирующего датчики интернета вещей // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

Данная статья описывает создание Android приложения, имитирующего поведение датчиков интернета вещей в части передачи данных посредством Bluetooth Low Energy (BLE). Дано краткое описание взаимодействия устройств Интернета вещей по Bluetooth LE. Обоснована необходимость создания указанного приложения. Приведены текущие результаты: разработан пользовательский интерфейс, реализованы работа с BLE, генерация данных, их передача и запись этих данных в файл.

Полученное приложение успешно прошло тестирование и было использовано для исследования масштабируемости клиентской части сервиса сбора и обработки медицинской телеметрии.

Ил.: 2 рис., Библиогр.: 22.

Британков Е. С., Пашков В. Н. Применение метода опорных векторов для обнаружения DoS-атак на контроллер в программно-конфигурируемых сетях // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В работе рассматривается проблема обнаружения DoS-атак на контроллер – ключевого элемента управления в программно-конфигурируемых сетях. Приводится краткий анализ существующих подходов к обнаружению DoS- и DDoS-атак на контроллер в ПКС с использованием методов машинного обучения. В работе предлагается подход на основе метода опорных векторов для обнаружения DoS-атаки на контроллер, приводятся результаты экспериментального исследования разработанного метода. Приложения для сбора и обработки данных, например, в сетях Интернета вещей, могут собирать данные с большого количества (сотни, тысячи) клиентов одновременно.

Ил.: 7 рис. Библиогр.: 12.

Загайнов Д. К., Курячий Г. В., Волканов Д. Ю. Построение и исследования графа python3-зависимостей в репозитории Sisyphus // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В работе рассматриваются требуемые и предоставляемые зависимости **python3**-пакетов, а также выделение подпакета из исходного **python3**-пакета. В ходе исследований был разработан набор инструментов для определения требуемых и предоставляемых зависимостей **python3**-пакета, а также построения, исследования и разбиения графа зависимостей **python3**-пакета. В качестве исследуемых объектов используются **python3**-пакеты репозитория **Sisyphus** дистрибутива ALT Linux.

Ил.: 1 табл. Библиогр.: 5.

Писковский В. О., Сагалевич В. Д. Исследование зависимости характеристик работы виртуальной машины в режиме расширенного фильтра записи от способа хранения и протокола доступа к данным виртуального диска // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

Рабочая инфраструктура централизованной вычислительной структуры предприятия может содержать репозитарий контейнеров с виртуальными машинами, реализующими рабочие места сотрудников и прошедшими принятую на предприятии процедуру внедрения в постоянную эксплуатацию. В работе исследуется зависимость времени запуска виртуальной машины в режиме расширенного фильтра записи от способа хранения и протокола доступа к данным виртуального диска.

Ил.: 5 рис., Библиогр.: 7.

Писковский В. О., Шибаев П. П. Блокчейн-хранилище для контроллера ПКС Runos: концепция и архитектура // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В статье исследуется использование блокчейн-хранилища данных на основе Cosmos SDK для приложения L2 learning switch ПКС-контроллера Runos. Описываются экспериментальный стенд, приводятся результаты исследований и обозначаются перспективы дальнейших работ в этой области.

Ил.: 2 рис., 1 табл. Библиогр.: 7.

Рязанов А. М., Волканов Д. Ю., Цыганов Н. И. Методы сбора и хранения сетевого трафика для решения задачи прогнозирования качества гетерогенных каналов в сетях передачи данных // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В работе рассматривается задача прогнозирования качества гетерогенных каналов в сетях передачи данных. Рассматриваются количественные параметры КО, методы сбора и хранения сетевого трафика, а также предлагается архитектура средств сбора, хранения и анализа сетевого трафика.

Ил.: 1 рис., Библиогр.: 6.

Тюшев М. В., Смелянский Р. Л. Об использовании графовых баз данных для маршрутизации в программно-конфигурируемых сетях // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В данной работе рассматривается альтернативный подход к решению задачи маршрутизации трафика в программно-конфигурируемых сетях. Исследуются различные реализации графовых СУБД. Выбранная графовая СУБД Neo4j используется для расчета маршрутов при изменении топологии

сети. Далее Neo4j сравнивается со средством маршрутизации, используемым в ПКС контроллере RunOS.

Ил.: 12 рис., Библиогр.: 9.

Цветкова В. П., Степанов Е. П. Исследование методов эффективного общения в многоагентном обучении для задачи балансировки сетевого трафика // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В статье представлено экспериментальное исследование методов эффективного общения, которые позволяют уменьшить обмен данными между агентами для обеспечения оптимального распределения сетевого трафика. Для оптимального распределения трафика по каналам в сети рассматривается многоагентный алгоритм "Субъект-Критик" ("Actor-Critic") — распределенный алгоритм обучения с подкреплением, который состоит из обучающихся агентов: субъектов и централизованного критика. У субъекта есть свое состояние, которое он передает соседям перед каждым принятием решения, поэтому требуется частый обмен информацией между агентами. Многоагентные алгоритмы обучения с эффективным общением помогают уменьшать накладные расходы. Метод CEAC позволил сократить передаваемый объем в 5 раз, а метод LAPG уменьшил количество передаваемых сообщений в 5 раз.

Ил.: 6 рис., 1 табл. Библиогр.: 14.

Королёв Л. Н. Генетические алгоритмы с формальной точки зрения // Программные системы и инструменты. Тематический сборник № 23, М.: Изд-во факультета ВМК МГУ, 2023.

В статье рассматривается вычислительный процесс поиска оптимального решения некоторой задачи, именуемый «генетическим» алгоритмом, которая имитирует механизмы естественного отбора. Схема такого алгоритма основана на принципе выживания в популяции особей, наиболее приспособленных к среде обитания. В работе рассматриваются этапы создания генетического алгоритма для конкретной оптимизационной задачи.

Ил.: 1 рис., Библиогр.: 3.

О пятой международной научной-технической конференции "Современные сетевые технологии" (МоНеТес-2024)

5-я Международная научно-техническая конференция "Современные сетевые технологи" собирает представителей международного научного сообщества, исследовательских подразделений корпораций, стартапов, промышленности и бизнеса, институтов развития и органов государственной власти для обсуждения перспективных и актуальных технологий в сфере компьютерных сетей, виртуализации сетевых ресурсов и облачных вычислений, использования методов искусственного интеллекта.

Технологии передачи данных являются основой современной цивилизации. Области телекоммуникации вбирают в себя и постоянно порождают все новые и новые технологии, которые открывают новые возможности, повышают качество сервиса и безопасность в современных сетях. Технологии программного управления в сетях, виртуализации сервисов, периферийные облачные вычисления стали ключевыми элементами построения современных сетей передачи данных и информационных инфраструктур в целом. В настоящее время в мире (и в России, в частности) начато их применение на практике. Однако творческая мысль не останавливается на достигнутом. Сегодня мы уже говорим о реконфигурируемых по требованию сетях (Intent Based Network), информационно-ориентированных Сетях (Information Centric Network), контент-ориентированных сетях (Content Centric Network). Возникает много новых проблем и направлений для исследований.

На конференции планируется выступление с пленарными докладами ряда зарубежных и отечественных ученых по перспективным направлениям развития современных сетей передачи данных и их приложений. Программа конференции также

предусматривает проведение нескольких школ по сетевым технологиям и применению отечественных решений по тематике конференции для молодых ученых, студентов старших курсов и аспирантов. Это будет способствовать расширению профессионального круга специалистов, способных поддерживать и развивать эти технологии и решения.

Направления работы MoNeTec-2024

- QoS control in Data Communication
- Resource Management and Control in Cloud Computing
- Edge Computing
- Information Security in SDN/Cloud
- 5G/6G Networks for Wireless Communication
- 6G Radio Access Networks
- Coding Theory Applications in Networking
- High-speed routing and switching
- Heterogeneous Channel Traffic modeling and analysis
- Large-scale network simulation: Methods and Tools
- Formal verification of network protocols and service
- AI-Driven IoT Sensing, Interaction, and Digitalization
- IIoT: Industrial Internet of Things
- Domain Specific Networks
- AI4net, AI for network and network for AI
- AIoT: Artificial Intelligence of Things
- Future Networking

Программный комитет приветствует подачу докладов, посвящённых применению методов искусственного интеллекта в указанных выше направлениях.

Программный комитет

В Программный комитет входят 36 ученых из четырёх стран, из них 17 состоят в IEEE. Список членов Программного комитета MoNeTec-2024 доступен на официальном сайте по ссылке: https://monetec.ru/committee

Организационный комитет

Список членов Организационного комитета MoNeTec-2024 доступен на официальном сайте по ссылке: https://monetec.ru/organizing_committee/
Контакты Организационного комитета:

- e-mail: info@monetec.ru
- тел: +7 (495) 9394671

Подача докладов

Доклад должен представлять собой оригинальный, ранее не опубликованный результат. Информация о требованиях к докладам и процедуре подачи докладов размещена на сайте конференции https://www.monetec.ru. Материалы для публикации - доклады объемом до 12 страниц в формате pdf на английском языке представляются через систему uConfy. Подробная информация о типах докладов представлена на сайте.

Важные даты

- Представление аннотаций (extended abstract) докладов: до 1 мая 2024 г.
- Результаты предварительного рецензирования: до 15 мая 2024 г.
- Представление докладов: до 15 июня 2024 г.
- Результаты рецензирования: до 1 сентября 2024 г.
- Предоставление финальной версии доклада, доработанного по результатам рецензирования: до 20 сентября 2024 г.

- Регистрация для участия в Школе: до 5 октября 2024 г.
- Школы: 27-28 октября 2024 г.
- Конференция: 29-31 октября 2024 г.

Формат и место проведения

Формат конференции: смешанный (очный и дистанционный).
Место проведения: Московский государственный университет имени М.В. Ломоносова

Приглашаем Вас к участию в конференции MoNeTec-2024.

Software systems and tools: Thematic collection / Edited by Smelianskii R.L. – M: Publishing Department of the Faculty of CMC Lomonosov Moscow State University (license ID №05899 from 24.09.2001y.; MAKS Press, 2023, № 23. – 140 p.

ISBN 978-5-89407-638-6 (CS MSU)
ISBN 978-5-317-07118-9 (MAKS Press)
https://doi.org/10.29003/m3791.978-5-317-07118-9

This volume contains student works that were recommended for publication by the scientific seminar of Automation of Computer Systems chair. The editorial board continues the publishing tradition in memory of L.N. Korolev. The proposed thematic collection contains papers on the actual problems of computer networks, methods of computations scheduling and load balancing, discrete optimization problems and algorithms, network traffic analysis.

These papers are of interest to students, graduate students and professionals in the development of applied software systems.

Keywords: Communication networks, software-defined networks, Internet of Things, machine learning, support vector machine, DoS attacks, SDN controller, virtual machine, blockchain, network traffic analysis, quality of service, heterogeneous channels, graph databases, traffic routing, multi-agent learning, load balancing, medical data collection, multi-agent systems, genetic algorithms.

Научное издание

ПРОГРАММНЫЕ СИСТЕМЫ И ИНСТРУМЕНТЫ

Тематический сборник

№ 23

Под общей редакцией чл.-корр. РАН,
профессора Р. Л. Смелянского

Издательство «МАКС Пресс»
Главный редактор: *Е. М. Бугачева*

Напечатано с готового оригинал-макета
Подписано в печать 20.10.2023 г.
Формат 60x90 1/16. Усл.печ.л. 8,75.
Тираж 40 экз. Заказ 211.

Издательство ООО “МАКС Пресс”
Лицензия ИД N 00510 от 01.12.99 г.
119992, ГСП-2, Москва, Ленинские горы, МГУ им. М.В. Ломоносова,
2-й учебный корпус, 527 к.
Тел. 8(495)939-3890/91. Тел./Факс 8(495)939-3891.

Отпечатано в полном соответствии с качеством
предоставленных материалов в ООО «Фотоэксперт»
109316, г. Москва, Волгоградский проспект, д. 42,
корп. 5, эт. 1, пом. I, ком. 6.3-23Н