Работа допущена к защите

Зав. Кафедрой

\_\_\_\_\_\_\_\_ И.Г.Черноруцкий

“\_\_\_\_”\_\_\_\_\_\_\_\_\_\_\_2015 г.

**ВЫПУСКНАЯ РАБОТА БАКАЛАВРА**

**Тема: «Система моделирования mesh-сети на базе Bluetooth Low Energy»**

**Направление:** 230100.62 - Информатика и вычислительная техника.

Выполнил

студент гр. 43504/3 \_\_\_\_\_\_\_\_\_А.А. Сакуновский

Руководитель

доцент \_\_\_\_\_\_\_\_\_Т. В. Коликова

# Реферат

Работа содержит 72 страницы, 62 иллюстрации и 1 таблицу.

Работа посвящена разработке приложения-симулятора mesh-сети на устройствах Bluetooth Low Energy.

Основной задачей данного приложения является симуляция работы mesh-сети, получение её характеристик, тестирование стека протоколов. В качестве протокола передачи данных используется протокол реальной системы, импортированный в симулятор.

Оглавление

[Реферат 3](#_Toc421377411)

[Список иллюстраций 6](#_Toc421377412)

[Список таблиц 9](#_Toc421377413)

[Введение 10](#_Toc421377414)

[Актуальность работы 10](#_Toc421377415)

[Цели и задачи 12](#_Toc421377416)

[Краткое содержание работы 13](#_Toc421377417)

[1. Анализ предметной области 14](#_Toc421377418)

[BTool 14](#_Toc421377419)

[Сравнение MOBLE Simulator и BTool 14](#_Toc421377420)

[Требования к разрабатываемой системе 15](#_Toc421377421)

[Mesh-сети 16](#_Toc421377422)

[Bluetooth LE 17](#_Toc421377423)

[Конечный автомат состояний 18](#_Toc421377424)

[Состояния и события 20](#_Toc421377425)

[2. Архитектура и особенности реализации 26](#_Toc421377426)

[Диаграмма классов 26](#_Toc421377427)

[Описание классов 26](#_Toc421377428)

[Topology Manager 26](#_Toc421377429)

[Класс Packet 30](#_Toc421377430)

[Класс BLEdevice 32](#_Toc421377431)

[Класс SimMngr 36](#_Toc421377432)

[Процесс инициализации системы 38](#_Toc421377433)

[Импорт стека протоколов 38](#_Toc421377434)

[3. Получение и анализ характеристик сети 40](#_Toc421377435)

[Топология “линия” 41](#_Toc421377436)

[Количество узлов – 5 41](#_Toc421377437)

[Количество узлов – 10 44](#_Toc421377438)

[Количество узлов – 15 46](#_Toc421377439)

[Сводные графики 49](#_Toc421377440)

[Полносвязная топология 53](#_Toc421377441)

[Количество узлов – 5 53](#_Toc421377442)

[Количество узлов – 10 56](#_Toc421377443)

[Количество узлов – 15 58](#_Toc421377444)

[Количество узлов – 20 61](#_Toc421377445)

[Количество узлов – 30 63](#_Toc421377446)

[Сводные графики 66](#_Toc421377447)

[Заключение 71](#_Toc421377448)

[Список литературы 72](#_Toc421377449)

# Список иллюстраций

[Рисунок 1. Конечный автомат состояний 19](#_Toc421377349)

[Рисунок 2. Ответные PDU 21](#_Toc421377350)

[Рисунок 3. Схема advertising состояния 22](#_Toc421377351)

[Рисунок 4.Non-connectable advertising event 23](#_Toc421377352)

[Рисунок 5. Диаграмма классов 26](#_Toc421377353)

[Рисунок 6. Матрица видимости 28](#_Toc421377354)

[Рисунок 7. Время релаксации системы и время передачи ("линия"-5) 41](#_Toc421377355)

[Рисунок 8. Средняя нагрузка сети байт/мс ("линия" - 5) 42](#_Toc421377356)

[Рисунок 9. Средняя нагрузка сети пакеты/мс ("линия" - 5) 42](#_Toc421377357)

[Рисунок 10. Гистограмма времени релаксации системы ("линия" - 5) 43](#_Toc421377358)

[Рисунок 11. Гистограмма времени передачи ("линия" - 5) 43](#_Toc421377359)

[Рисунок 12. Время релаксации системы и время передачи ("линия"-10) 44](#_Toc421377360)

[Рисунок 13. Средняя нагрузка сети байт/мс ("линия" - 10) 45](#_Toc421377361)

[Рисунок 14. Средняя нагрузка сети пакеты/мс ("линия" - 5) 45](#_Toc421377362)

[Рисунок 15. Гистограмма времени релаксации системы ("линия" - 10) 45](#_Toc421377363)

[Рисунок 16. Гистограмма времени передачи ("линия" - 10) 46](#_Toc421377364)

[Рисунок 17. Время релаксации системы и время передачи ("линия"-15) 46](#_Toc421377365)

[Рисунок 18. Средняя нагрузка сети байт/мс ("линия" - 15) 47](#_Toc421377366)

[Рисунок 19. Средняя нагрузка сети пакеты/мс ("линия" - 15) 47](#_Toc421377367)

[Рисунок 20. Гистограмма времени релаксации системы ("линия" - 15) 48](#_Toc421377368)

[Рисунок 21. Гистограмма времени передачи ("линия" - 15) 48](#_Toc421377369)

[Рисунок 22. Сводный график времени релаксации системы ("линия") 49](#_Toc421377370)

[Рисунок 23. Сводный график времени передачи пакета ("линия") 49](#_Toc421377371)

[Рисунок 24. Сводный график средней нагрузки системы байт/мс (“линия”) 50](#_Toc421377372)

[Рисунок 25. Сводный график средней нагрузки сети пакеты/мс ("линия") 50](#_Toc421377373)

[Рисунок 26. Аппроксимация зависимости времени релаксации системы от количества узлов в системе ("линия") 51](#_Toc421377374)

[Рисунок 27. Аппроксимация зависимости времени передачи пакета от количества узлов в системе ("линия") 51](#_Toc421377375)

[Рисунок 28. Аппроксимация зависимости нагрузки сети от количества узлов в сети ("линия") 52](#_Toc421377376)

[Рисунок 29. Аппроксимация зависимости нагрузки сети от количества узлов в сети ("линия") 52](#_Toc421377377)

[Рисунок 30. Время релаксации системы и время передачи (полносвязная-5) 53](#_Toc421377378)

[Рисунок 31. Средняя нагрузка сети байт/мс (полносвязная - 5) 54](#_Toc421377379)

[Рисунок 32. Средняя нагрузка сети пакеты/мс (полносвязная - 5) 54](#_Toc421377380)

[Рисунок 33. Гистограмма времени релаксации системы (полносвязная - 5) 55](#_Toc421377381)

[Рисунок 34. Гистограмма времени передачи (полносвязная - 5) 55](#_Toc421377382)

[Рисунок 35. Время релаксации системы и время передачи (полносвязная-10) 56](#_Toc421377383)

[Рисунок 36. Средняя нагрузка сети байт/мс (полносвязная - 10) 56](#_Toc421377384)

[Рисунок 37. Средняя нагрузка сети пакеты/мс (полносвязная - 10) 57](#_Toc421377385)

[Рисунок 38. Гистограмма времени релаксации системы (полносвязная - 10) 57](#_Toc421377386)

[Рисунок 39. Гистограмма времени передачи (полносвязная - 10) 58](#_Toc421377387)

[Рисунок 40. Время релаксации системы и время передачи (полносвязная-15) 58](#_Toc421377388)

[Рисунок 41. Средняя нагрузка сети байт/мс (полносвязная - 15) 59](#_Toc421377389)

[Рисунок 42. Средняя нагрузка сети пакеты/мс (полносвязная - 15) 59](#_Toc421377390)

[Рисунок 43. Гистограмма времени релаксации системы (полносвязная - 15) 60](#_Toc421377391)

[Рисунок 44. Гистограмма времени передачи (полносвязная - 15) 60](#_Toc421377392)

[Рисунок 45. Время релаксации системы и время передачи (полносвязная-20) 61](#_Toc421377393)

[Рисунок 46. Средняя нагрузка сети байт/мс (полносвязная - 20) 61](#_Toc421377394)

[Рисунок 47. Средняя нагрузка сети пакеты/мс (полносвязная - 20) 62](#_Toc421377395)

[Рисунок 48. Гистограмма времени релаксации системы (полносвязная - 20) 62](#_Toc421377396)

[Рисунок 49. Гистограмма времени передачи (полносвязная - 20) 63](#_Toc421377397)

[Рисунок 50. Время релаксации системы и время передачи (полносвязная-30) 63](#_Toc421377398)

[Рисунок 51. Средняя нагрузка сети байт/мс (полносвязная - 30) 64](#_Toc421377399)

[Рисунок 52. Средняя нагрузка сети пакеты/мс (полносвязная - 30) 64](#_Toc421377400)

[Рисунок 53. Гистограмма времени релаксации системы (полносвязная - 30) 65](#_Toc421377401)

[Рисунок 54. Гистограмма времени передачи (полносвязная - 30) 65](#_Toc421377402)

[Рисунок 55. Сводный график времени релаксации системы (полносвязная) 66](#_Toc421377403)

[Рисунок 56. Сводный график времени передачи пакета (полносвязная) 66](#_Toc421377404)

[Рисунок 57. Сводный график средней нагрузки системы байт/мс 67](#_Toc421377405)

[Рисунок 58. Сводный график средней нагрузки системы пакеты/мс 67](#_Toc421377406)

[Рисунок 59. График зависимости среднего времени релаксации от количества узлов в сети (полносвязная) 68](#_Toc421377407)

[Рисунок 60. График среднего времени передачи пакета (полносвязная) 68](#_Toc421377408)

[Рисунок 61. Аппроксимация зависимости нагрузки сети от количества узлов в сети (полнносвязная) 69](#_Toc421377409)

[Рисунок 62. Аппроксимация зависимости нагрузки сети от количества узлов в сети (полносвязная) 69](#_Toc421377410)

# Список таблиц

[Таблица 1. Сравнение систем BTool и MOBLE Simulator 14](#_Toc421202903)

# Введение

## Актуальность работы

На данный момент, наиболее встречающимся словосочетанием в IT-публикациях является Internet Of Things (“Интернет вещей”). Точного определения данной технологии нет, но, как известно, любая технология создается для автоматизации процессов окружающей человека среды. С развитием информационных технологий, увеличивается число устройств, с помощью которых, человек получает необходимую ему информацию. Человек может непрерывно находиться в сети и участвовать в информационном обмене, получая данные из разных уголков мира или от датчиков в соседней комнате, управлять удаленными объектами. Таким образом, интернет вещей – это концепция сети физических устройств, со встроенными технологиями передачи данных между собой и сетью Internet.

Большинство технологий, относящихся к интернету вещей, основано на получении и анализе данных от датчиков c автономным питанием, непрерывно передающих информацию о состоянии того или иного объекта. В этом случае важным является вопрос энергопотребления. Устройства, оснащенные беспроводной технологией Bluetooth Low Energy, позволяют решить проблему высокого энергопотребления. Устройства, использующие данную технологию, потребляют меньше энергии, чем устройства Bluetooth предыдущих поколений. Многие устройства могут работать более года на миниатюрной батарейке без подзарядки. Это делает возможным создание сети из небольших датчиков, работающих непрерывно.

При разработке аппаратно-программного комплекса решений для систем, относящихся к концепции интернета вещей, таких как система освещения умного дома, требуются огромные затраты на тестирование (закупка определенного количества датчиков, например), что вызывает некоторые неудобства. Данная проблема встречается повсеместно. В таких случаях, во избежание затрат на натурные испытания, используют системы моделирования. Но не всегда готовые продукты удовлетворяют всем требованиям разработчиков. В этом случае проводятся работы по созданию собственной системы моделирования.

## Цели и задачи

Проект MOBLE (Mesh Over Bluetooth Low Energy) включает в себя программное и аппаратное решение для автоматизации управления освещением в системе “умный дом”. Каждый светоизлучающий прибор в помещении содержит устройство на безе технологии Bluetooth Low Energy. Таким образом, с помощью Bluetooth устройств, световые приборы могут обмениваться информацией о своем состоянии и командами между собой, и с управляющим устройством, например, смартфоном.

Целью работы является создание системы моделирования mesh-сети на базе устройств Bluetooth Low Energy, использующей реальный стек протоколов, осуществляющей подсчет характеристик mesh-сети.

Для достижения поставленной цели необходимо выполнить следующие задачи:

* Анализ существующих на рынке систем.
* Определение требований к симулятору.
* Изучение технологии Bluetooth Low Energy.
* Реализация симулятора.
* Получение и анализ характеристик mesh-сети.

## Краткое содержание работы

Работа содержит четыре части.

В первой части проводится анализ существующих на рынке аналогичных продуктов, формулируются требования к разрабатываемой системе, проводится ознакомление с понятием mesh-сеть и технологией Bluetooth.

Во второй части идет речь об архитектуре системы. Представлена диаграмма классов, их описание, процесс инициализации системы, особенности взаимодействия со стеком протоколов.

В третьей части представлены графики зависимостей характеристик сети и их анализ.

В заключении подводятся итоги работы, формулируются возможные пути развития симулятора, перспективы.

# Анализ предметной области

Ведущие производители чипов Bluetooth предоставляют программные средства для тестирования своих продуктов. Но, как правило, поставляемое программное обеспечение ориентировано на тестирование конкретного устройства. В качестве примера, рассмотрим приложение BTool от Texas Instruments.

## BTool

BTool – приложение, позволяющее пользователю устанавливать соединение с Bluetooth Low Energy устройством под видом “Master”-устройства (роли, участвующих в соединении устройств будут рассмотрены в разделе Bluetooth LE). Взаимодействие между компьютером и BLE устройством происходит через Windows COM порт или USB Dongle (USB-адаптер Bluetooth).

BTool предоставляет пользователю следующие возможности:

* Поиск доступных устройств
* Создание соединения с найденными устройствами
* Установление параметров соединения
* Чтение и запись доступных GATT(Generic Attribute Profile)-характеристик
* Использование доступных GATT сервисов (Battery service, accelerometer service и т.д.)

## Сравнение MOBLE Simulator и BTool

Как было сказано выше, приложение BTool ориентировано на тестирование реального устройства и его стека протоколов. MOBLE Simulator ориентирован на тестирование модели сети устройств с одинаковой конфигурацией и стеком протоколов.

|  |  |  |
| --- | --- | --- |
| Свойство | BTool | MOBLE Simulator |
| Поддержка Connection State (Конечный автомат состояний устройства будет рассмотрен в разделе Bluetooth LE) | Присутствует | Отсутствует |
| Конфигурирование топологии сети (задание количества устройств, задание полей видимости устройств) | Отсутствует | Присутствует |
| Взаимодействие с реальным устройством | Присутствует | Отсутствует |
| Сбор характеристик сети (нагрузка, время релаксации, время передачи пакета и т.д.) | Отсутствует | Присутствует |
| Задание сценария работы устройств (расписание отсылки пакетов) | Отсутствует | Присутствует |

Таблица 1. Сравнение систем BTool и MOBLE Simulator

Продукт BTool не подходит для достижения поставленных целей. Таким образом, возникает необходимость в создании собственной системы.

## Требования к разрабатываемой системе

* Возможность конфигурирования топологии сети
* Импорт стека протоколов без его последующего изменения
* Реализация сбора характеристик сети (время релаксации, время передачи, средняя нагрузка)
* Автоматическое определение модельного времени, необходимого для получения данных характеристик
* Задание размера выборки характеристик

## Mesh-сети

Mesh-сеть (или ячеистая сеть) является одной из топологий сети, чаще всего используемой в LAN, при которой каждое устройство соединено с несколькими другими устройствами (присуща высокая избыточность связей), может передавать данные, быть использованным в качестве промежуточного узла передачи данных, брать на себя функции коммутатора. Особенность архитектуры mesh-сети обеспечивает данный вид сетей такими свойствами как: самоорганизация, отказоустойчивость, масштабируемость. Mesh-сети являются децентрализованными и не имеют постоянной структуры. Рабочие станции могут выходить из строя, некоторые рабочие станции могут быть мобильными и менять своё местоположение (предполагается, что смена местоположения будет не частой). Но непостоянность структуры не окажет критического влияния на работу сети. В зависимости от используемых протоколов маршрутизации, данные могут быть перенаправлены через другие доступные узлы. Локальные mesh-сети, благодаря свойству масштабируемости, легко интегрируются в городские, а городские, в свою очередь, в глобальную сеть. Типичная схема построения mesh-сети выглядит следующим образом: территория покрытия делится на участки, соединенные между собой электрическим проводом или оптоволокном; в каждом участке установлены рабочие станции, связь между которыми обеспечивается через радиоканал; Одна или несколько рабочих станций подключена к магистральному каналу и выполняет функции узловой. Mesh-сети рассчитаны на частично или полностью фиксированную инфраструктуру.

## Bluetooth LE

Беспроводная технология Bluetooth – коммуникационная система для передачи данных на небольшие расстояния, призванная заменить подключение стационарных или мобильных устройств посредством кабеля.

Существует две версии беспроводной технологии Bluetooth: Basic Rate (BR) и Low Energy (LE). Обе версии оснащены возможностью поиска устройств в диапазоне видимости, возможностью устанавливать соединения с найденными устройствами и механизмом установления соединений. Система BR опционально включает в себя расширения управления доступом к среде и физического уровня (Alternate Media Access Control/PHY, AMP), повышенную скорость передачи данных (Enhanced Data Rate). BR система осуществляет возможность синхронного и асинхронного подключения со скоростью передачи данных 721.2 kbps для стандартной версии, 2.1 Mbps с EDR и до 24 Mbps совместно с 802.11 AMP. LE системы включают в себя функционал для работы с простыми и недорогими устройствами, с более низким уровнем электропотребления нежели BR/EDR системы.

Как и BDR/EDR системы, LE системы работают в диапазоне 2.4GHz ISM, используют метод расширения спектра со скачкообразной перестройкой частот (FHSS) для уменьшения интерференции и воздействий со стороны устройств, работающих рядом.

В LE используется Gaussian Frequency Shift Keying modulation с параметрами BT (отношение полуширины полосы частот к скорости передачи символов) равным 0.5 и индексом модуляции находящимся в диапазоне от 0.45 до 0.55.

Девиация сигнала при входной последовательности 1010 должна быть не менее чем 80% от девиации сигнала при входной последовательности 00001111.

В LE используются две схемы контроля доступа к среде: FDMA (Frequency Division Multiple Access) и TDMA (Time Division Multiple Access). В FDMA схеме используются 40 каналов разделенные промежутками по 2 MHz: 3 “advertising” канала и 37 “data” каналов (2402+k\*2MHz, где k=0,…,39). TDMA схема используется, если устройства отправляют пакеты в предопределенное время.

### Конечный автомат состояний

Сетевой уровень может находиться в одном из пяти состояний:

* Standby State. В данном состоянии устройство не посылает и не принимает пакеты. Войти в данное состояние можно из любого другого.
* Advertising State. В данном состоянии устройство передаёт пакеты по advertising каналам и слушает ответы. В данное состояние можно войти из Standby State.
* Scanning State. В данном состоянии устройство прослушивает advertising каналы с целью принять пакеты от устройств, находящихся в Advertising State. В данное состояние можно войти из Standby State.
* Initiating State.В данном состоянии устройство прослушивает advertising каналы, с целью принять пакет от определенного устройства и отправить ответ для установления соединения. Войти в данное состояние можно из Standby State.
* Connection State. Войти в данное состояние можно из Initiator State или из Advertising State. Если войти Connection State из Initiator State, то устройство будет выполнять роль master устройства (ведущее устройство). Если войти Connection State из Advertising State, то устройство будет выполнять роль slave (ведомое устройство) устройства.

В каждый момент времени устройство может находиться только в одном состоянии. Опционально возможна поддержка нескольких состояний.

В MOBLE Simulator реализованы Advertising, Scanning и Standby состояния. Остальные состояния не являются необходимыми для достижения текущих целей.

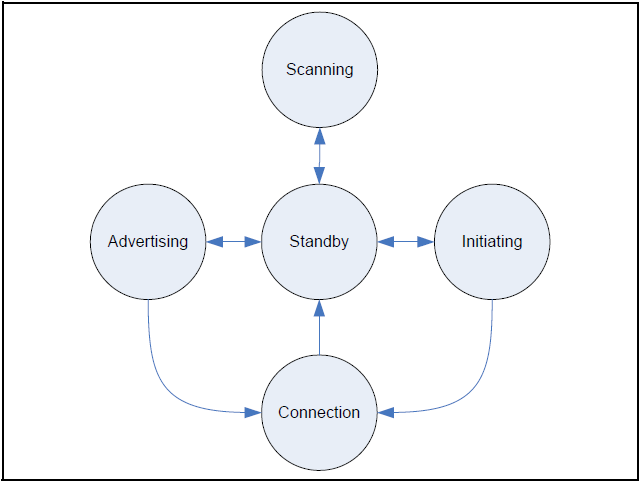


Рисунок 1. Конечный автомат состояний

### Состояния и события

Передача пакетов по каналам производится в промежутки времени, названные событиями (Events). Всего два типа событий: Connection и Advertising events.

Устройства, посылающие пакеты по advertising каналам, называют “advertisers”. Устройства, принимающие пакеты по advertising каналам называют “scanners”. Передача пакетов по advertising каналам происходит во время advertising events. В начале каждого advertise event, advertiser посылает advertise пакет в соответствии с типом advertise event. В зависимости от типа advertise пакета, “scanner” может послать запрос устройству, от которого был принят advertise пакет, по тому же физическому каналу. В ответ на запрос возможен ответ от advertiser по тому же физическому каналу. Смена канала происходит с отправкой следующего пакета в том же advertise event.

Рассмотрим состояния устройств и возможные события более подробно.

#### Advertising состояние

Находясь в advertising состоянии, устройство посылает advertise пакет во время advertise event. В каждом advertise event может быть от одного до трёх PDU (Packet Data Unit, пакет). Advertise event должен завершиться после того, как по каждому из advertise каналов было отправлено по одному пакету, или ранее. Существуют следующие типы advertise event: connectable undirected event, connectable directed event, non-connectable undirected event, scannable undirected event. Тип advertise event определяет допустимые типы ответа. Таблица ниже демонстрирует возможные ответы для каждого типа advertise event:

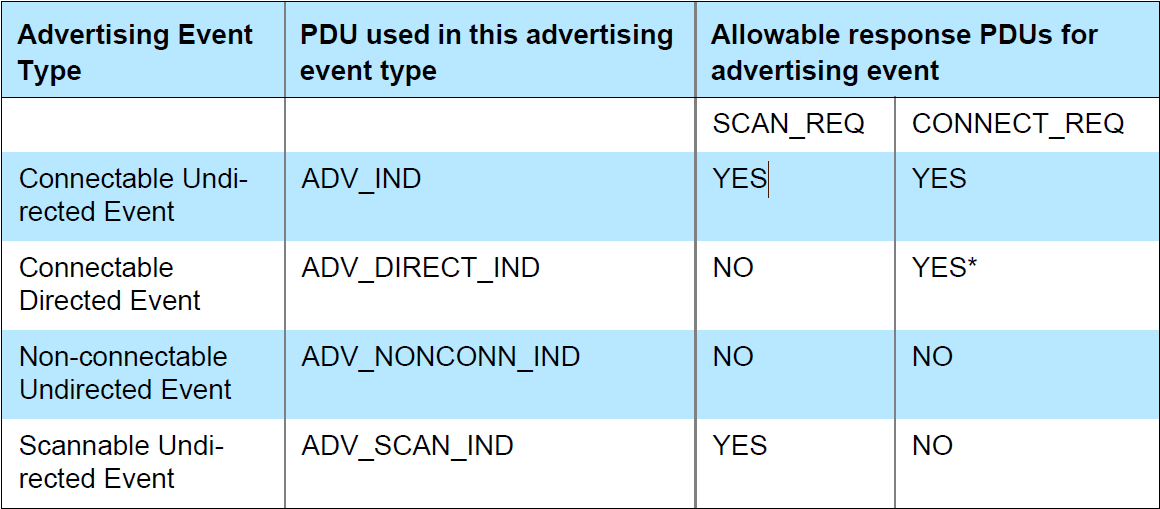


Рисунок 2. Ответные PDU

Как видно из иллюстрации – Non-connectable Undirected Event не требует ответа. Именно этот тип advertise события используется в MOBLE Simulator для достижения поставленных задач.

Для всех undirected advertise events, время между соседними advertise event (T\_advEvent) рассчитывается следующим образом:

T\_advEvent=advInterval+advDelay

advInterval – число, делящееся на 0.625, в диапазоне от 20ms до 10.24s. Если тип advertise event scannable undirected event или non-connectable undirect event, advInterval не должен быть меньше 100ms.

advDelay – псевдо случайное значение в диапазоне от 0 до 10ms, генерируемое для каждого advertise event.

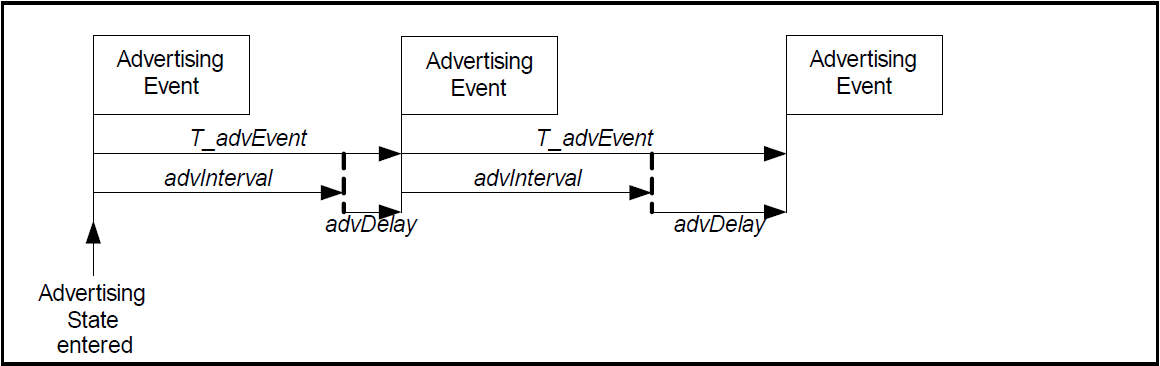


Рисунок 3. Схема advertising состояния

##### Non-connectable Undirected Event Type

При использовании non-connectable undirected event посылаются ADV\_NONCONN\_IND PDU.

Non-connectable undirected event type позволяет scanner-у принять информацию от advertiser-а.

После отправки каждого ADV\_NONCONN\_IND PDU advertiser должен сменить канал и отправить следующий ADV\_NONCONN\_IND, или закрыть advertising event.

Интервал между соседними ADV\_NONCONN\_IND PDU должен быть меньше или равен 10мс.

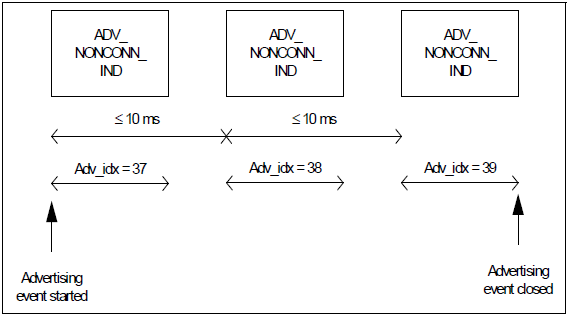


Рисунок 4.Non-connectable advertising event

#### Scanning состояние

Устройство переходит в scanning состояние по запросу хоста. В scanning состоянии устройство должно слушать advertising каналы. Существует два режима сканирования: активный (active) и пассивный (passive).

Во время сканирования, устройство слушает advertising канал на протяжении окна сканирования, scanWindow. ScanInterval – расстояние между началами двух соседних окон сканирования.

Устройство должно слушать advertise канал на протяжении scanWindow каждый scanInterval. В каждом окне сканирования устройство должно прослушивать разные advertising каналы.

Параметры scanWindow и scanInterval должны быть меньше или равны 10.24 с. ScanWindow должен быть меньше или равен scanInterval. Если хост установит значения параметров scanWindow и scanInterval равными, то устройство будет сканировать непрерывно.

Для каждого уникального ADV\_DIRECT\_IND PDU, ADV\_IND PDU, ADV\_SCAN\_IND PDU, ADV\_NONCONN\_IND PDU или SCAN\_RSP PDU устройство должно отправить advertising report (отчет) с соответствующей информацией хосту.

Passive Scanning – режим сканирования, при котором устройство только принимает пакеты.

Active Scanning – режим сканирования, при котором устройство принимает пакты и, в зависимости от типа PDU, может отправить соответствующие запросы. (данный вид сканирования не реализован в MOBLE Simulator по причине отсутствия необходимости в достижении задач симулятора)

Устройство в scanning состоянии должно отправить хотя бы один SCAN\_REQ PDU advertiser-ам, от которых были приняты ADV\_IND или ADV\_SCAN\_IND PDU.

Ниже в качестве ознакомления представлены состояния сетевого уровня Bluetooth Low Energy, не реализованные в текущей версии симулятора. Реализация данных состояний является одним из направлений развития проекта. О перспективах MOBLE Simulator будет рассказано подробнее в разделе Заключение.

#### Initiating состояние

В initiating состояние устройство входит по запросу хоста. Устройство в initiating состоянии прослушивает advertising каналы c целью установки соединения.

Если были приняты ADV\_IND или ADV\_DIRECT\_IND PDU, устройство должно отправить CONNECT\_REQ PDU advertiser-ам. После отправки CONNECT\_REQ PDU, устройство выходит из initiating состояния и входит в connection состояние в качестве master устройства.

#### Connection состояние

Устройство входит в connection состояние после того как initiator отправил CONNECT\_REQ PDU advertiser-у или после того как advertiser принял CONNECT\_REQ PDU от initiator-а. После входа в Connection состояние соединение считается созданным, но не установленным. Соединение считается установленным после передачи пакета data каналу.

Когда два устройства соединены, они выполняют две разные роли: master и slave.

# Архитектура и особенности реализации

## Диаграмма классов

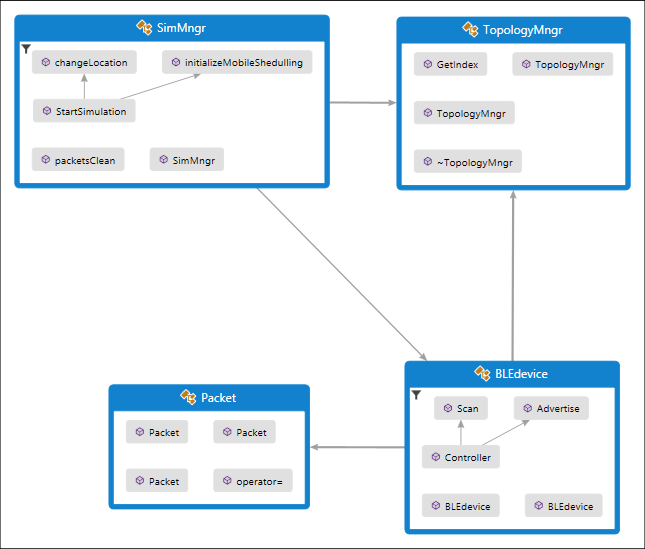


Рисунок 5. Диаграмма классов

## Описание классов

### Topology Manager

struct Groups

{

int GNA;

std::vector<int> devices;

};

struct MobileShedulling

{

int NA;

std::vector<int> rout;

std::vector<double> shedulling;

double boarder;

int lastIndex;

};

class TopologyMngr{

public:

int deviceQuantity;

int mobileDeviceQuantity;

int groupQuantity;

int N;

int M;

std::vector<std::vector<double>> vision\_propabilityMatrix;

std::vector<Groups> groupStack;

std::vector<MobileShedulling> mobileShed;

TopologyMngr(char\* fname);

int GetIndex(int na);

};

Класс TopologyMngr отвечает за хранение информации о топологии сети.

deviceQuantity – количество устройств в сети (без учета мобильных).

mobileDeviceQuantity – количество мобильных устройств в сети.

groupQuantity – количество групп устройств.

N – групповая метрика сети (максимум групп может существовать в данной сети).

M – метрика сети по количеству устройств (максимум устройств может быть зарегистрировано в данной сети ).

vision\_propabilityMatrix – матрица видимости устройств. В данной матрице каждая строка и столбец соответствует устройству в сети. Ненулевые значения в матрице означают, что устройство с уникальным идентификатором (адресом в сети), соответствующим номеру столбца, находится в зоне видимости (досягаемости) для устройства с уникальным идентификатором, соответствующим номеру строки. Для наглядности рассмотрим иллюстрацию:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 5 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 8 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 |
| 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

Рисунок 6. Матрица видимости

Ненулевое значение в ячейке (4,3) говорит о том, что устройство с адресом 3 находится в поле видимости (досягаемости) устройства с адресом 4.

groupStack – контейнер, хранящий информацию о группах устройств в виде структур Groups. Структура Groups состоит из двух полей: GNA – групповой адрес, devices – список адресов устройств, входящих в данную группу.

mobileShed – контейнер, хранящий информацию о мобильных устройствах в виде структур MobileShedulling. Перемещаясь, мобильные устройства изменяют свою область видимости. В данной модели предполагается, что мобильные устройства передвигаются между статических устройств, и в каждый момент времени зона видимости мобильного устройства, совпадает с зоной видимости одного из статических устройств. На иллюстрации можно заметить ненулевые значения на диагонали, что не совсем верно, исходя из представленной выше логики работы с матрицей видимости (устройство видит само себя, а значит, может послать пакет самому себе). Эти значения необходимы для корректного обмена информацией между мобильными устройствами. Рассмотрим ситуацию, когда в момент передачи пакета от одного мобильного устройства другому, они оказались в одной точке. Тогда, исходя из матрицы видимости, в случае обнаружения нуля, устройства не смогли бы друг друга видеть. В случае передачи статическим устройством, значения на диагонали не учитываются.

Структура MobileShedulling состоит из пяти полей: NA – уникальный адрес мобильного устройства; rout – контейнер, хранящий маршрут мобильного устройства в виде адресов статических устройств; shedulling – контейнер, содержащий время задержки на каждой из точек маршрута; boarder – временная метка смены местоположения; lastIndex – индекс последней точки маршрута в контейнере.

TopologyMngr(char\* fname) – конструктор класса, принимающий на вход строку с названием конфигурационного файла.

int GetIndex(int na) – функция, принимающая на вход адрес устройства в сети, возвращает индекс этого устройства в контейнере mobileShed.

### Класс Packet

class Packet{

public:

int ID;

double initialSendTime;

double sendTime;

int chanel;

double timeToReceive;

int Length;

MOBLEUINT8 data[MOBLE\_MAX\_ADVSCAN];

int originatorNA;

int destinationNA;

int senderNA;

Packet(int Id, int len);

Packet();

};

struct PacketDest

{

Packet packet;

std::vector<int> devices;

int initialDevices;

};

Класс Packet хранит информацию о пакете, необходимую для корректной работы симулятора, и подсчета характеристик.

ID – уникальный идентификатор пакета в симуляторе.

initialSendTime – время отправки пакета инициатором передачи данных (оригинальный пакет).

sendTime – время отправки пакета.

channel – номер канала, по которому был отправлен пакет.

timeToReceive – время возможного получения пакета.

Length – длина пакета в байтах.

data – блок данных для передачи, формируемый стеком протоколов.

originatorNA – адрес инициатора передачи.

destinationNA – адрес получателя.

senderNA – адрес последнего отправителя.

Packet(int Id, int Len) – конструктор класса, принимающий на вход уникальный идентификатор и длину пакета.

Packet() – конструктор класса по умолчанию.

Структура PacketDest является основной единицей передачи данных внутри симулятора. Структура состоит из трех полей: packet – пакет; devices – контейнер, хранящий адреса устройств, которым этот пакет виден; initialDevices – количество устройств, которым этот пакет виден.

При отправке пакета, устройство заполняет данную структуру соответствующим пакетом, списком устройств из области досягаемости (матрица vision\_propabilityMatrix), и заполняет поле initialDevices числом, равным количеству видимых устройств. При получении пакета, устройство удаляет свой адрес из initialDevices.

### Класс BLEdevice

class BLEdevice{

public:

double\* Time;

double advInterval;

double delay;

double sendInterval;

double scanWindow;

double scanInterval;

double transmissionTime;

double startTime;

double becomeFreeTime;

double startScanInt;

bool startScanFlag;

int NA;

int GNA;

bool isMobile;

bool isActive;

int chanels[3];

int lastScanIndex;

int mobileIndex;

int status;

std::vector<Packet> receivedPackets;

std::vector<PacketDest>\* pcktD;

std::vector<BLEdevice\*>\* dev;

TopologyMngr\* Tplgmngr;

MOBLE\_NETWORK\_CONFIGURATION1 config;

HANDLE\_MOBLE handle;

MOBLE\_DATA\_MAP dataMap;

BLEdevice(double tT, int nA, int gnA, int fGNA, bool isM, double\* time, TopologyMngr\* t, std::vector<PacketDest>\* pcktD, std::vector<BLEdevice\*>\* dev);

BLEdevice();

void Advertise(TopologyMngr\*t, td::vector<PacketDest>\* p, std::vector<BLEdevice\*> d);

void Scan(std::vector<PacketDest>\* p,TopologyMngr\* t);

void Controller();

Класс BLEdevice хранит информацию об устройстве (конфигурация, текущее состояние и т.д.), а также описывает его функционал.

Time – ссылка на переменную, хранящую текущее модельное время.

advInterval – advertise interval.

delay – случайная задержка между advertise интервалами.

sendInterval – интервал между двумя ближайшими пакетами в пределах одного advertise события.

scanWindow – окно сканирования.

scanInterval – интервал сканирования.

transmissionTime – время передачи пакета между ближайшими узлами.

startTime – время включения устройства.

becomeFreeTime – время завершения текущего advertise события.

startScanInt – временная метка начала последнего интервала сканирования.

startScanFlag – флаг обновления переменной startScanInt.

NA – адрес устройства.

GNA – групповой адрес устройства.

isMobile – флаг мобильности устройства.

isActive – флаг активности устройства.

chanels – массив индексов каналов передачи данных.

lastScanIndex – индекс последнего использованного канала при сканировании.

mobileIndex – индекс текущей точки маршрута мобильного устройства.

status – текущий статус активности устройства.

receivedPackets – контейнер, хранящий принятые устройством пакеты.

pcktD – ссылка на контейнер, хранящий все отправленные устройствами пакеты.

dev – ссылка на контейнер, хранящий ссылки на устройства в сети.

Tplgmngr – ссылка на объект класса TopologyMngr.

config – структура, хранящая конфигурацию устройства.

handle – указатель на область данных стека, привязанную к данному устройству.

Исполняемый код стека протоколов использует исходные данные устройства, по чьей инициативе было вызвано исполнение тех или иных функций стека протоколов. Таким образом, реализуется система, в которой некоторое количество объектов класса BLEdevice используют один и тот же стек протоколов, вызывая его функции с указателем на свои исходные данные.

dataMap – структура хранения данных устройства (этими данными устройства обмениваются в процессе работы системы).

BLEdevice(double aI, double sI, double sW, double scI, double pT, double tT, int nA, int gnA,int fGNA, bool isM, double\* time, TopologyMngr\* t, std::vector<PacketDest>\* pcktD, std::vector<BLEdevice\*>\* dev) – перегруженный конструктор класса BLEdevice.

BLEdevice() – конструктор класса BLEdevice по умолчанию.

Advertise(TopologyMngr\* t,std::vector<PacketDest>\* p, std::vector<BLEdevice\*> d) – функция, выполняющая передачу данных.

Scan(std::vector<PacketDest>\* p, TopologyMngr\* t) – функция, выполняющая сканирование.

Controller() – функция, определяющая текущее действие устройства.

Часть функционала данного класса скрыта политикой конфиденциальности.

### Класс SimMngr

struct PacketGeneration

{

int NA;

int destNA;

int destGNA;

double time;

};

class SimMngr{

public:

int simulationState;

bool verifyState;

int packetsQuantity;

double transmissionTime;

std::vector<BLEdevice\*> deviceStack;

std::vector<PacketDest> packetStack;

std::vector<PacketGeneration> generation;

SimMngr(TopologyMngr T);

void StartSimulation(TopologyMngr T);

void packetsClean();

void changeLocation(double time, TopologyMngr\* t);

void initializeMobileShedulling(TopologyMngr\* t);

};

Класс SimMngr отвечает за проведение симуляции и подсчет характеристик.

simulationState – состояние симуляции (0 – передача пакетов из сценария не началась, 1 – началась передача пакетов из сценария, 2 – ни одно из устройств не передаёт пакетов из сценария).

verifyState – вспомогательная переменная для определения состояния симуляции.

transmissionTime – время передачи пакета между узлами из области видимости.

packetsQuantity – количество пакетов в сценарии.

deviceStack – контейнер указателей на объекты типа BLEdevice.

packetStack – контейнер, хранящий все отправленные пакеты в формате структуры packetDest.

generation – контейнер, хранящий сценарий отсылки пакетов в формате структуры PacketGeneration.

Структура PacketGeneration состоит из трёх полей: NA – адрес отправителя, destNA – адрес получателя, destGNA – групповой адрес получателя, time – время поступления пакета в систему.

SimMngr(TopologyMngr T) – конструктор класса со входным параметром типа TopologyMngr.

packetsClean() – функция отчистки контейнера packetStack.

changeLocation(double time, TopologyMngr\* t) – функция смены местоположения мобильных устройств.

initializeMobileShedulling(TopologyMngr\* t) – установка начального положения мобильных устройств.

### Процесс инициализации системы

Процесс инициализации системы проходит в два этапа: инициализация топологии сети, инициализация устройств и сценария симуляции.

Инициализацию топологии сети проводит конструктор класса TopologyMngr. Для инициализации топологии используется файл config.txt, заполненный в определенном порядке: метрика сети N, метрика сети M, количество устройств, количество групп, матрица видимости устройств, список групп (формат: номер группы, количество устройств в группе, адреса устройств), количество мобильных устройств, список маршрутов для мобильных устройств (формат: количество точек в маршруте, адреса устройств-точек, значения задержек на каждой точке в соответствующем порядке).

Далее создаётся объект класса SimMngr конструктор которого, используя объект класса TopologyMngr, создает и инициализирует объекты класса BLEdevice.

Для инициализации сценария симулирования используется файл Shedull.txt, заполненный в определенном порядке: время передачи пакета между узлами из области видимости, количество пакетов, список пакетов (формат: адрес отправителя, группа получателя, адрес получателя, время появления пакета в системе).

## Импорт стека протоколов

Стек протоколов импортируется путем подключения к проекту соответствующих файлов исходного кода и заголовочных файлов.

В процессе функционирования системы, все объекты класса BLEdevice используют один и тот же программный код стека протоколов. Это возможно благодаря особенности реализации стека протоколов. Каждый объект класса BLEdevice хранит некоторую особую структуру данных, содержащую необходимую информацию для функционирования стека протоколов. В процессе работы системы, объекты класса BLEdevice сообщают стеку протоколов свою особую структуру данных, что позволяет идентифицировать устройство.

# Получение и анализ характеристик сети

В качестве примера работы программы предоставляется анализ работы системы на двух различных топологиях сети: “линия” и полносвязная топология. В топологии “линия” сеть представляет собой цепочку из устройств, область видимости которых ограничена лишь ближайшими соседями. Данная топология обеспечивает худшее время передачи пакета от первого устройства в цепочки до последнего. В полносвязной топологии все устройства “видят” друг друга. Данная топология обеспечивает наилучшее время доставки пакета от любого узла к любому узлу, но худшее время релаксации системы. Под временем релаксации системы в данном случае подразумевается время от начала передачи пакета каким-либо из устройств до момента, когда данный пакет пропадет из сети, т.е. ни одно из устройств не будет его передавать.

Текущая версия программы-симулятора производит подсчёт следующих характеристик: время релаксации системы, время передачи пакета, средняя нагрузка сети (пакеты в миллисекунду), средняя нагрузка сети (байты в миллисекунду). Измерения производятся при передаче одного пакета между самыми отдаленными узлами (в случае топологии “линия”); для полносвязной топологии не важно, между какими узлами осуществляется передача. Объем выборки задается пользователем на старте симуляции. В данном примере объем выборки равен 20.

Симулятор представляет данные в виде текста. Графический материал был создан с помощью приложения MS Excel.

## Топология “линия”

### Количество узлов – 5

Рисунок 7. Время релаксации системы и время передачи ("линия"-5)

Рисунок 8. Средняя нагрузка сети байт/мс ("линия" - 5)

Рисунок 9. Средняя нагрузка сети пакеты/мс ("линия" - 5)

Рисунок 10. Гистограмма времени релаксации системы ("линия" - 5)

Рисунок 11. Гистограмма времени передачи ("линия" - 5)

### Количество узлов – 10

Рисунок 12. Время релаксации системы и время передачи ("линия"-10)

Рисунок 13. Средняя нагрузка сети байт/мс ("линия" - 10)

Рисунок 14. Средняя нагрузка сети пакеты/мс ("линия" - 5)

Рисунок 15. Гистограмма времени релаксации системы ("линия" - 10)

Рисунок 16. Гистограмма времени передачи ("линия" - 10)

### Количество узлов – 15

Рисунок 17. Время релаксации системы и время передачи ("линия"-15)

Рисунок 18. Средняя нагрузка сети байт/мс ("линия" - 15)

Рисунок 19. Средняя нагрузка сети пакеты/мс ("линия" - 15)

Рисунок 20. Гистограмма времени релаксации системы ("линия" - 15)

Рисунок 21. Гистограмма времени передачи ("линия" - 15)

### Сводные графики

Рисунок 22. Сводный график времени релаксации системы ("линия")

Рисунок 23. Сводный график времени передачи пакета ("линия")

Рисунок 24. Сводный график средней нагрузки системы байт/мс (“линия”)

Рисунок 25. Сводный график средней нагрузки сети пакеты/мс ("линия")

Рисунок 26. Аппроксимация зависимости времени релаксации системы от количества узлов в системе ("линия")

Рисунок 27. Аппроксимация зависимости времени передачи пакета от количества узлов в системе ("линия")

Рисунок 28. Аппроксимация зависимости нагрузки сети от количества узлов в сети ("линия")

Рисунок 29. Аппроксимация зависимости нагрузки сети от количества узлов в сети ("линия")

## Полносвязная топология

### Количество узлов – 5

Рисунок 30. Время релаксации системы и время передачи (полносвязная-5)

Рисунок 31. Средняя нагрузка сети байт/мс (полносвязная - 5)

Рисунок 32. Средняя нагрузка сети пакеты/мс (полносвязная - 5)

Рисунок 33. Гистограмма времени релаксации системы (полносвязная - 5)

Рисунок 34. Гистограмма времени передачи (полносвязная - 5)

### Количество узлов – 10

Рисунок 35. Время релаксации системы и время передачи (полносвязная-10)

Рисунок 36. Средняя нагрузка сети байт/мс (полносвязная - 10)

Рисунок 37. Средняя нагрузка сети пакеты/мс (полносвязная - 10)

Рисунок 38. Гистограмма времени релаксации системы (полносвязная - 10)

Рисунок 39. Гистограмма времени передачи (полносвязная - 10)

### Количество узлов – 15

Рисунок 40. Время релаксации системы и время передачи (полносвязная-15)

Рисунок 41. Средняя нагрузка сети байт/мс (полносвязная - 15)

Рисунок 42. Средняя нагрузка сети пакеты/мс (полносвязная - 15)

Рисунок 43. Гистограмма времени релаксации системы (полносвязная - 15)

Рисунок 44. Гистограмма времени передачи (полносвязная - 15)

### Количество узлов – 20

Рисунок 45. Время релаксации системы и время передачи (полносвязная-20)

Рисунок 46. Средняя нагрузка сети байт/мс (полносвязная - 20)

Рисунок 47. Средняя нагрузка сети пакеты/мс (полносвязная - 20)

Рисунок 48. Гистограмма времени релаксации системы (полносвязная - 20)

Рисунок 49. Гистограмма времени передачи (полносвязная - 20)

### Количество узлов – 30

Рисунок 50. Время релаксации системы и время передачи (полносвязная-30)

Рисунок 51. Средняя нагрузка сети байт/мс (полносвязная - 30)

Рисунок 52. Средняя нагрузка сети пакеты/мс (полносвязная - 30)

Рисунок 53. Гистограмма времени релаксации системы (полносвязная - 30)

Рисунок 54. Гистограмма времени передачи (полносвязная - 30)

### Сводные графики

Рисунок 55. Сводный график времени релаксации системы (полносвязная)

Рисунок 56. Сводный график времени передачи пакета (полносвязная)

Рисунок 57. Сводный график средней нагрузки системы байт/мс

Рисунок 58. Сводный график средней нагрузки системы пакеты/мс

Рисунок 59. График зависимости среднего времени релаксации от количества узлов в сети (полносвязная)

Рисунок 60. График среднего времени передачи пакета (полносвязная)

Рисунок 61. Аппроксимация зависимости нагрузки сети от количества узлов в сети (полнносвязная)

Рисунок 62. Аппроксимация зависимости нагрузки сети от количества узлов в сети (полносвязная)

В обоих экспериментах система продемонстрировала ожидаемое поведение.

В топологии “линия” c ростом числа узлов в сети наблюдается близкое к линейному увеличение времени релаксации системы, времени передачи пакета, увеличение нагрузки сети.

В полносвязной топологии с ростом числа узлов в сети наблюдается увеличение времени релаксации в диапазоне от 5 узлов до 10. При количестве узлов более 10, время релаксации незначительно колеблется около некоторого среднего значения. Это вызвано переполнением буферов устройств. Время передачи в полносвязной топологии не зависит от количества узлов. Зависимость нагрузки сети от количества узлов близка к линейной.

# Заключение

В результате проделанной работы, было разработано приложение-симулятор mesh-сети на устройствах Bluetooth Low Energy, использующее стек протоколов реальной системы, позволяющее избежать расходов на натурные испытания системы, получить необходимые характеристики сети, изучить особенности поведения системы до её реализации.

В качестве перспектив развития проекта можно отметить следующие направления: расширение набора характеристик (например, оценка размножения пакетов, среднее количество пакетов в буфере устройств и т.п.), увеличение скорости работы симулятора, улучшение формы представления данных, универсализация процедуры импорта стека протоколов, расширение режимов работы Bluetooth Low Energy устройства в симуляторе.

В ходе работы над данным проектом, были получены знания в области технологии Bluetooth, расширены знания языка программирования c++. Получен опыт работы в большой компании.

# Список литературы

* Specification of the Bluetooth system v.4.0.
* Moble Network Layers
* Yan Zhang, Wireless Mesh Networking / Yan Zhang, Jijun Luo, Honglin Hu. – Auerbah Publications. – 610 стр.
* http://www.bluetooth.com