Санкт-Петербургский Государственный Университет

Факультет прикладной математики - процессов управления

Голубева Мария Александровна

Выпускная квалификационная работа

О совместном использовании узлов децентрализованной беспроводной самоорганизующейся сети

Направление 02.03.02

Фундаментальная информатика и информационные технологии

Научный руководитель:
доктор физико-математических
наук, профессор Громова Е. В.

Рецензент:
ассистент Шувалов Г. М.

Санкт-Петербург,

2020

Оглавление

Оглавление 2

1. Введение 4

Постановка задачи 5

Обзор литературы 6

2. Беспроводные децентрализованные самоорганизующиеся сети MANET 8

2.1 Определение 8

2.2 Применение 8

2.3 Особенности 10

2.4 Преимущества и недостатки сетей MANET 12

2.5 Протоколы и алгоритмы маршрутизации 13

3. Симулятор NS-3 16

3.1 Определение 16

3.2 Особенности программного обеспечения NS-3 16

4. Симуляция сети с движущимися узлами 18

4.1 Постановка задачи для моделирования 18

4.2 Способы визуализации 18

4.3 Проведение симуляции 19

5. Проверка оптимальности сети 22

5.1 Постановка задачи для моделирования 22

5.2 Начальные условия 23

5.3 Переменные для анализа 23

5.4 Проведение симуляции 24

5.5 Анализ сети 25

5.6 Результаты исследования симуляции 29
6. Заключение 31
Список литературы 32
7. Приложения 34
Приложение 1. Таблица потоков без узла 34
Приложение 2. Таблица потоков с узлом 37
Приложение 3. Исходный код программы для симуляции сети с движущимися узлами 40
Приложение 4. Библиотека для кластеризации узлов 46
Приложение 5. Исходный код программы симуляции для задачи о проверке оптимальности сети 51

1. Введение

Современный мир, столкнувшийся с огромным ростом производительности технологий, становится все более взаимосвязанным, оцифрованным, распределенным и разнообразным. Благодаря тому, что большое множество вещей обладает способностью обрабатывать данные, вычислительные модели готовы значительно расшириться и объединиться в сеть. И по мере добавления к нему устройств и пользователей ценность и важность сети продолжает расти в геометрической прогрессии.

Однако не всегда при организации сетей имеется необходимость задействовать глобальную сеть, или сеть Интернет. Для множества задач требуется более узконаправленное решение, и необходимости в обустройстве крупной и сложной сети попросту нет, но есть другая необходимость – в короткие сроки построить сеть для обмена информацией на местности, где провести другие коммуникации бывает довольно сложно. Данная проблема уже давно и успешно решается путем организации сетей MANET.

Постановка задачи

Основной целью работы являлось изучение механизмов численной симуляции работы децентрализованных самоорганизующихся сетей MANET. В связи с этим можно выделить следующие основные задачи:

1. Подробно изучить архитектуру сетей типа MANET и способы оптимального использования данной технологии.
2. Изучить один из способов симуляции различных сетей, в частности MANET-сетей, представленный программным обеспечением Network Simulator-3 (NS-3).
3. Провести симуляцию в среде NS-3 для сети с узлами, движущимися в случайном направлении. Разработать кластеризацию исходного положения узлов, визуализировать полученную сеть и проанализировать работу полученной модели.
4. Провести пробную симуляцию сетей MANET, в которые добавлены дополнительные узлы с целью улучшения характеристик сети, проанализировать характеристики сети до и после добавления узлов.

Обзор литературы

При написании работы была использована следующая литература:

1. Christian Makaya, Samuel Pierre, Emerging Wireless Networks: Concepts, Techniques and Applications.

   В данном сборнике исследовательских работ и обзоров рассматриваются последние разработки в технологиях беспроводных сетей следующего поколения (Next-Generation Wireless Networks - NGWN) и мобильных широкополосных сетей, включая 4G, 3G, Wi-Fi, беспроводные децентрализованные сети, ячеистые сети и беспроводные сенсорные сети.

2. Amjad Umar, Mobile Computing and Wireless Communications: Applications, Networks, Platforms, Architectures, and Security.

   Эта книга представляет собой всеобъемлющий охват технических аспектов мобильных вычислений и беспроводной связи. Вместо одной узкой темы в книге рассматриваются основные строительные блоки (мобильные приложения, мобильные вычислительные платформы, беспроводные сети, архитектуры, безопасность и управление) мобильных вычислений и беспроводной связи.

3. Fahad Taha AL-Dhief, Naseer Sabri, M.S. Salim, S. Fouad, S. A. Aljunid, MANET Routing Protocols Evaluation: AODV, DSR and DSDV Perspective.

   В этой статье представлено сравнение производительности между реактивными протоколами маршрутизации, представленными динамической маршрутизацией от источника (DSR - Dynamic Source Routing) и специальной векторной маршрутизацией по требованию (AODV - Ad hoc on demand distance Vector Routing), и

проактивным протоколом маршрутизации, представленным алгоритмом вектора последовательного расстояния до места назначения (DSDV - Destination Sequenced Distance Vector), для точного определения того, какой протокол более эффективен.

4. S. Blakeway, D. V. Gromov, E. V. Gromova, A. S. Kirpichnikova, T. M. Plekhanova, “Increasing the performance of a Mobile Ad-hoc Network using a game-theoretic approach to drone positioning”, Вестн. С.-Петербург. ун-та. Сер. 10. Прикл. матем. Информ. Проц. упр., 15:1 (2019), 22–38

   В данной статье рассматривается задача размещения оптимальных узлов для децентрализованных самоорганизующихся сетей MANET. Авторы решают поставленную проблему в теоретико-игровой форме, рассматривая несколько случаев возможных сценариев. Также был предложен алгоритм, который повышает производительность сети путем установки дополнительных узлов. Программа генерирует различные игровые ситуации и определяет ход каждого игрока, решая соответствующие задачи оптимизации.

2. Беспроводные децентрализованные самоорганизующиеся сети MANET

2.1 Определение

MANET (Mobile Ad-hoc Network) – беспроводная самоорганизующаяся децентрализованная сеть, которая состоит из мобильных узлов, способных устанавливать и поддерживать соединения между узлами. Иногда такие сети называют сетями «на лету» или «спонтанными сетями». Сеть не зависит от ранее существующей инфраструктуры, такой как маршрутизаторы в проводных сетях или точки доступа в управляемых (инфраструктурных) беспроводных сетях. Вместо этого каждый узел участвует в маршрутизации путем пересылки данных для других узлов, поэтому определение, какие узлы пересылают данные, производится динамически на основе сетевого подключения и используемого алгоритма маршрутизации.

MANET также определяется как автономная система мобильных маршрутизаторов (и связанных хостов), соединенных беспроводными линиями, которые устанавливают связь друг с другом. Маршрутизаторы могут свободно перемещаться и организовываться произвольно: таким образом, беспроводная топология сети может изменяться быстро и непредсказуемо.

2.2 Применение

MANET – широко распространенная технология во многих исследовательских и правительственных проектах. Однако самые известные технологии, базирующиеся на MANET, – это Bluetooth и Jini. Идея MANET может быть использована и реализована в совершенно различных направлениях. Такая сеть может быть установлена, например, между домашним компьютером и разными устройствами, подключаемыми к нему, такими как беспроводные мыши, клавиатуры, принтеры и другое. MANET может использоваться при передаче информации между несколькими

ноутбуками, или даже при организации спасательной операции, при необходимости использовать одни датчики разными службами спасения.

Множество примеров использования MANET можно найти в реальной жизни, в ситуациях, когда необходимо развернуть сеть на местности, где для этого не создано никакой инфраструктуры. Например, при проведении спасательных операций нет времени для установки точек доступа, и тогда единственное решение – это развертывание сети MANET.

Типичные примеры MANET:

- При чрезвычайных ситуациях, когда, к примеру, здание было разрушено в результате пожара, землетрясения или сброса бомб. В таком случае важно настроить сеть в кратчайшее время, и сети на основе MANET идеально подходят для таких ситуаций. Так, в таких ситуациях полиция и пожарные могут передавать сообщения через MANET и выполнять свои операции без надлежащего покрытия беспроводной сети.
- При необходимости группового взаимодействия, как, например, настройка связи на выставках, конференциях, презентациях, собраниях и лекциях, где точки доступа могут не существовать заранее. Еще один пример – это групповые встречи, когда возникает необходимость разным членам команды общаться друг с другом, находясь за пределами офиса. Кроме того, подключение мобильных телефонов к ноутбукам и общественным сетям также является важным приложением данной технологии в повседневной жизни.
- Сенсорные устройства могут формировать мобильную ad-hoc сеть интеллектуальных датчиков (MANIS – Mobile Ad-hoc Network of Intelligent Sensors) для особых ситуаций. MANIS может адаптироваться практически к любому операционному

развертыванию благодаря гибкости, предлагаемой мобильными ad-hoc сетями.

2.3 Особенности

К настоящему времени должно стать очевидным, что сети MANET сильно отличаются от традиционных сетей, проводных или беспроводных. На рисунке ниже показана типичная конфигурация MANET, где мобильные узлы A, B, C, D и E образуют специальную сеть. Маршрутизатор Intemet также может участвовать в этой сети для передачи информации на корпоративный сайт или в центр управления.

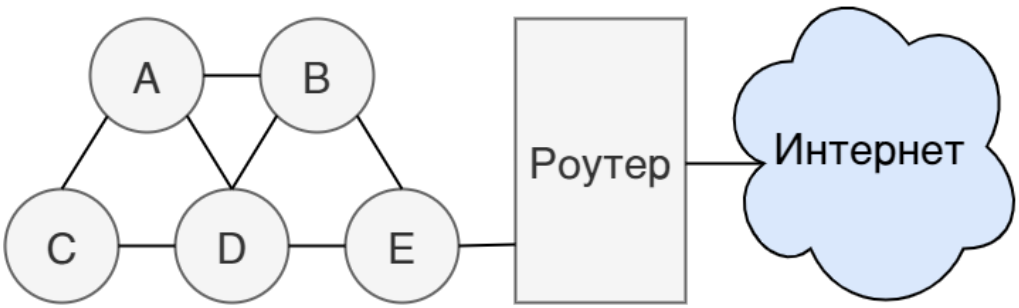


Рисунок 1. Общий алгоритм MANET сетей

Ключевыми характеристиками MANET являются:

- Отсутствие фиксированных компонентов инфраструктуры, таких как точки доступа или базовые станции. В MANET два или более устройства оснащены возможностями беспроводной связи и работы в сети. Такие устройства могут обмениваться данными с другим узлом, который находится непосредственно в пределах своего радиодиапазона (одноранговая связь), или тем, который находится за пределами своего радиодиапазона, с использованием промежуточного узла(ов) для ретрансляции пакетов от источника к месту назначения.
- Работа сети неразрывно связана с изменениями маршрута, и источникам, возможно, придется проходить несколько различных

соединений, чтобы каждый раз достигать пунктов назначения, потому что все узлы имеют возможность двигаться. Из-за этого традиционные протоколы маршрутизации терпят неудачу, потому что они предполагают фиксированную топологию сети.

- Самоорганизация и возможность к приспособлению. Это означает, что сформированная сеть может быть сформирована на лету без необходимости какого-либо системного администрирования. Это позволяет быстро развертывать сети, когда это необходимо, и быстро отключать, когда необходимость в сети пропадает.
- Сеть может состоять из разнородных устройств. Узлы могут быть разных типов (КПК, ноутбуки, мобильные телефоны, маршрутизаторы, принтеры и т. д.) с различными возможностями вычисления, хранения и связи. Единственное требование состоит в том, чтобы базовое программное обеспечение MANET могло работать на устройствах.
- Потребляемая мощность может быть довольно высокой, поскольку узлы должны поддерживаться в рабочем состоянии для пересылки пакетов данных, отправляемых другими узлами, которые случайно оказались в окрестности. Это является особенно большой проблемой для небольших датчиков.

Есть несколько следствий этих особенностей сети. Во-первых, традиционные интернет-протоколы работают не так хорошо, потому что Интернет предполагает, что его подключение и топология будут меняться очень медленно с течением времени. Именно поэтому интернет-протоколы оптимизированы для сетей с надежной связью между узлами. В среде MANET интернет-протоколы не работают, потому что они не могут справиться с быстрыми колебаниями соединений между узлами. Для MANET необходимы

новые протоколы маршрутизации, которые могут обрабатывать изменения топологии. Такие протоколы будут рассмотрены далее.

2.4 Преимущества и недостатки сетей MANET

Мобильная ad-hoc сеть имеет ряд преимуществ перед традиционными беспроводными сетями, в том числе:

- Простота и скорость развертывания – беспроводная локальная сеть развертывается в очень ограниченное время.
- Минимальные затраты, т.к. отсутствует необходимость в дорогой инфраструктуре для развертывания сети.
- Отказоустойчивость, т.к. существует временный резервный механизм на случай, если обычно доступные устройства инфраструктуры (точки доступа или маршрутизаторы) перестают функционировать.

Благодаря этим преимуществам MANET широко используется в сферах, упомянутых ранее (ликвидация чрезвычайных ситуаций, организация групповых встреч, выставок, конференций, презентаций и др.). Тем не менее, сети MANET сталкиваются с некоторыми проблемами, такими как следующие:

- Маршрутизация. Как упоминалось ранее, топология сети изменяется случайным образом со временем. Таким образом, протокол маршрутизации должен часто обновлять маршруты и ссылки. Маршрутизация также может быть скорректирована, чтобы справиться с потерей радиолиний и мобильных устройств.
- Безопасность: MANET не использует централизованный административный объект для обеспечения безопасности и аутентификации пользователей. Это означает, что любая станция в радиусе действия может подключаться к другим станциям, настроенным для работы с сетью ad-hoc. Если станция

подключена к корпоративной сети и настроена так, чтобы разрешить сетевое соединение, потенциальный взломщик может атаковать станцию и получить доступ к сети. Весь трафик от взломщика появляется в проводной сети как исходящий от авторизованной станции.

- Управление питанием: узлы MANET питаются от легких батарей, которые имеют ограниченный срок службы и, таким образом, накладывают ограничения на дальность передачи, коммуникационную активность и вычислительные возможности этих узлов.
- “Эгоистичные узлы”: так как мощность играет весомую роль, некоторые узлы могут стать “эгоистичными” и отказаться от маршрутизации пакетов других узлов. Для выявления и исправления таких ситуаций необходимо применять сложные меры. Например, эгоистичные узлы можно регулировать путем отказа в пересылке их пакетов другими узлами.
- Отказоустойчивость и качество обслуживания: MANET должен обеспечивать отказоустойчивость и гарантировать высокое качество обслуживания в очень тяжелых ситуациях. Например, изменяющиеся свойства физической линии связи затрудняют обеспечение минимального уровня обслуживания.

2.5 Протоколы и алгоритмы маршрутизации

Как известно, алгоритмы маршрутизации определяют оптимальный путь между отправителями и получателями на основе определенных метрик, таких как кратчайшая задержка или минимальная стоимость (см. Рисунок 2). Определение оптимальных маршрутов в больших сетях было областью активного исследования на протяжении многих лет с приложениями для коммивояжеров, маршрутов школьных автобусов, маршрутов рейсов и других. Важным фактором при разработке алгоритма маршрутизации является

время T, необходимое для построения пути маршрута. Если T больше среднего времени между изменениями топологии, алгоритм не может достаточно быстро обновить таблицу маршрутизации. Например, если топология меняется каждые 20 секунд, но для поиска маршрута требуется минута, то в таблицах маршрутизации не будет правильной информации о маршрутизации, и вся система маршрутизации не сможет корректно работать. Эта проблема является основной проблемой в маршрутизации MANET. В сетях MANET стандартные алгоритмы маршрутизации Intemet не работают должным образом, поскольку они предполагают, что топология будет меняться очень редко, поэтому предполагается, что на поиск оптимального пути имеется неограниченное количество времени.

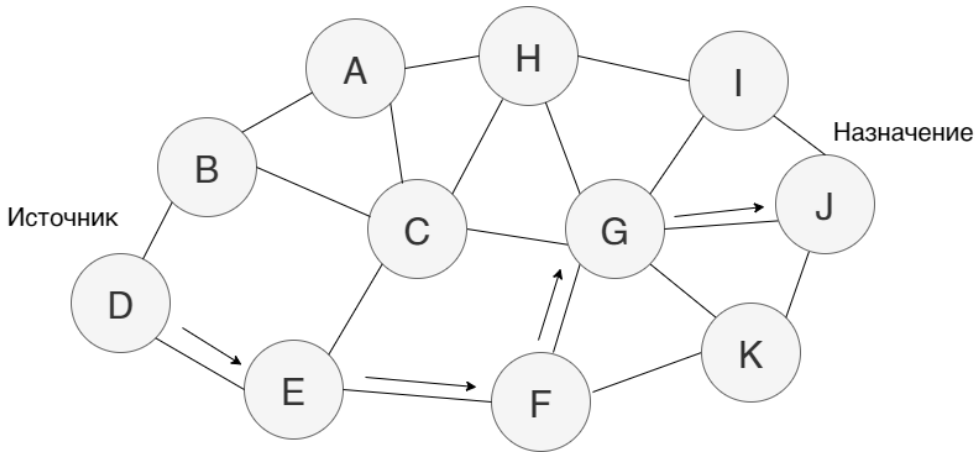


Рисунок 2. Алгоритм маршрутизации

Для мобильных одноранговых сетей основные функции маршрутизации включают в себя:

- Генерация возможных путей между отправителями и получателями;
- Определение подходящих путей на основе критерия выбора (например, минимальное время);
- Передача трафика пользователя по выбранным путям;
- Проверка, что выбранный маршрут поддерживается, и поиск альтернативы в случае проблем.

Протоколы маршрутизации для сетей MANET должны быть высоко адаптивными, быстрыми и экономичными относительно потребления энергии.

В MANET существует три основных типа протоколов маршрутизации: проактивный (Proactive), реактивный (Reactive) и гибридный (Hybrid). На рисунке 3 показана классификация протоколов маршрутизации. В проактивных протоколах узлы имеют таблицу информации о маршрутизации и узел создает маршруты, прежде чем в них возникает необходимость. Поэтому операция обнаружения маршрута реализуется быстрее, чем при использовании реактивных протоколов. Узлы в реактивных протоколах создают маршруты тогда, когда возникает необходимость в передаче данных соседнему узлу. Узлы в гибридных протоколах объединяют стратегии как реактивных, так и проактивных протоколов.

Рисунок 3. Алгоритмы маршрутизации MANET

3. Симулятор NS-3

3.1 Определение

В качестве среды для симуляции был выбран симулятор сетей NS-3. NS-3 – это симулятор дискретных событий, в котором каждое событие связано со временем его выполнения, и моделирование выполняется в ограниченном дискретном промежутке времени.

NS-3 был разработан, чтобы обеспечить открытую расширяемую платформу сетевого моделирования для сетевых исследований и образования. NS-3 предоставляет модели того, как работают сети пакетной передачи данных, и предоставляет пользователям механизм моделирования для проведения своих экспериментов. Этот инструмент широко используется для изучения работы больших сетей, а также как средство проверки для предлагаемых аналитических моделей и алгоритмов. Данный сетевой симулятор позволяет проводить эксперименты без необходимости разворачивать реальную сеть, что позволяет существенно сократить расходы, уменьшить трудоемкость процесса и обеспечить достаточную скорость и эффективность при организации моделирования.

Можно также отметить, что существующая модель, установленная в ns-3, фокусируется на моделировании работы интернет-протоколов и сетей, но симулятор не ограничивается интернет-системами – существует возможность использования ns-3 для моделирования неинтернетных систем, таких как поведение людей в обществе, анализ эпидемиологической обстановки, транспортной системы и др.

3.2 Особенности программного обеспечения NS-3

Существует много инструментов моделирования для изучения сетевых симуляций. Ниже приведены некоторые отличительные особенности NS-3 в отличие от других инструментов.

Симулятор NS-3 разработан как набор библиотек, которые могут быть объединены друг с другом, а также с другими внешними библиотеками программного обеспечения. В то время как некоторые платформы моделирования предоставляют пользователям единую интегрированную среду графического интерфейса пользователя, в которой выполняются все задачи, ns-3 является более модульным в этом отношении. В NS-3 могут использоваться внешние аниматоры и инструменты анализа и визуализации данных. Тем не менее, пользователи должны работать с командной строкой и с инструментами разработки программного обеспечения C ++ и /или Python.

NS-3 в основном используется в системах Linux или macOS, хотя существует поддержка для систем BSD, а также для сред Windows, которые могут создавать код Linux, таких как Windows Subsystem для Linux или Cygwin. Собственная Windows Visual Studio в настоящее время не поддерживается, хотя разработчик работает над будущей поддержкой и данной операционной системы. Для пользователей Windows есть возможность использовать виртуальную машину Linux.

4. Симуляция сети с движущимися узлами

4.1 Постановка задачи для моделирования

Для того, чтобы на практическом опыте рассмотреть работу программного обеспечения NS-3, было решено провести моделирование случайной сети, узлы которой сообщаются друг с другом по беспроводной связи. Все узлы имеют начальное случайное положение в пространстве на сетке 1500x1500 единиц, имеют случайную скорость до 5 единиц в секунду и в каждый момент времени могут выбрать случайное направление для дальнейшего перемещения.

Для того, чтобы перемещение узлов было наглядно видно, проведем кластеризацию исходных узлов по их положению на сетке методом k-means. Данный алгоритм большую получил большую популярность благодаря своей простоте, наглядности реализации и достаточно высокому качеству работы. Узлы, принадлежащие разным кластерам, раскрасим в разные цвета – для данного случая было принято решение делить все узлы на два кластера, однако алгоритм возможно применять для любого числа кластеров, изменяя входной параметр числа кластеров.

4.2 Способы визуализации

Однако базовыми средствами NS-3 не получится визуально представить полученную сеть – все результаты в первоначальном виде представлены лишь объектами классов и переменными с числами. Для того, чтобы наглядно рассмотреть полученную сеть, можно воспользоваться несколькими программными утилитами, которые позволяют визуализировать сеть, построенную в NS-3. Самыми популярными из этих утилит являются:

- NetAnim
- NS-3 PyViz

Основным отличием данных программных средств является то, что PyViz – это визуализатор в режиме реального времени, то есть он не использует файлы трассировки. Это может быть наиболее полезно для целей отладки, то есть, чтобы выяснить, соответствуют ли модели мобильности ожиданиям исследователя, где теряются пакеты и т.д. NetAnim, в свою очередь, использует xml файл трассировки для визуализации работы сети, т.е. после завершения процесса симуляции генерируется файл, который передается программе, и на его основе происходит построение визуализации. Однако, оба программных решения позволяют качественно оценить построенную сеть и проанализировать состояние узлов, переданных и полученных пакетов и многое другое.

4.3 Проведение симуляции

Итак, подготовим исходную случайную сеть с пятьюдесятью узлами. На рисунке 4 представлен скриншот визуализации этой сети в программе PyViz.

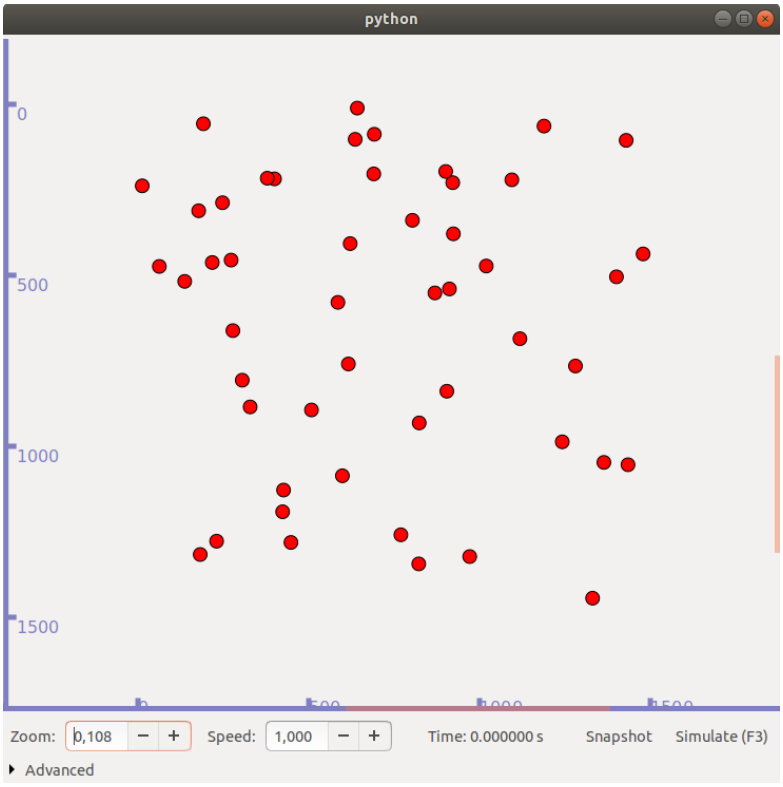


Рисунок 4. Исходная сеть

Теперь проведем кластеризацию узлов сети согласно их изначальному положению, а также найдем центры кластеров и узлы, принадлежащие каждому кластеру. Узлы, принадлежащие первому кластеру, раскрасим в красный цвет, а узлы, принадлежащие второму кластеру, – в зеленый. Т.к. PyViz не предоставляет средств для перекраски узлов, воспользуемся программным пакетом NetAnim для визуализации сети с кластеризованными узлами (см. рисунок 5).

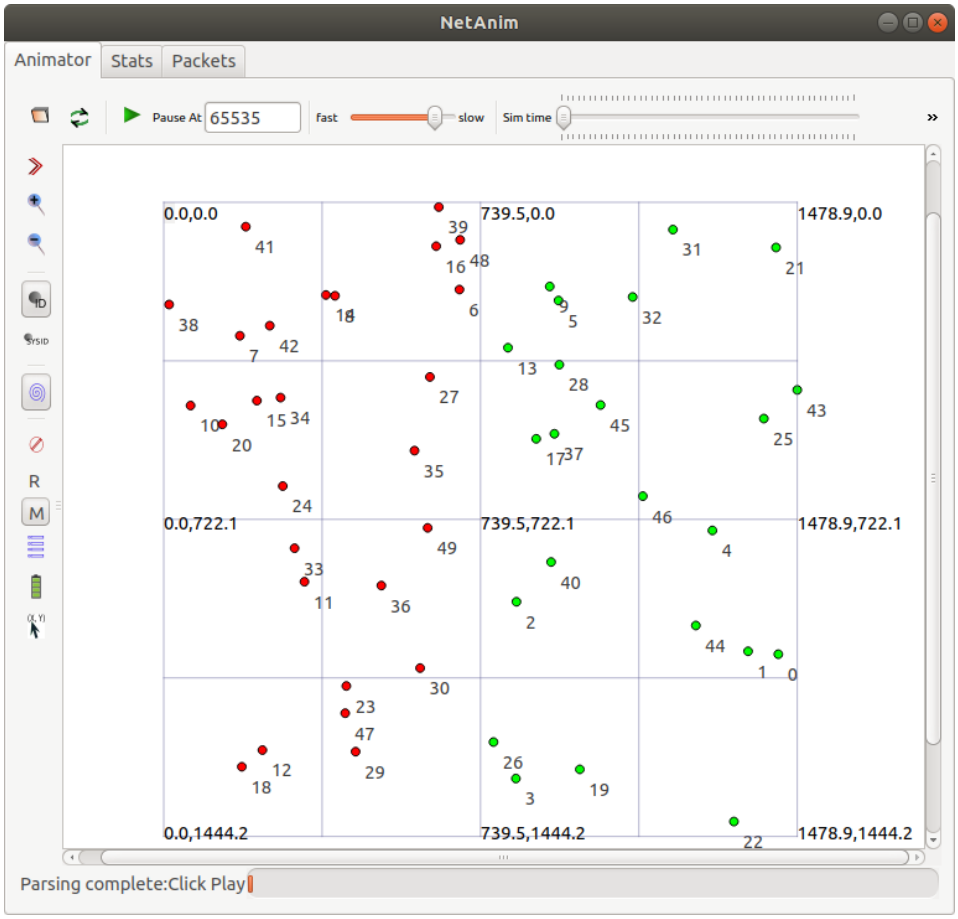
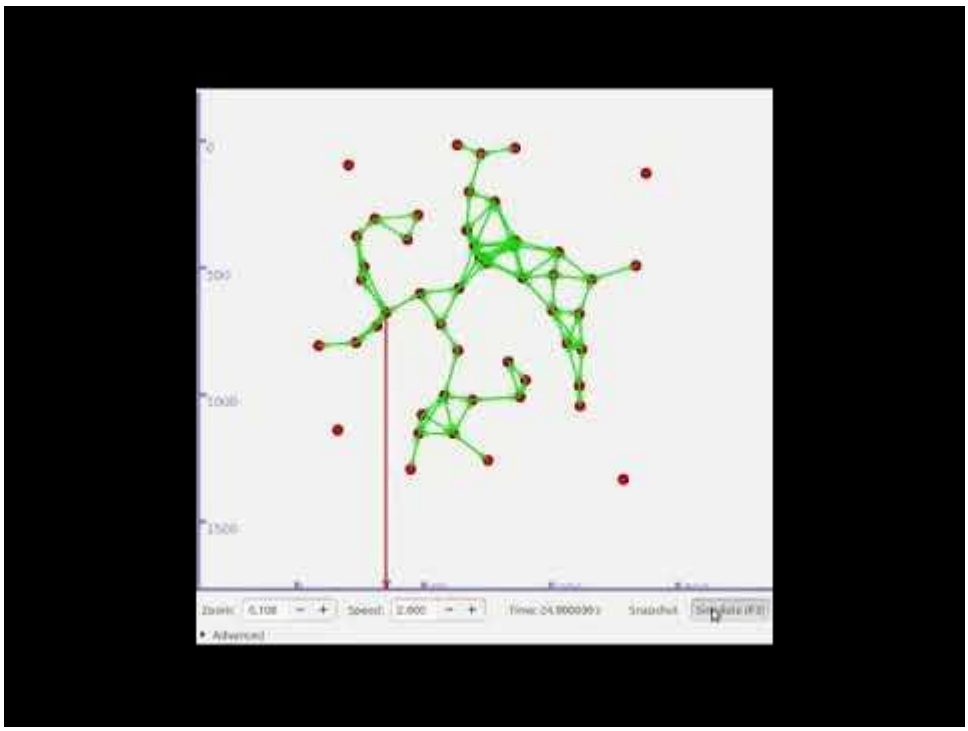
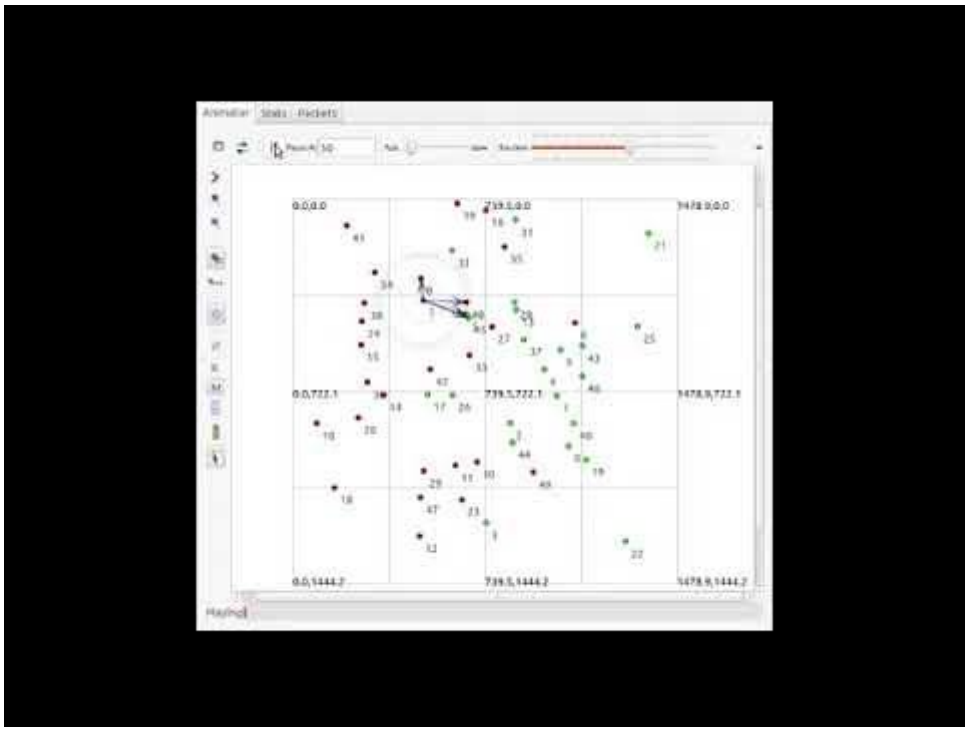


Рисунок 5. Сеть с кластеризованными узлами

Результаты работы сети наглядно можем видеть на ниже представленных видео. Визуализация была проведена как в программе PyViz (видео 1), так и в программе NetAnim (видео 2).



Видео 1. Визуализация работы сети PyViz



Видео 2. Визуализация работы сети NetAnim

5. Проверка оптимальности сети

5.1 Постановка задачи для моделирования

В работе С.А.Киреева «Оптимизация передачи информации в самоорганизующихся сетях» [10] рассматривается задача нахождения оптимального положения дрона игроков для оптимизации работы сети и уменьшения диаметра графа, на котором основана сеть.

Дадим математическую постановку задачи, решаемой автором. Пусть имеется непустое множество $N = \{1, \dots, n\}$ игроков, для каждого из которых существует непустое множество M_i агентов. Пусть X – множество позиций, а ρ – функция расстояния, заданная на подмножестве декартового произведения $X \times X$:

$$\bar{X} \subset (X \times X),\ \rho: \bar{X} \to R.$$

Кроме того, для каждого узла определена следующая функция, определяющая, является ли узел a_i соседом узла a_j:

$$\delta(a_i, x_i, a_j, x_j) = \begin{cases} 1, & \textit{если } a_j - \textit{сосед } a_i \\ 0, & \textit{иначе,} \end{cases},$$

где x_i – позиция узла a_i, а x_i – позиция узла a_j.

Граф G, на основе которого построена сеть, определим следующим образом: $G = (V, \delta, \rho)$, где V – множество пар вида (a_s^i, x_s), в которых $a_s^i \in M_i$, а $x_s \in X$, δ – функция, которая определяет, являются ли узлы, расположенные в заданных позициях, соседями друг другу, ρ – функция, которая задает расстояния между позициями узлов.

У каждого игрока имеется возможность поставить дополнительного подвижного агента. Тогда получим расширенный граф $G' = (V', \delta, \rho)$, где $V' = V \cup \bar{V}$, $\bar{V}$ – множество добавленных узлов-агентов.

В результате работы С.А.Киреев составил программу, позволяющую находить такие оптимальные положения дополнительных узлов $\bar{V}$ сети для любого исходного графа G. Данную задачу автор решает в кооперативном варианте – оптимизируется суммарный выигрыш игроков.

На основе работы С.А.Киреева в данной работе рассмотрена одна из оптимизированных сетей, для которой было найдено оптимальное положение узла сети, и выполнено экспериментальное моделирование в среде NS-3 с целью доказать или опровергнуть состоятельность выводов упомянутого выше автора. Моделирование сети произведено на основе алгоритмов организации сетей MANET посредством библиотек для моделирования NS-3.

5.2 Начальные условия

Программное обеспечение было написано на языке C++ с использование сторонних библиотек, предоставляемых симулятором NS-3. Программе на вход подается файл, который имеет следующий формат: первая строка – количество узлов в сети. Далее идут координаты всех узлов сети, которую необходимо симулировать. После – матрица смежности A всех узлов сети: если между узлом i и узлом j есть связь, то в матрице смежности элемент $a_{i,j}$ будем равен 1, иначе – 0. Координаты узлов присваиваются узлам сети NS-3 и между ними производится связь в соответствие с матрицей смежности.

5.3 Переменные для анализа

Для того, чтобы проанализировать качество построенной сети, необходимо выделить некоторое значение, которое является ключевым и показательным в работе симуляции. Для нас таким значением будет общее число переданных пакетов за все время работы симуляции и объем переданной информации в байтах.

Для начала выясним, как в симуляторе NS-3 определяются пакеты с информацией. Каждый сетевой пакет содержит байтовый буфер, набор

байтовых тегов, набор тегов пакетов и метаданные. Рассмотрим эти параметры подробнее.

Байтовый буфер хранит упорядоченное содержимое заголовков, добавленных в пакет. Ожидается, что упорядоченное представление этих заголовков будет совпадать с битами в битах реальных сетевых пакетов, что означает, что ожидается, что содержимое буфера пакетов будет содержимым реального пакета.

Набор тегов содержит информацию, специфичную для моделирования, которая не может быть сохранена в байтовом буфере пакета, потому что заголовки протокола или трейлеры не имеют стандартного поля для этой информации. Так называемые «байтовые» теги используются для маркировки подмножества байтов в байтовом буфере пакета, в то время как «пакетные» теги используются для маркировки самого пакета. Основное различие между этими двумя типами тегов заключается в том, что происходит, когда пакеты копируются, фрагментируются и повторно собираются: «байтовые» теги следуют за байтами, а «пакетные» теги следуют за пакетами. Другое важное различие между этими двумя типами тегов состоит в том, что байтовые теги нельзя удалить, и ожидается, что они будут записаны один раз и прочитаны много раз, в то время как теги пакетов должны быть записаны один раз, прочитаны много раз и удалены ровно один раз.

5.4 Проведение симуляции

Для того, чтобы доказать состоятельность полученных результатов улучшения существующей сети путем установки дополнительного узла, рассмотрим конкретную сеть, которая была получена в результате работы программы. Исходная сеть представлена на рисунке 6, сеть с дополнительным узлом представлена на рисунке 7.

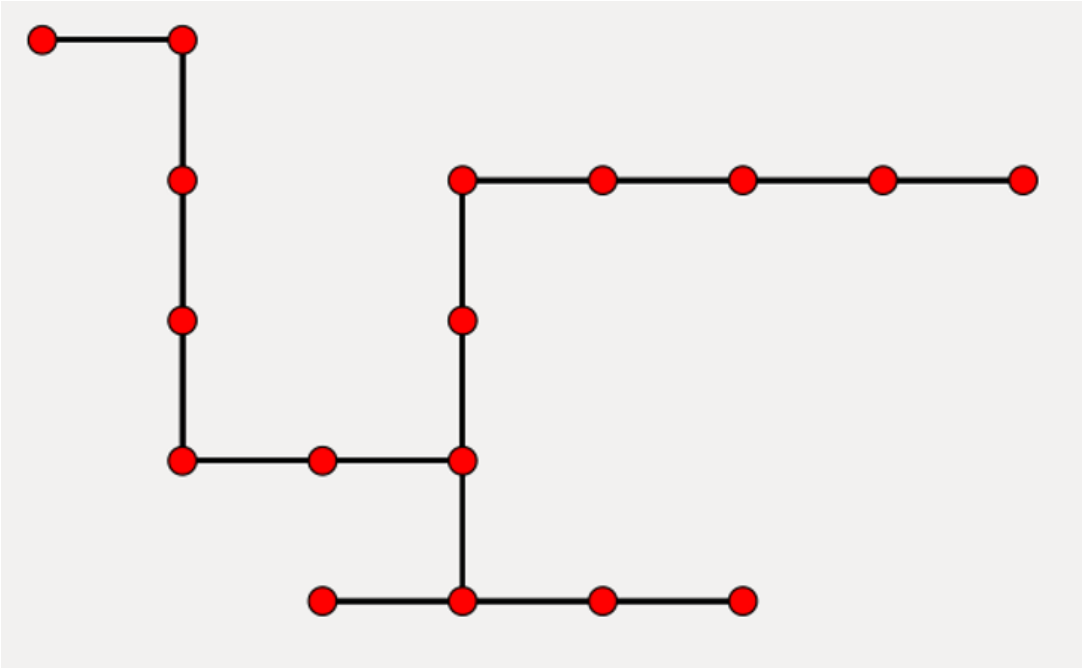


Рисунок 6. Исходная сеть

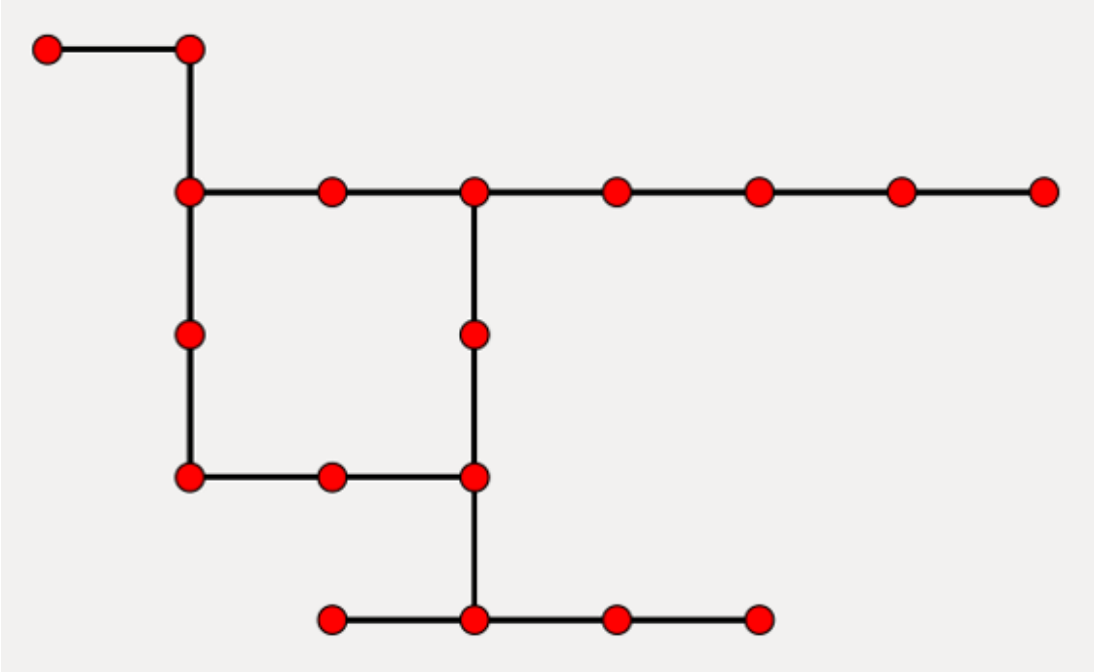


Рисунок 7. Улучшенная сеть с дополнительным узлом

Предполагается, что новый узел улучшит работу сети и позволит передавать внутри сети больший объем информации.

5.5 Анализ сети

Необходимые нам данные можно получить с помощью встроенной технологии мониторинга потоков в сети NS-3. Цель модуля мониторинга

потоков (Flow Monitor) - предоставить гибкую систему для измерения производительности сетевых протоколов. Модуль использует датчики, установленные в сетевых узлах, для отслеживания пакетов, которыми обмениваются узлы. Пакеты передаются соответственно потокам, которым они принадлежат, и каждый поток определяется в соответствии с характеристиками датчиков (например, для датчика IP такими характеристиками будут являться кортежи вида {protocol, source (IP, port), destination (IP, port)}. Статистика, собранная для каждого потока, может быть экспортирована в формате XML. Кроме того, NS-3 позволяет получить доступ к датчикам напрямую, чтобы запросить конкретные статистические данные о каждом потоке.

В ходе выполнения программы в файле .flowmon генерируется информация о всех потоках, протекавших в сети. Отсюда можно наблюдать, в какой момент времени симуляции начался поток, в какое время закончился, сколько было передано пакетов за время передачи данных и каким размером. Все эти данные, несомненно, важны для общего анализа сети, однако мы остановимся на рассмотрении количества переданных пакетов в потоке и размере переданной информации.

Полученный .xml файл преобразуем для большей наглядности и более удобного оперирования данными в .csv файл и рассмотрим полученные результаты (см. приложения 1, 2).

Представим наглядно полученные данные на графиках:

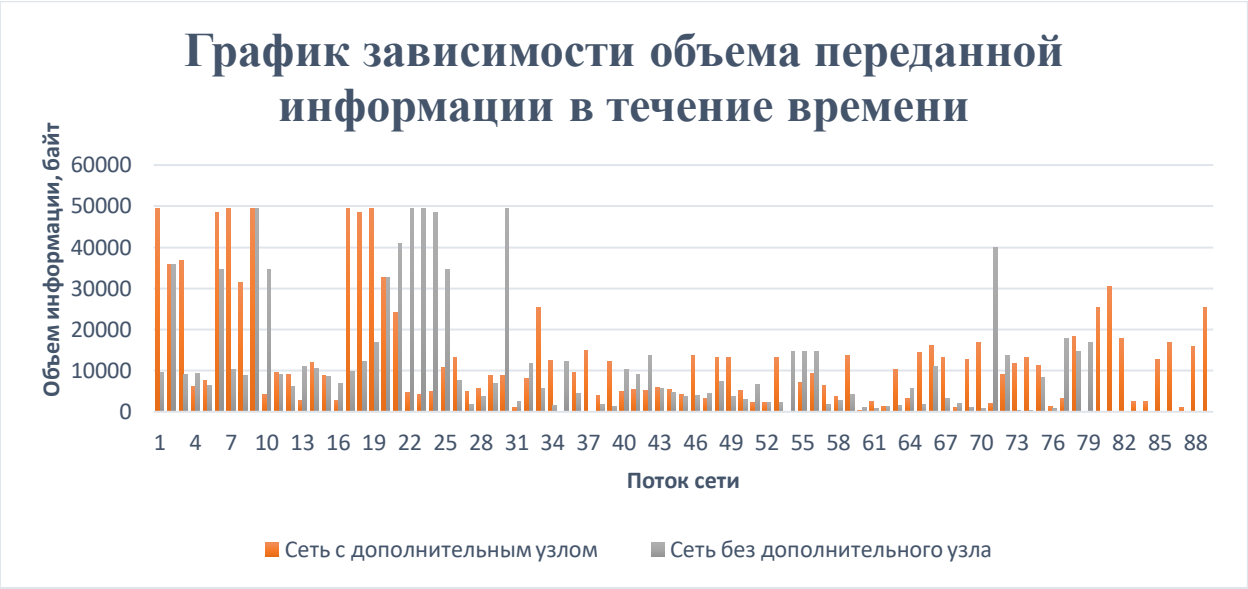


График 1. Зависимость объема переданной информации в течение времени

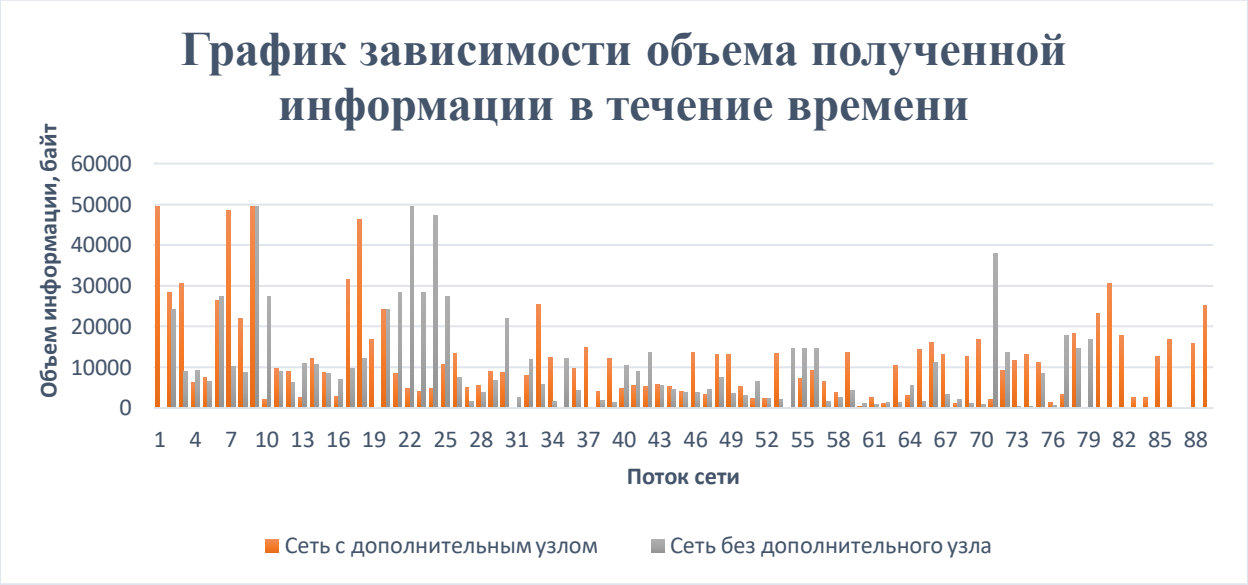


График 2. Зависимость объема полученной информации в течение времени

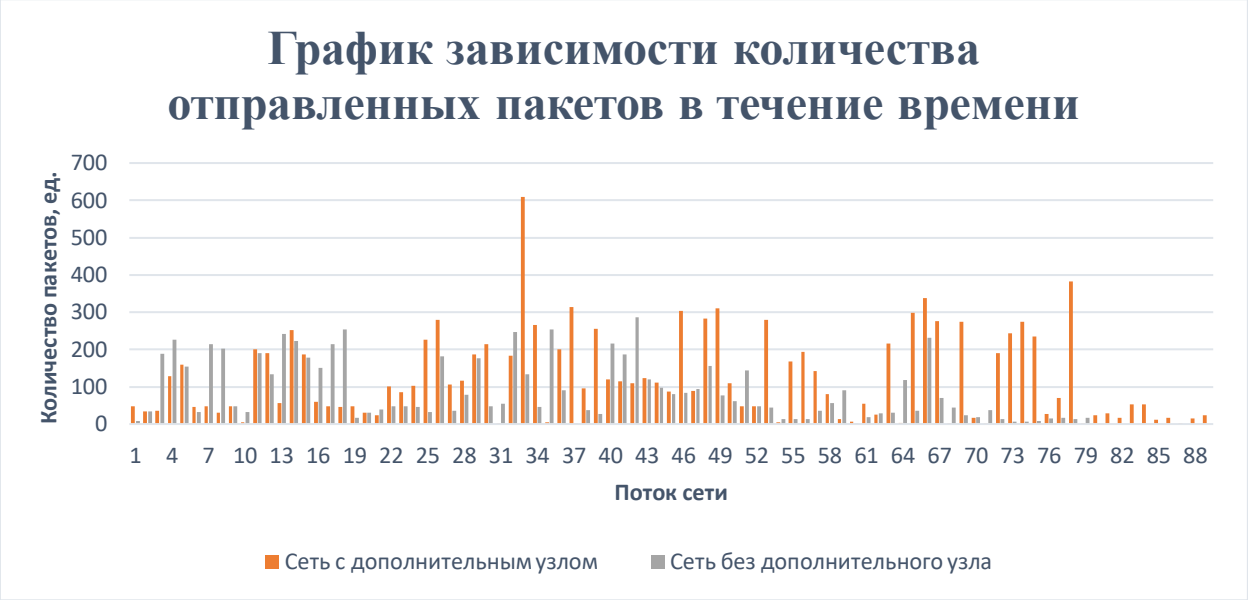


График 3. Зависимость количества отправленных пакетов в течение времени

График 4. Зависимость количества полученных пакетов в течение времени

График 5. Сравнительная диаграмма числа пакетов в полученных сетях

График 6. Сравнительная диаграмма объема информации в полученных сетях

5.6 Результаты исследования симуляции

Первое, что необходимо отметить в исследовании качества улучшения сети, – количество потоков в сети возросло. За одно и то же время симуляции

в сети до добавления узла происходила передача информации в 79 потоков, после добавления узла – в 89. Увеличение числа потоков позволило передавать большее количество пакетов информации, что, безусловно, положительно влияет на работоспособность сети.

Вторым важным показателем улучшения сети является число переданных и полученных пакетов: при симуляции сети с добавленным узлом было передано на 3912 пакетов больше, чем при симуляции сети без такового узла. Этот показатель играет существенную роль при формировании сети: возможность передавать большее число пакетов данных за то же время является ключевой при проектировании любой сети, следовательно, такое весомое увеличение числа переданных пакетов доказывает состоятельность данной модели.

Кроме того, весомо возрос объем переданных данных, что является важнейшим сравнительным критерием при анализе работоспособности и качества сети. Как можно видеть на графике 6, общий объем информации в сети, построенной с дополнительным узлом, существенно выше объема информации в сети, построенной без дополнительного узла. В числовом эквиваленте разница между объемами информации, которые были переданы между узлами сетей, составила приблизительно 230 тыс. байт.

Подводя итоги анализа, можно утверждать, что сеть, построенная на основе графа с дополнительным узлом, найденным в результате работы программы С.Киреева, действительно является более эффективной, и качество работы построенной сети существенно выше качества работы сети без такового узла, что экспериментально подтверждает состоятельность его работы.

6. Заключение

В ходе работы были выполнены все поставленные цели. Был проведен подробный анализ сетей MANET, широко используемых в настоящее время. Были выделены ключевые особенности сети, рассмотрено применение MANET в современных технологиях и различных областях. Проведенный анализ позволил также выделить преимущества и недостатки такого способа организации сети и выделить основные алгоритмы маршрутизации, используемые при организации этих сетей.

Также был изучен один из способов симуляции различных сетей, в частности MANET-сетей, представленный программным обеспечением NS-3. С его помощью было проведено исследование сети со случайно движущимися узлами, проведена визуализация полученной сети.

Кроме того, с использованием симулятора NS-3 было проведено практическое исследование организации MANET-сети на базе работы С.А.Киреева. Результат проведенного исследования позволил экспериментально подтвердить выдвинутую автором теорию о возможном улучшении работы сети путем установления дополнительных узлов при кооперативном установлении сети. Данные выводы могут быть практически использованы для дальнейшего исследования оптимизированных сетей и позволят проводить показательные экспериментальные симуляции и для других сетей.

Список литературы

1. Emerging Wireless Networks: Concepts, Techniques and Applications, Christian Makaya, Samuel Pierre

2. Mobile Computing and Wireless Communications: Applications, Networks, Platforms, Architectures, and Security; Amjad Umar, 2004

3. MANET Routing Protocols Evaluation: AODV, DSR and DSDV Perspective, Fahad Taha AL-Dhief, Naseer Sabri, M.S. Salim, S. Fouad, S. A. Aljunid, 2017

4. Pareto-optimal Solutions in a Game of Mobile Agents Placements in a MANET, E. Gromova, A. Vorontsov, S. Blakeway, A. Kirpichnikova, 2019

5. Blakeway S., Gromov D. V., Gromova E. V., Kirpichnikova A. S., Plekhanova T. M. Increasing the performance of a Mobile Ad-hoc Network using a game-theoretic approach to drone positioning // Вестник Санкт-Петербургского университета. Прикладная математика. Информатика. Процессы управления. 2019. Т. 14. Вып. 1. С. 22–38. https:// doi.org/10.21638/11702/spbu10.2019.102

6. Introduction of Mobile Ad hoc Network (MANET) https://www.geeksforgeeks.org/introduction-of-mobile-ad-hoc-network-manet/

7. А.А. Павлов, И.О. Датьев, Протоколы маршрутизации в беспроводных сетях

8. NS-3 Documentation https://www.nsnam.org/doxygen/index.html

9. S. Blakeway, D. V. Gromov, E. V. Gromova, A. S. Kirpichnikova, T. M. Plekhanova, “Increasing the performance of a Mobile Ad-hoc Network using a game-theoretic approach to drone positioning”, Вестн. С.-Петербург. ун-та. Сер. 10. Прикл. матем. Информ. Проц. упр., 15:1 (2019), 22–38

10. Киреев С.А. Оптимизация передачи информации в самоорганизующихся сетях // Процессы управления и устойчивость. 2020. Т. 67 № 1.

7. Приложения

Приложение 1. Таблица потоков без узла

Передано информации, байт	Получено информации, байт	Передано пакетов	Получено пакетов
9468	0	9	0
35768	24196	34	23
9000	9000	189	189
9294	9294	227	227
6504	6504	154	154
34716	27352	33	26
10218	10218	214	214
8760	8760	203	203
49444	49444	47	47
34716	27352	33	26
9084	9084	190	190
6230	6230	134	134
11058	11058	242	242
10584	10584	222	222
8472	8472	178	178
6974	6974	151	151
9726	9726	215	215
12118	12118	253	253
16832	0	16	0
32612	24196	31	23
41028	28404	39	27
49444	49444	47	47
49444	28404	47	27
48392	47340	46	45
34716	27352	33	26
7588	7588	181	181
1728	1728	36	36
3748	3748	79	79
6768	6768	177	177

49444	22092	47	21
2556	2556	54	54
11808	11808	246	246
5694	5694	134	134
1596	1596	46	46
12174	12174	254	254
4368	4368	91	91
48	48	1	1
1776	1776	37	37
1296	1296	27	27
10368	10368	216	216
8976	8976	187	187
13728	13728	286	286
5598	5598	120	120
4614	4614	98	98
3732	3732	80	80
3870	3870	84	84
4512	4512	94	94
7440	7440	155	155
3696	3696	77	77
2976	2976	62	62
6592	6592	144	144
2256	2256	47	47
2160	2160	45	45
14728	14728	14	14
14728	14728	14	14
14728	14728	14	14
1664	1664	35	35
2688	2688	56	56
4284	4284	90	90
1052	1052	1	1
864	864	18	18
1344	1344	28	28
1440	1440	30	30

5592	5592	118	118
1680	1680	35	35
11088	11088	231	231
3352	3352	70	70
2112	2112	44	44
1104	1104	23	23
912	912	19	19
39976	37872	38	36
13676	13676	13	13
336	336	7	7
336	336	7	7
8416	8416	8	8
720	720	15	15
17884	17884	17	17
14728	14728	14	14
16832	16832	16	16
Итого:			
941976	809424	7067	6941

Приложение 2. Таблица потоков с узлом

Передано информации, байт	Получено информации, байт	Передано пакетов	Получено пакетов
49444	49444	47	47
35768	28404	34	27
36820	30508	35	29
6174	6174	129	129
7596	7596	159	159
48392	26300	46	25
49444	48392	47	46
31560	22092	30	21
49444	49444	47	47
4208	2104	4	2
9648	9648	201	201
9042	9042	191	191
2674	2674	57	57
12096	12096	252	252
8816	8816	187	187
2844	2844	60	60
49444	31560	47	30
48392	46288	46	44
49444	16832	47	16
32612	24196	31	23
24196	8416	23	8
4794	4794	101	101
4110	4110	86	86
4908	4908	103	103
10806	10806	227	227
13276	13276	279	279
4980	4980	106	106
5616	5616	117	117
8928	8928	186	186
8836	8836	215	215
1052	0	1	0

8020	8020	183	183
25398	25398	609	609
12522	12522	265	265
192	192	4	4
9648	9648	201	201
14912	14912	313	313
4038	4038	95	95
12214	12214	255	255
4916	4916	120	120
5520	5520	115	115
5218	5218	110	110
5868	5868	123	123
5320	5320	111	111
4176	4176	87	87
13600	13600	303	303
3318	3318	89	89
13164	13164	283	283
13234	13138	310	308
5272	5272	110	110
2256	2256	47	47
2304	2304	48	48
13290	13290	279	279
192	192	4	4
7242	7242	167	167
9264	9264	193	193
6456	6456	142	142
3840	3840	80	80
13676	13676	13	13
336	336	7	7
2536	2536	54	54
1232	1232	26	26
10368	10368	216	216
3156	3156	3	3
14304	14304	298	298

16160	16160	337	337
13248	13248	276	276
1052	1052	1	1
12702	12702	275	275
16832	16832	16	16
2104	2104	2	2
9168	9168	191	191
11676	11676	244	244
13192	13192	275	275
11272	11272	235	235
1256	1256	27	27
3288	3288	70	70
18328	18328	382	382
48	48	1	1
25248	23144	24	22
30508	30508	29	29
17884	17884	17	17
2544	2544	53	53
2496	2496	52	52
12624	12624	12	12
16832	16832	16	16
1052	0	1	0
15780	15780	15	15
25248	25248	24	24
Итого:			
1172908	1043416	10979	10854

Приложение 3. Исходный код программы для симуляции сети с движущимися узлами

```
#include <fstream>
#include <iostream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
#include "ns3/dsdv-module.h"
#include "ns3/dsr-module.h"
#include "ns3/applications-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/netanim-module.h"
#include "kmeans.h"
#include <string>
#include <vector>


using namespace ns3;
using namespace dsr;

NS_LOG_COMPONENT_DEFINE ("manet-routing-compare");

class RoutingExperiment
{
public:
  RoutingExperiment ();
  void Run (int nSinks, double txp, std::string CSVfileName);
  //static void SetMACParam (ns3::NetDeviceContainer & devices,
  //                         int slotDistance);
  std::string CommandSetup (int argc, char **argv);

private:
  Ptr<Socket> SetupPacketReceive (Ipv4Address addr, Ptr<Node> node);
  void ReceivePacket (Ptr<Socket> socket);
  void CheckThroughput ();

  uint32_t port;
  uint32_t bytesTotal;
  uint32_t packetsReceived;

  std::string m_CSVfileName;
  int m_nSinks;
  std::string m_protocolName;
  double m_txp;
  bool m_traceMobility;
  uint32_t m_protocol;
};

RoutingExperiment::RoutingExperiment ()
  : port (9),
    bytesTotal (0),
    packetsReceived (0),
    m_CSVfileName ("DSR.csv"),
    m_traceMobility (false),
    m_protocol (2) // AODV
{
}

static inline std::string
PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet, Address senderAddress)
```

```
{
  std::ostringstream oss;

  oss << Simulator::Now ().GetSeconds () << " " << socket->GetNode ()->GetId ();

  if (InetSocketAddress::IsMatchingType (senderAddress))
    {
      InetSocketAddress addr = InetSocketAddress::ConvertFrom (senderAddress);
      oss << " received one packet from " << addr.GetIpv4 ();
    }
  else
    {
      oss << " received one packet!";
    }
  return oss.str ();
}

void
RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
{
  Ptr<Packet> packet;
  Address senderAddress;
  while ((packet = socket->RecvFrom (senderAddress)))
    {
      bytesTotal += packet->GetSize ();
      packetsReceived += 1;
      NS_LOG_UNCOND (PrintReceivedPacket (socket, packet, senderAddress));
    }
}

void
RoutingExperiment::CheckThroughput ()
{
  double kbs = (bytesTotal * 8.0) / 1000;
  bytesTotal = 0;

  std::ofstream out (m_CSVfileName.c_str (), std::ios::app);

        out << (Simulator::Now ()).GetSeconds () << ","
            << kbs << ","
            << packetsReceived << ","
            << m_nSinks << ","
            << m_protocolName << ","
            << m_txp << ""
            << std::endl;

        out.close ();
        packetsReceived = 0;
        Simulator::Schedule (Seconds (1.0), &RoutingExperiment::CheckThroughput, this
);
      }

      Ptr<Socket>
      RoutingExperiment::SetupPacketReceive (Ipv4Address addr, Ptr<Node> node)
      {
        TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
        Ptr<Socket> sink = Socket::CreateSocket (node, tid);
        InetSocketAddress local = InetSocketAddress (addr, port);
        sink->Bind (local);
        sink-
>SetRecvCallback (MakeCallback (&RoutingExperiment::ReceivePacket, this));

        return sink;
      }

      std::string
      RoutingExperiment::CommandSetup (int argc, char **argv)
      {
```

```
129.          CommandLine cmd;
130.          cmd.AddValue ("CSVfileName", "The name of the CSV output file name", m_CSVfil
    eName);
131.          cmd.AddValue ("traceMobility", "Enable mobility tracing", m_traceMobility);
132.          cmd.AddValue ("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR", m_protocol);
133.          cmd.Parse (argc, argv);
134.          return m_CSVfileName;
135.        }
136.
137.        int
138.        main (int argc, char *argv[])
139.        {
140.          RoutingExperiment experiment;
141.          std::string CSVfileName = experiment.CommandSetup (argc,argv);
142.
143.          //blank out the last output file and write the column headers
144.          std::ofstream out (CSVfileName.c_str ());
145.          out << "SimulationSecond," <<
146.          "ReceiveRate," <<
147.          "PacketsReceived," <<
148.          "NumberOfSinks," <<
149.          "RoutingProtocol," <<
150.          "TransmissionPower" <<
151.          std::endl;
152.          out.close ();
153.
154.          int nSinks = 10;
155.          double txp = 10;
156.
157.          experiment.Run (nSinks, txp, CSVfileName);
158.        }
159.
160.        void
161.        RoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
162.        {
163.          Packet::EnablePrinting ();
164.          m_nSinks = nSinks;
165.          m_txp = txp;
166.          m_CSVfileName = CSVfileName;
167.
168.          int nWifis = 50;
169.
170.          double TotalTime = 50.0;
171.          std::string rate ("2048bps");
172.          std::string phyMode ("DsssRate11Mbps");
173.          std::string tr_name ("DSR");
174.          int nodeSpeed = 20; //in m/s
175.          int nodePause = 0; //in s
176.          m_protocolName = "protocol";
177.
178.          Config::SetDefault  ("ns3::OnOffApplication::PacketSize",StringValue ("64"));

179.          Config::SetDefault ("ns3::OnOffApplication::DataRate",  StringValue (rate));

180.
181.          //Set Non-unicastMode rate to unicast mode
182.          Config::SetDefault ("ns3::WifiRemoteStationManager::NonUnicastMode",StringVal
    ue (phyMode));
183.
184.          NodeContainer adhocNodes;
185.          adhocNodes.Create (nWifis);
186.
187.          // setting up wifi phy and channel using helpers
188.          WifiHelper wifi;
189.          wifi.SetStandard (WIFI_PHY_STANDARD_80211b);
190.
191.          YansWifiPhyHelper wifiPhy =  YansWifiPhyHelper::Default ();
192.          YansWifiChannelHelper wifiChannel;
```

```
193.          wifiChannel.SetPropagationDelay ("ns3::ConstantSpeedPropagationDelayModel");
194.          wifiChannel.AddPropagationLoss ("ns3::FriisPropagationLossModel");
195.          wifiPhy.SetChannel (wifiChannel.Create ());
196.
197.          // Add a mac and disable rate control
198.          WifiMacHelper wifiMac;
199.          wifi.SetRemoteStationManager ("ns3::ConstantRateWifiManager",
200.                                        "DataMode",StringValue (phyMode),
201.                                        "ControlMode",StringValue (phyMode));
202.
203.          wifiPhy.Set ("TxPowerStart",DoubleValue (txp));
204.          wifiPhy.Set ("TxPowerEnd", DoubleValue (txp));
205.
206.          wifiMac.SetType ("ns3::AdhocWifiMac");
207.          NetDeviceContainer adhocDevices = wifi.Install (wifiPhy, wifiMac, adhocNodes)
    ;
208.
209.          MobilityHelper mobilityAdhoc;
210.          int64_t streamIndex = 0; // used to get consistent mobility across scenarios
211.
212.          ObjectFactory pos;
213.          pos.SetTypeId ("ns3::RandomRectanglePositionAllocator");
214.          pos.Set ("X", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1500.0]"))
    ;
215.          pos.Set ("Y", StringValue ("ns3::UniformRandomVariable[Min=0.0|Max=1500.0]"))
    ;
216.
217.          Ptr<PositionAllocator> taPositionAlloc = pos.Create ()-
    >GetObject<PositionAllocator> ();
218.          streamIndex += taPositionAlloc->AssignStreams (streamIndex);
219.
220.          std::stringstream ssSpeed;
221.          ssSpeed << "ns3::UniformRandomVariable[Min=0.0|Max=" << nodeSpeed << "]";
222.          std::stringstream ssPause;
223.          ssPause << "ns3::ConstantRandomVariable[Constant=" << nodePause << "]";
224.          mobilityAdhoc.SetMobilityModel ("ns3::RandomWaypointMobilityModel",
225.                                          "Speed", StringValue (ssSpeed.str ()),
226.                                          "Pause", StringValue (ssPause.str ()),
227.                                          "PositionAllocator", PointerValue (taPosition
    Alloc));
228.          mobilityAdhoc.SetPositionAllocator (taPositionAlloc);
229.          mobilityAdhoc.Install (adhocNodes);
230.          streamIndex += mobilityAdhoc.AssignStreams (adhocNodes, streamIndex);
231.          NS_UNUSED (streamIndex); // From this point, streamIndex is unused
232.
233.          AodvHelper aodv;
234.          Ipv4ListRoutingHelper list;
235.          InternetStackHelper internet;
236.
237.          list.Add (aodv, 100);
238.          m_protocolName = "AODV";
239.
240.          internet.SetRoutingHelper (list);
241.          internet.Install (adhocNodes);
242.
243.
244.          NS_LOG_INFO ("assigning ip address");
245.
246.          Ipv4AddressHelper addressAdhoc;
247.          addressAdhoc.SetBase ("10.1.1.0", "255.255.255.0");
248.          Ipv4InterfaceContainer adhocInterfaces;
249.          adhocInterfaces = addressAdhoc.Assign (adhocDevices);
250.
251.          OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
252.          onoff1.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Cons
    tant=1.0]"));
```

```
253.         onoff1.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Con
    stant=0.0]"));
254.
255.         for (int i = 0; i < nSinks; i++)
256.           {
257.             Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (i), ad
    hocNodes.Get (i));
258.
259.             AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.GetAddress
     (i), port));
260.             onoff1.SetAttribute ("Remote", remoteAddress);
261.
262.             Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable> ();

263.             ApplicationContainer temp = onoff1.Install (adhocNodes.Get (i + nSinks));

264.             temp.Start (Seconds (var->GetValue (10.0,11.0)));
265.             temp.Stop (Seconds (TotalTime));
266.           }
267.
268.         std::stringstream ss;
269.         ss << nWifis;
270.         std::string nodes = ss.str ();
271.
272.         std::stringstream ss2;
273.         ss2 << nodeSpeed;
274.         std::string sNodeSpeed = ss2.str ();
275.
276.         std::stringstream ss3;
277.         ss3 << nodePause;
278.         std::string sNodePause = ss3.str ();
279.
280.         std::stringstream ss4;
281.         ss4 << rate;
282.         std::string sRate = ss4.str ();
283.
284.         AsciiTraceHelper ascii;
285.         MobilityHelper::EnableAsciiAll (ascii.CreateFileStream (tr_name + ".mob"));
286.
287.         Ptr<FlowMonitor> flowmon;
288.         FlowMonitorHelper flowmonHelper;
289.         flowmon = flowmonHelper.InstallAll ();
290.
291.
292.         NS_LOG_INFO ("Run Simulation.");
293.
294.         CheckThroughput ();
295.
296.         AnimationInterface anim ("anim1.xml");
297.
298.         int pointId = 0;
299.         vector<Point> all_points;
300.         string line;
301.         // int x,y;
302.         int K = 2;
303.
304.         for (NodeContainer::Iterator j = adhocNodes.Begin ();
305.              j != adhocNodes.End (); ++j)
306.           {
307.             Ptr<Node> object = *j;
308.             Ptr<MobilityModel> position = object->GetObject<MobilityModel> ();
309.             NS_ASSERT (position != 0);
310.             Vector pos = position->GetPosition ();
311.
312.             line = to_string(pos.x) + ' ' + to_string(pos.y) + ' ';
313.
314.             Point point(pointId, line);
315.             all_points.push_back(point);
```

```
316.             pointId++;
317.           }
318.
319.         int iters = 100;
320.       // Adjasting k-means clustering
321.         KMeans kmeans(K, iters);
322.         kmeans.run(all_points);
323.
324.         ifstream read_kmeans;
325.
326.         read_kmeans.open("cluster_points.txt");
327.
328.         struct rgb {
329.           int r;
330.           int g;
331.           int b;
332.         };
333.
334.         struct rgb color = {255, 0, 0};
335.
336.         if (read_kmeans.is_open()){
337.           while (getline(read_kmeans, line)) {
338.             if (line == "*"){
339.               color = {0, 255, 0};
340.             }
341.             else
342.             {
343.               int nodeId = stoi(line);
344.               anim.UpdateNodeColor (nodeId, color.r, color.g, color.b);
345.             }
346.           }
347.         }
348.
349.         Simulator::Stop (Seconds (TotalTime));
350.         Simulator::Run ();
351.
352.         flowmon->SerializeToXmlFile ((tr_name + ".flowmon").c_str(), false, false);
353.
354.         Simulator::Destroy ();
355.       }
```

Приложение 4. Библиотека для кластеризации узлов

```
#include <iostream>
#include <vector>
#include <cmath>
#include <fstream>
#include <sstream>
#include <algorithm>

using namespace std;

class Point{

private:
    int pointId, clusterId;
    int dimensions;
    vector<double> values;

public:
    Point(int id, string line){
        dimensions = 0;
        pointId = id;
        stringstream is(line);
        double val;
        while(is >> val){
            values.push_back(val);
            dimensions++;
        }
        clusterId = 0; //Initially not assigned to any cluster
    }

    int getDimensions(){
        return dimensions;
    }

    int getCluster(){
        return clusterId;
    }

    int getID(){
        return pointId;
    }

    void setCluster(int val){
        clusterId = val;
    }

    double getVal(int pos){
        return values[pos];
    }
};

class Cluster{

private:
    int clusterId;
    vector<double> centroid;
    vector<Point> points;

public:
    Cluster(int clusterId, Point centroid){
        this->clusterId = clusterId;
        for(int i=0; i<centroid.getDimensions(); i++){
            this->centroid.push_back(centroid.getVal(i));
        }
        this->addPoint(centroid);
```

```
65.     }
66.
67.     void addPoint(Point p){
68.         p.setCluster(this->clusterId);
69.         points.push_back(p);
70.     }
71.
72.     bool removePoint(int pointId){
73.         int size = points.size();
74.
75.         for(int i = 0; i < size; i++)
76.         {
77.             if(points[i].getID() == pointId)
78.             {
79.                 points.erase(points.begin() + i);
80.                 return true;
81.             }
82.         }
83.         return false;
84.     }
85.
86.     int getId(){
87.         return clusterId;
88.     }
89.
90.     Point getPoint(int pos){
91.         return points[pos];
92.     }
93.
94.     int getSize(){
95.         return points.size();
96.     }
97.
98.     double getCentroidByPos(int pos) {
99.         return centroid[pos];
100.        }
101.
102.        void setCentroidByPos(int pos, double val){
103.            this->centroid[pos] = val;
104.        }
105.    };
106.
107.    class KMeans{
108.    private:
109.        int K, iters, dimensions, total_points;
110.        vector<Cluster> clusters;
111.
112.        int getNearestClusterId(Point point){
113.            double sum = 0.0, min_dist;
114.            int NearestClusterId;
115.
116.            for(int i = 0; i < dimensions; i++)
117.            {
118.                sum += pow(clusters[0].getCentroidByPos(i) - point.getVal(i), 2.0);
119.            }
120.
121.            min_dist = sqrt(sum);
122.            NearestClusterId = clusters[0].getId();
123.
124.            for(int i = 1; i < K; i++)
125.            {
126.                double dist;
127.                sum = 0.0;
128.
129.                for(int j = 0; j < dimensions; j++)
130.                {
```

```
131.                         sum += pow(clusters[i].getCentroidByPos(j) - point.getVal(j), 2
    .0);
132.                     }
133.
134.                     dist = sqrt(sum);
135.
136.                     if(dist < min_dist)
137.                     {
138.                         min_dist = dist;
139.                         NearestClusterId = clusters[i].getId();
140.                     }
141.                 }
142.
143.                 return NearestClusterId;
144.             }
145.
146.         public:
147.             KMeans(int K, int iterations){
148.                 this->K = K;
149.                 this->iters = iterations;
150.             }
151.
152.             void run(vector<Point>& all_points){
153.
154.                 total_points = all_points.size();
155.                 dimensions = all_points[0].getDimensions();
156.
157.
158.                 //Initializing Clusters
159.                 vector<int> used_pointIds;
160.
161.                 for(int i=1; i<=K; i++)
162.                 {
163.                     while(true)
164.                     {
165.                         int index = rand() % total_points;
166.
167.                         if(find(used_pointIds.begin(), used_pointIds.end(), index) == u
    sed_pointIds.end())
168.                         {
169.                             used_pointIds.push_back(index);
170.                             all_points[index].setCluster(i);
171.                             Cluster cluster(i, all_points[index]);
172.                             clusters.push_back(cluster);
173.                             break;
174.                         }
175.                     }
176.                 }
177.                 cout<<"Clusters initialized = "<<clusters.size()<<endl<<endl;
178.
179.
180.                 cout<<"Running K-Means Clustering.."<<endl;
181.
182.                 int iter = 1;
183.                 while(true)
184.                 {
185.                     cout<<"Iter - "<<iter<<"/"<<iters<<endl;
186.                     bool done = true;
187.
188.                     // Add all points to their nearest cluster
189.                     for(int i = 0; i < total_points; i++)
190.                     {
191.                         int currentClusterId = all_points[i].getCluster();
192.                         int nearestClusterId = getNearestClusterId(all_points[i]);
193.
194.                         if(currentClusterId != nearestClusterId)
195.                         {
196.                             if(currentClusterId != 0){
```

```
                                for(int j=0; j<K; j++){
                                    if(clusters[j].getId() == currentClusterId){
                                        clusters[j].removePoint(all_points[i].getID());

                                    }
                                }
                            }

                            for(int j=0; j<K; j++){
                                if(clusters[j].getId() == nearestClusterId){
                                    clusters[j].addPoint(all_points[i]);
                                }
                            }
                            all_points[i].setCluster(nearestClusterId);
                            done = false;
                        }
                    }

                    // Recalculating the center of each cluster
                    for(int i = 0; i < K; i++)
                    {
                        int ClusterSize = clusters[i].getSize();

                        for(int j = 0; j < dimensions; j++)
                        {
                            double sum = 0.0;
                            if(ClusterSize > 0)
                            {
                                for(int p = 0; p < ClusterSize; p++)
                                    sum += clusters[i].getPoint(p).getVal(j);
                                clusters[i].setCentroidByPos(j, sum / ClusterSize);
                            }
                        }
                    }

                    if(done || iter >= iters)
                    {
                        cout << "Clustering completed in iteration : " <<iter<<endl<<en
    dl;
                        break;
                    }
                    iter++;
                }


                //Print pointIds in each cluster
                ofstream outfile;
                outfile.open("cluster_points.txt");
                for(int i=0; i<K; i++){
                    cout<<"Points in cluster "<<clusters[i].getId()<<" : ";
                    for(int j=0; j<clusters[i].getSize(); j++){
                        cout<<clusters[i].getPoint(j).getID()<<" ";
                        outfile<<clusters[i].getPoint(j).getID()<<endl;
                        //outfile<<clusters[i].getPoint(j).getVal(0)<<" "<<clusters[i].
    getPoint(j).getVal(1)<<endl;
                    }
                    outfile<<"*"<<endl;
                    cout<<endl<<endl;
                }
                cout<<"========================"<<endl<<endl;
                outfile.close();

                //Write cluster centers to file

                outfile.open("clusters.txt");
                if(outfile.is_open()){
                    for(int i=0; i<K; i++){
                        cout<<"Cluster "<<clusters[i].getId()<<" centroid : ";
```

```
                        for(int j=0; j<dimensions; j++){
                            cout<<clusters[i].getCentroidByPos(j)<<" ";     //Output to console
                            outfile<<clusters[i].getCentroidByPos(j)<<" ";  //Output to file
                        }
                        cout<<endl;
                        outfile<<endl;
                    }
                    outfile.close();
                }
                else{
                    cout<<"Error: Unable to write to clusters.txt";
                }

            }
        };
```

Приложение 5. Исходный код программы симуляции для задачи о проверке оптимальности сети

```
#include <fstream>
#include <iostream>
#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/aodv-module.h"
#include "ns3/olsr-module.h"
#include "ns3/dsdv-module.h"
#include "ns3/dsr-module.h"
#include "ns3/applications-module.h"
#include "ns3/yans-wifi-helper.h"
#include "ns3/flow-monitor-helper.h"
#include "ns3/netanim-module.h"
#include "ns3/default-deleter.h"
#include "ns3/point-to-point-module.h"
#include "kmeans.h"
#include <string>
#include <vector>

using namespace ns3;
using namespace dsr;

NS_LOG_COMPONENT_DEFINE ("manet-routing");

class RoutingExperiment
{
public:
  RoutingExperiment ();
  void Run (int nSinks, double txp, std::string CSVfileName);
  //static void SetMACParam (ns3::NetDeviceContainer & devices,
  //                         int slotDistance);
  std::string CommandSetup (int argc, char **argv);

private:
  Ptr<Socket> SetupPacketReceive (Ipv4Address addr, Ptr<Node> node);
  void ReceivePacket (Ptr<Socket> socket);
  void CheckThroughput ();

  uint32_t port;
  uint32_t bytesTotal;
  uint32_t packetsReceived;

  std::string m_CSVfileName;
  int m_nSinks;
  std::string m_protocolName;
  double m_txp;
  bool m_traceMobility;
  uint32_t m_protocol;
};

RoutingExperiment::RoutingExperiment ()
  : port (9),
    bytesTotal (0),
    packetsReceived (0),
    m_CSVfileName ("p2p_manet.csv"),
    m_traceMobility (false),
    m_protocol (2) // AODV
{
}

static inline std::string
```

```
63. PrintReceivedPacket (Ptr<Socket> socket, Ptr<Packet> packet, Address senderAddress)
64. {
65.   std::ostringstream oss;
66.
67.   oss << Simulator::Now ().GetSeconds () << " " << socket->GetNode ()->GetId ();
68.
69.   if (InetSocketAddress::IsMatchingType (senderAddress))
70.     {
71.       InetSocketAddress addr = InetSocketAddress::ConvertFrom (senderAddress);
72.       oss << " received one packet from " << addr.GetIpv4 ();
73.     }
74.   else
75.     {
76.       oss << " received one packet!";
77.     }
78.   return oss.str ();
79. }
80.
81. void
82. RoutingExperiment::ReceivePacket (Ptr<Socket> socket)
83. {
84.   Ptr<Packet> packet;
85.   Address senderAddress;
86.   while ((packet = socket->RecvFrom (senderAddress)))
87.     {
88.       bytesTotal += packet->GetSize ();
89.       packetsReceived += 1;
90.       NS_LOG_UNCOND (PrintReceivedPacket (socket, packet, senderAddress));
91.     }
92. }
93.
94. void
95. RoutingExperiment::CheckThroughput ()
96. {
97.   double kbs = (bytesTotal * 8.0) / 1000;
98.   bytesTotal = 0;
99.
100.         std::ofstream out (m_CSVfileName.c_str (), std::ios::app);
101.
102.         out << (Simulator::Now ()).GetSeconds () << ","
103.             << kbs << ","
104.             << packetsReceived << ","
105.             << m_nSinks << ","
106.             << m_protocolName << ","
107.             << m_txp << ""
108.             << std::endl;
109.
110.         out.close ();
111.         packetsReceived = 0;
112.         Simulator::Schedule (Seconds (1.0), &RoutingExperiment::CheckThroughput, this
    );
113.       }
114.
115.       Ptr<Socket>
116.       RoutingExperiment::SetupPacketReceive (Ipv4Address addr, Ptr<Node> node)
117.       {
118.         TypeId tid = TypeId::LookupByName ("ns3::UdpSocketFactory");
119.         Ptr<Socket> sink = Socket::CreateSocket (node, tid);
120.         InetSocketAddress local = InetSocketAddress (addr, port);
121.         sink->Bind (local);
122.         sink-
    >SetRecvCallback (MakeCallback (&RoutingExperiment::ReceivePacket, this));
123.
124.         return sink;
125.       }
126.
127.       std::string
128.       RoutingExperiment::CommandSetup (int argc, char **argv)
```

```
129.       {
130.         CommandLine cmd;
131.         cmd.AddValue ("CSVfileName", "The name of the CSV output file name", m_CSVfil
    eName);
132.         cmd.AddValue ("traceMobility", "Enable mobility tracing", m_traceMobility);
133.         cmd.AddValue ("protocol", "1=OLSR;2=AODV;3=DSDV;4=DSR", m_protocol);
134.         cmd.Parse (argc, argv);
135.         return m_CSVfileName;
136.       }
137.
138.       int
139.       main (int argc, char *argv[])
140.       {
141.         RoutingExperiment experiment;
142.         std::string CSVfileName = experiment.CommandSetup (argc,argv);
143.
144.         //blank out the last output file and write the column headers
145.         std::ofstream out (CSVfileName.c_str ());
146.         out << "SimulationSecond," <<
147.         "ReceiveRate," <<
148.         "PacketsReceived," <<
149.         "NumberOfSinks," <<
150.         "RoutingProtocol," <<
151.         "TransmissionPower" <<
152.         std::endl;
153.         out.close ();
154.
155.         int nSinks = 4;
156.         double txp = 0.1;
157.
158.         experiment.Run (nSinks, txp, CSVfileName);
159.       }
160.
161.       void
162.       RoutingExperiment::Run (int nSinks, double txp, std::string CSVfileName)
163.       {
164.         Packet::EnablePrinting ();
165.         m_nSinks = nSinks;
166.         m_txp = txp;
167.         m_CSVfileName = CSVfileName;
168.
169.         int nWifis = 50;
170.
171.         std::string rate ("2048bps");
172.         std::string phyMode ("DsssRate11Mbps");
173.         std::string tr_name ("p2p_manet");
174.         int nodeSpeed = 1; //in m/s
175.         int nodePause = 0; //in s
176.         m_protocolName = "AODV";
177.         string line;
178.         std::string LinkRate ("1Mbps");
179.         std::string LinkDelay ("2ms");
180.         vector<NodeContainer> adhocNodesArray;
181.
182.         Config::SetDefault  ("ns3::OnOffApplication::PacketSize",StringValue ("1024")
    );
183.         Config::SetDefault ("ns3::OnOffApplication::DataRate",  StringValue (rate));

184.
185.         PointToPointHelper p2p;
186.         p2p.SetDeviceAttribute ("DataRate", StringValue (LinkRate));
187.         p2p.SetChannelAttribute ("Delay", StringValue (LinkDelay));
188.
189.         InternetStackHelper internet;
190.
191.         Ipv4AddressHelper ipv4_n;
192.         ipv4_n.SetBase ("10.0.0.0", "255.255.255.252");
193.
```

```
194.          AodvHelper aodv;
195.          Ipv4ListRoutingHelper list;
196.          list.Add (aodv, 100);
197.
198.          internet.SetRoutingHelper (list);
199.
200.
201.          MobilityHelper mobilityAdhoc;
202.          Ipv4InterfaceContainer adhocInterfaces;
203.
204.          std::cout << "START OF READING FILE\n";
205.
206.          // Reading file
207.            int playersNumber;
208.            ifstream in;
209.            in.open("network_with_nodes.txt");
210.
211.            in >> playersNumber;
212.
213.            NodeContainer adhocNode;
214.
215.          vector<int> playersNodes;
216.            // For each player reading nodes coordinates and adjacency matrix
217.            for (int player = 0; player < playersNumber; player++)
218.            {
219.                std::cout << "PLAYER " << player << endl;
220.                adhocNodesArray.push_back(adhocNode);
221.
222.                cout << "ALL GOOD 1\n";
223.                int nodesNumber;
224.                // Number of nodes
225.                in >> nodesNumber;
226.            playersNodes.push_back(nodesNumber);
227.                cout << "ALL GOOD 1.5\n";
228.                adhocNodesArray[player].Create(nodesNumber);
229.            internet.Install(adhocNodesArray[player]);
230.                cout << "ALL GOOD2\n";
231.
232.                Ptr<ListPositionAllocator> positionAlloc = CreateObject <ListPositionAl
    locator>();
233.
234.                cout << "ALL GOOD3\n";
235.                // Nodes coordinates
236.                for (int node = 0; node < nodesNumber; node++)
237.                {
238.                    double x, y, z;
239.                    in >> x >> y >> z;
240.
241.                    cout << x << ' ' << y << ' ' << z << endl;
242.                positionAlloc ->Add(Vector(x, y, 0));
243.                }
244.
245.            cout << "READING ADJ MATRIX\n";
246.                // Adjacency matrix
247.                for (int i = 0; i < nodesNumber; i++)
248.                {
249.                    vector<double> adjMatrix;
250.                    for (int j = 0; j < nodesNumber; j++)
251.                    {
252.                        double coord;
253.                        in >> coord;
254.                cout <<coord << " ";
255.                if (coord == 1)
256.                {
257.                  NodeContainer n_links = NodeContainer (adhocNodesArray[player].Get (i
    ), adhocNodesArray[player].Get (j));
258.                  NetDeviceContainer n_devs = p2p.Install (n_links);
259.                  adhocInterfaces.Add(ipv4_n.Assign (n_devs));
```

```
260.                 ipv4_n.NewNetwork ();
261.               }
262.
263.                  }
264.             cout << endl;
265.
266.               }
267.               mobilityAdhoc.SetPositionAllocator(positionAlloc);
268.               mobilityAdhoc.SetMobilityModel ("ns3::ConstantPositionMobilityModel");

269.
270.               mobilityAdhoc.Install (adhocNodesArray[player]);
271.           }
272.
273.         Ipv4GlobalRoutingHelper::PopulateRoutingTables ();
274.
275.
276.         OnOffHelper onoff1 ("ns3::UdpSocketFactory",Address ());
277.         onoff1.SetAttribute ("OnTime", StringValue ("ns3::ConstantRandomVariable[Cons
    tant=1.0]"));
278.         onoff1.SetAttribute ("OffTime", StringValue ("ns3::ConstantRandomVariable[Con
    stant=0.0]"));
279.
280.         uint16_t port = 9;
281.         double TotalTime = 200.0;
282.
283.         for (int player = 0; player < playersNumber; player++)
284.         {
285.           cout << playersNodes[player] << endl;
286.           for (int i = 0; i < playersNodes[player]-nSinks; i++)
287.             {
288.               cout << player <<" " << i << endl;
289.               Ptr<Socket> sink = SetupPacketReceive (adhocInterfaces.GetAddress (i),
    adhocNodesArray[player].Get (i));
290.
291.               AddressValue remoteAddress (InetSocketAddress (adhocInterfaces.GetAddre
    ss (i), port));
292.               onoff1.SetAttribute ("Remote", remoteAddress);
293.
294.               Ptr<UniformRandomVariable> var = CreateObject<UniformRandomVariable> ()
    ;
295.               ApplicationContainer temp = onoff1.Install (adhocNodesArray[player].Get
     (i + nSinks));
296.               temp.Start (Seconds (var->GetValue (10.0,11.0)));
297.               temp.Stop (Seconds (TotalTime));
298.             }
299.         };
300.
301.         std::stringstream ss;
302.         ss << nWifis;
303.         std::string nodes = ss.str ();
304.
305.         std::stringstream ss2;
306.         ss2 << nodeSpeed;
307.         std::string sNodeSpeed = ss2.str ();
308.
309.         std::stringstream ss3;
310.         ss3 << nodePause;
311.         std::string sNodePause = ss3.str ();
312.
313.         std::stringstream ss4;
314.         ss4 << rate;
315.         std::string sRate = ss4.str ();
316.
317.         AsciiTraceHelper ascii;
318.         MobilityHelper::EnableAsciiAll (ascii.CreateFileStream (tr_name + ".mob"));
319.
320.         cout << "FLOWMONITOR INITIALIZE\n";
```

```
321.          Ptr<FlowMonitor> flowmon;
322.          FlowMonitorHelper flowmonHelper;
323.          flowmon = flowmonHelper.InstallAll ();
324.
325.
326.          NS_LOG_INFO ("Run Simulation.");
327.
328.          CheckThroughput ();
329.
330.          Simulator::Stop (Seconds (TotalTime));
331.          Simulator::Run ();
332.
333.          flowmon->SerializeToXmlFile ((tr_name + ".flowmon").c_str(), false, false);
334.
335.          Simulator::Destroy ();
336.
337.        }
```