Операционная система Contiki для моделирования сетей с потерями и низким потреблением энергии LLNs

Содержание

1. Сети с потерями и низким потреблением энергии LLNs

Беспроводную сенсорную сеть также называют сетью с низким потреблением энергии (LLNs–Low power and Lossy Networks), она является классом сетей, состоящих из устройств с коммуникационной инфраструктурой, предназначенной для мониторинга физических или экологических условий в различных регионах.

Обычные контролируемые параметры - это температура, влажность, давление, напряжение и мощность в линии, жизненно важные функции организма и т.д.. Размер этих устройств, как правило, очень маленький, питание осуществляется от батареи, поэтому они нуждаются в очень разумном использовании ресурсов. LLNs предназначены для приложений, создающих небольшой трафик. Такой трафик является очень распространенным для приложений умного дома, всепроникающих вычислений.

Беспроводные сенсорные сети поддерживает 3 вида трафика: точка-точа (между устройствами внутри LLN), точка- многоточка (между центральным устройством до других устройств внутри LLN), многоточка - точка (между устройствами внутри LLN до центрального устройства). Поскольку эти устройства имеют ограниченный диапазон передачи, маршрутизация очень важна в этих устройствах для успешной передачи информации.

Маршрутизация отвечает за управление выбором маршрутов среди сенсорных узлов и определение наиболее эффективного маршрута для передачи пакетов.

Маршрутизация в беспроводных сенсорных сетях очень важна, поскольку большинство приложений, разработанных для запуска таких типов сетей, основано на процессе коммуникации внутри узла, а также процессе передачи данных от отдельных узлов до центральной точки сбора.

Маршрутизация в беспроводных сенсорных сетях отличается от маршрутизации в сетях IP тем, что схема обращения часто не важна для данных, которые управляют приложением. Для большинства приложений данные собираются от множественных узлов и отправляются в центральный узел. Так же у узлов БСС есть серьезные ограничения ресурса, такие как вычислительная мощность, объем памяти, энергия.

Традиционная реализация TCP/IP требует слишком много ресурсов таких, как размер кода и используемая память, которая не является полезной для умных объектов с ограниченными ресурсами. Поэтому, стек TCP/IP БСС предназначен для того, чтобы иметь только абсолютно минимальный набор особенностей от полного стека TCP/IP.

Поскольку топология беспроводных сенсорных сетей может быть довольно динамичной, то логично можно рассмотреть возможность использования протоколов маршрутизации MANET. В этом разделе рассмотрим протоколы AODV, DSDV, DSR. Другие протоколы маршрутизации

такие, как (RIP, EIGRP, OSPF, IS-IS) требуют очень большой вычислительной мощности с точки зрения БСС и далее не рассматриваются.

Работа над конкретными технологиями и протоколами для LLN сетей была начата в конце 1990-х годов специалистами рабочей группы инженеров Интернета. Именно тогда была основана группа MANET, создавшая два семейства протоколов: реактивные протоколы (в частности, AODV,на смену которому впоследствии пришел DYMO) и проактивные, в числе которых нужно выделить OSLR (сегодня ему на смену пришел OSLRv2). В 2008-2009 годах появилась возможность использовать протокол IPv6 в сетях LLN благодаря разработке спецификации 6LoWPAN.

IPv6 является более подходящим протоколом для потребностей БСС, чем IPv4. Для взаимодействия по протоколу IPv6 поверх маломощных беспроводных персональных сетей стандарта IEEE 802.15.4 в рабочей группе IETF был разработан стандарт 6LoWPAN(IPv6 over Low power Wireless Personal Area Networks) для обеспечения взаимодействия беспроводных персональных сетей IEEE 802.15 с широко распространёнными сетями IP.

После этого некоторые рабочие группы начали переходить на IPv6 для сетей LLNs, например на рисунке 1 в IETF есть две рабочих групп, первая IETFWGROLL, в которой был разработан

протокол сетевого уровня Routing Protocol for LLNs (RPL)[62], и вторая рабочая группа CoRE, в которой был разработан протокол прикладного уровня Constrained Application Protocol (CoAP).

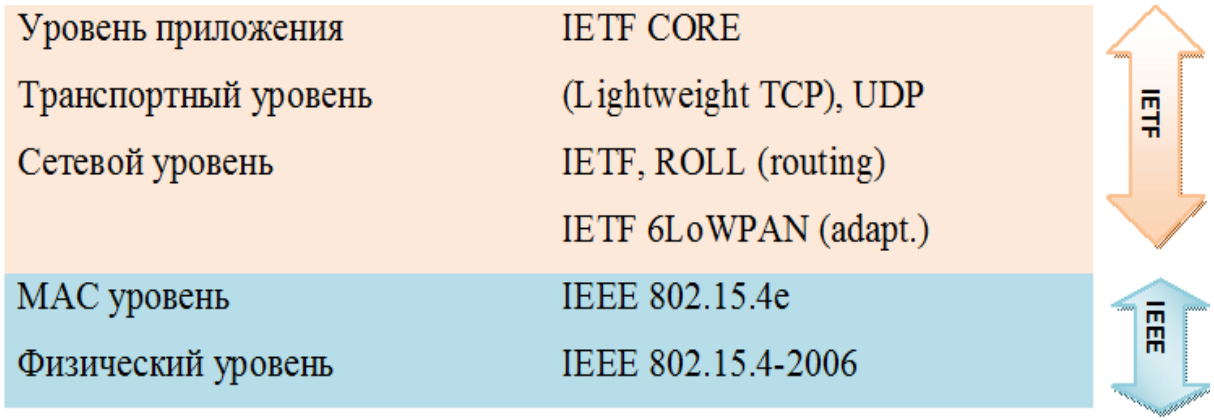


Рисунок 1 – Уровни взаимодействия для LLNs

Сетевой уровень обеспечивает принятие решения о маршрутизации и перенаправляет пакеты от узла к узлу. Стек протоколов этого уровня является протоколом RPL, который используется для маршрутизации и переадресации пакетов. Протокол RPL использует для управления пакетов процедуры DIO, DIS, DAO для построения дерева топологии и сохраненных маршрутов. Пакеты управления также несут информацию о различных параметрах сети.

2. Протоколы физического уровня

Стремительное развитие беспроводных технологий, связанное с распространением «блокнотных» компьютеров и инструментов поискового вызова, популяризацией систем вида «персональный секретарь» (Personal Digital Assistant (PDA)), а также расширением функциональных возможностей мобильных телефонов, позволяет отнести их к наиболее быстро прогрессирующей отрасли телекоммуникаций, о чем свидетельствуют, например, даты разработки и внедрения принципиально новых стандартов и технологий.

Успехи современной радиоэлектроники, галопирующее развитие микропроцессорной техники и новые алгоритмы цифровой обработки сигналов вместе с использованием перспективных телекоммуникационных технологий открывают новые возможности по созданию беспроводных систем связи с выполнением жестких требований, касающихся высокой пропускной способности и помехозащищенности. Такие системы призваны обеспечивать расчет времени, деловое планирование, поддержку постоянной связи с удаленными станциями, хранение документов. Девизом беспроводных технологий стало выражение «any time and any where», то есть предоставление услуг связи независимо от места и времени.

На сегодняшний день наиболее известными технологиями беспроводной связи являются

Bluetooth, ZigBee и конечно же Wi-Fi. При этом каждая из этих технологий является эффективной в различных сферах и ситуациях, рассмотрим их более подробно, выделим преимущества и недостатки, а также проанализируем в каких сферах деятельности они наиболее эффективны.

2.1. Технология беспроводной передачи данных Wi-Fi

Технология Wi-Fi, или RadioEthernet IEEE 802.11 - является первым промышленным стандартом, который позволил организовать беспроводные локальные сети (Wireless Local Area Networks - WLAN) на ограниченной территории, т.е. когда несколько пользователей имеют равный доступ к общему каналу передачи данных . Стандарт был создан в Инженерном институте электротехники и радиоэлектроники (Institute Electrical and Electronics Engineers - IEEE) и может сравниться со стандартом 802.3 для обыкновенных проводных Ethernet сетей. Центром беспроводной сети WI-FI является точка доступа (Access Point), которая может подключаться к любой наземной сетевой инфраструктуре, например, офисной Ethernet-сети и обеспечивать передачу радиосигнала .

На сегодняшний день разработано уже множество версий стандарта - IEEE 802.11 с соответствующими буквенными индексами a, b, c, d, e, g, h, i, j, k, i, m, n, o, p, q, r, s, u, v, w.

Однако только пять из них (a, b, g и n) являются наиболее распространенными и популярными у производителей оборудования, другие же представляют собой усовершенствования, дополнения или исправления уже принятых спецификаций. Первая спецификация стандарта IEEE 802.11 была принята в 1997 году. Она установила передачу данных на скорости 1 и 2 Мбит/с в нелицензионном диапазоне частот 2,4 ГГц, а также механизм управления доступом к физической среде (радиоканалу), который использует метод множественного доступа с распознаванием несущей и устранением коллизий (Carrier Sense Multiple Access with Collision Avoidance, CSMA-CA). В табл.1.1 представлены ключевые технические характеристики основных стандартов IEEE 802.11.

Таблица 1.1 – Ключевые характеристики основных стандартов IEEE 802.11

Стандарт	**IEEE 802.11a**	**IEEE 802.11b**	**IEEE 802.11g**	**IEEE 802.11n**
Год ратификации Wi-Fi альянсом	1999	1999	2003	2009
Частотный диапазон, ГГц	5.15-5.25 5.67-5.85	2.4-2.483	2.4-2.483	2.4-2.483 5.15-5.25

				5.67-5.85
Доступ к радиоканалу	CSMA-CA	CSMA-CA	CSMA-CA	CSMA-CA
Количество абонентов на один канал	64	64	64	64
Максимальная скорость обмена данными	54 Мбит/с	11 Мбит/с	54 Мбит/с	600 Мбит/с
Обычная скорость передачи данных	23 Мбит/с	4 Мбит/с	20 Мбит/с	120 Мбит/с
Ширина канала	20 МГц	22 МГц	20 МГц	40 МГц
Метод модуляции	OFDM	BPSK,CCK	OFDM	BPSK, QPSK, 16-QAM, 64-QAM
Дальность действия в помещении	10-20	20-100	20-50	10-20

Одновременно следует отметить, что Международная организация Wi-Fi Alliance презентовала новый стандарт беспроводной связи Wi-Fi 802.11ah «HaLow». Диапазон работы

нового стандарта 900 МГц, именно на этой частоте устройства Wi-Fi Certified могут подключаться на большем расстоянии с наименьшими затратами энергии. По радиусу действия Wi-Fi «HaLow» приблизительно в два раза превосходит используемые в настоящее время варианты Wi-Fi. Кроме того, диапазон 900 МГц предоставляет возможности, которые необходимы для таких приложений, как мобильные электронные устройства и датчики, которые пользователи носят с собой .

В отличие от уже принятых стандартов, которые работают на частоте в 2,4 ГГц и 5 ГГц, Wi-Fi 802.11ah удваивает радиус действия сигнала, обеспечивает надежное соединение, в случае, если радиоволны проникают через препятствия, например, стены (см. рис.1.1). Отдельно следует обратить внимание на тот факт, что, в связи с альтернативной частотой, новый стандарт не подвержен помехам от микроволновых печей и прочих бытовых приборов.

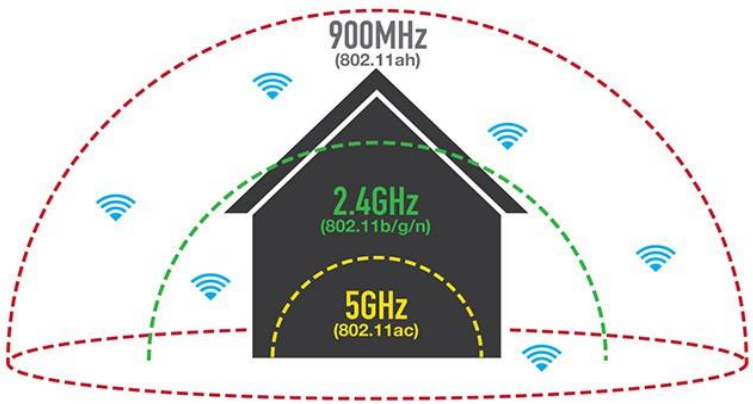


Рисунок 2 – Диапазон действия нового стандарта беспроводной связи Wi-Fi 802.11ah «HaLow» .

Разработчики уверяют, что Wi-Fi 802.11ah будет обладать всеми достоинствами Wi-Fi, включая простоту установки, широкую совместимость оборудования, а также надежную защиту данных. Кроме того, планируется, что устройства, с поддержкой Wi-Fi «HaLow» также будут работать в диапазонах 2,4 и 5 ГГц, это позволит им интегрироваться в действующую экосистему Wi-Fi, в которую входит более 7 млрд. устройств. Аналогично устройствам Wi-Fi устройства Wi-Fi «HaLow» будут поддерживать подключение по IP, благодаря чему они смогут работать с облачными сервисами, что имеет большое значение и важность для интернета вещей. Еще одно несомненное достоинство Wi-Fi «HaLow» заключается в возможности подключения к одной точке доступа тысяч устройств.

Помимо нового стандарта Wi-Fi 802.11ah ключевые перспективы развития Wi-Fi в ближайшее время заключаются в следующем:

1. Освоение диапазона 60 ГГц. Wireless Gigabit Alliance (WiGig Alliance) развивает технологию Wi-Fi в диапазоне 60 ГГц с максимальной скоростью передачи 7 Гбит/с для сценариев пикосотового покрытия.
2. Развитие технологии Wi-Fi Direct, которая позволяет обеспечить с обычной скоростью Wi-Fi прямые соединения между различными клиентскими устройствами, обходя традиционные точки доступа и беспроводные маршрутизаторы.
3. Поддержка усовершенствованных решений VoIP с обновленным набором WFA-протоколов для развития конкурентоспособных альтернативных услуг.
4. Развитие сотовых (mesh) Wi-Fi сетей, основанных на дешевых модулях, каждый из которых по радиоканалу соединен со всеми соседями в зоне радиовидимости.
5. Дальнейшее усовершенствование радио интерфейса Wi-Fi.
6. Совершенствование клиентского опыта Wi-Fi с помощью оптимизации взаимодействия с точками доступа .

И в завершении анализа технологии беспроводной передачи данных Wi-Fi отметим, что ее основные преимущества заключаются в:

- простоте использования готовых модулей;
- легкости интеграции с существующими проводными сетями (LAN);

– высокой скорости передачи информации;
– безопасности передачи информации (64/128-битное шифрование).

В то же время основные недостатки технологии Wi-Fi связаны с:

– более высокой (по сравнению с другими беспроводными сетями) ценой на оборудование;
– большим (по сравнению с другими беспроводными сетями) энергопотреблением;
– работой множества устройств в диапазоне частот 2.4 ГГц, например, имеющих Bluetooth адаптер, это ухудшает качество передачи информации;
– существующими в каждой стране эксплуатационными ограничениями и частотными диапазонами;
– недостаточно защищенным стандартом шифрования WEP, который используется для защиты доступа к сети.

Рассматривая более подробно сферы применения технологии беспроводной передачи данных Wi-Fi, можно отметить, что сегодня она используется во многих областях деятельности. Наибольший спрос технология получила у Интернет провайдеров, поскольку это позволяет им отказаться от десятков километров проводов. Также данная технология используется в игровой индустрии. Такие известные бренды как Sony и Nintendo монтируют Wi-Fi устройства в свои

игровые консоли, для обеспечения доступа к Интернету. Некоторые коммерческие организации предоставляют доступ к сети Интернет с использованием данной технологии. Если же рассматривать крупные компании и корпорации, то они используют Wi-Fi для создания корпоративной сети, так как это дешевле чем создавать проводную сеть Ethernet.

2.2. Технология беспроводной передачи данных Bluetooth

Технология Bluetooth (стандарт IEEE 802.15) является первой технологией, которая позволила организовать беспроводную персональную сеть передачи данных (WPAN - Wireless Personal Network). Она дает возможность проводить передачу голоса и данных с использованием радиоканала на короткие расстояния (10-100 м) в нелицензионном диапазоне частот 2,4 ГГц, а также соединять мобильные телефоны, ПК и прочие устройства в условиях отсутствия прямой видимости.

Стандарт Bluetooth имеет более 10 профилей, то есть наборов функции устройств Bluetooth. Основными профилями, утвержденными группой разработчиков SIG являются:

- Advanced Audio Distribution Profile (A2DP) этот профиль предназначен для обеспечения передачи музыки в беспроводные наушники;
- Audio / Video Remote Control Profile (AVRCP) профиль, который позволяет управлять

функциями телевизора;

- File Transfer Profile (FTP_profile) профиль, обеспечивающий обмен данными между устройствами;
- Hands-Free Profile (HFP) профиль предназначен для соединения беспроводных наушников и мобильных устройств, оснащенный также функцией разговора по телефону;
- LAN Access Profile (LAP) профиль, который обеспечивает доступ к сетям LAN, WAN или Internet с использованием средств другого Bluetooth устройства;
- SIM Access Profile (SAP, SIM) профиль позволяющий получить доступ к SIM-карте мобильного устройства и использовать одну SIM-карту на нескольких устройствах;
- Wireless Application Protocol Bearer (WAPB) профиль который уравнивает протокол для организации (Point-to-Point) соединения через Bluetooth, а также другие профили.

Как известно, Wi-Fi и Bluetooth используют диапазон 2,4 ГГц. В случае если Bluetooth устройства расположены в зоне покрытия устройств Wi-Fi и проводят обмен информацией между собой, зачастую возникают коллизиии, что может негативным образом отразиться на работоспособности устройств . Технология AFH дает возможность исключить появление коллизий: в процессе обмена информацией для преодоления интерференций технология Bluetooth использует скачкообразное изменение частоты канала, при выборе которого не принимаются во

внимание частотные каналы, обслуживающие обмен данными устройства Wi-Fi. На рис. 3 проиллюстрирован принцип технологии AFH.

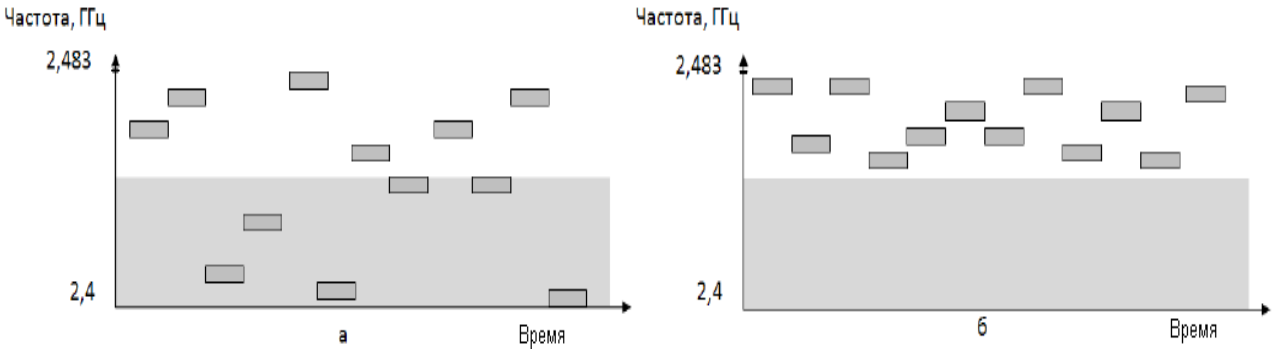


Рисунок 3 – Принцип работы технологии AFH. **а** – коллизии; **б** – переход от коллизий с помощью адаптивной перестройки частоты.

Подытоживая можно сказать, что преимуществами технологии Bluetooth являются: мобильность и малые размеры; простота использования готовых модулей; значительная скорость передачи данных; безопасность передачи данных; доступность; необходимость авторизации устройства; низкий порог чувствительности к помехам (зависит от толщины и материала

препятствия); высокий уровень стандартизации.

Однако следует отметить, что данная технология не лишена и недостатков. Если, к примеру, два пользователя хотят обменяться данными, то в процессе поиска устройств друг друга будут найдены все устройства, которые включены в радиусе 10-15 метров, что снижает скорость инициализации устройств. Кроме того, следует отметить невозможность построения сетей сложной топологии и большие (по сравнению с сетями ZigBee) объемы энергопотребления.

Чаще всего технология Bluetooth применяется для обеспечения радиосвязи между различными видами электронных устройств путем замены ведущего последовательного соединения между двумя устройствами на беспроводное.

2.3. Технология беспроводной передачи данных ZigBee

Технология беспроводной передачи данных ZigBee появилась на рынке уже после Bluetooth и Wi-Fi. Необходимость разработки технологии ZigBee связана, прежде всего, с тем, что для определенных операций, например, таких как удаленное управление освещением или воротами, считывание информации с датчиков ключевыми критериями при выборе эффективной технологии беспроводной передачи являются низкая стоимость и низкое энергопотребление аппаратной части

.

Сети ZigBee называют сетями, которые самоорганизуются и самовосстанавливаются. Это связано с тем, что ZigBee-устройства, благодаря встроенному программному обеспечению, при включении питания могут самостоятельно находить друг друга и создавать сеть, а в случае поломки какого-либо узла наделены возможностями устанавливать новые маршруты для передачи сообщений. Таким образом, технологию ZigBee можно использовать как для обеспечения простых соединений «точка-точка» и «звезда», так и для обслуживания сложных сетей.

ZigBee основан на стандарте IEEE 802.15.4-2006 для беспроводных персональных сетей. IEEE 802.15.4 - стандарт, который определяет физический слой и управление доступом к среде для беспроводных персональных сетей с небольшой скоростью. Архитектура стандарта показана на рисунке 4.

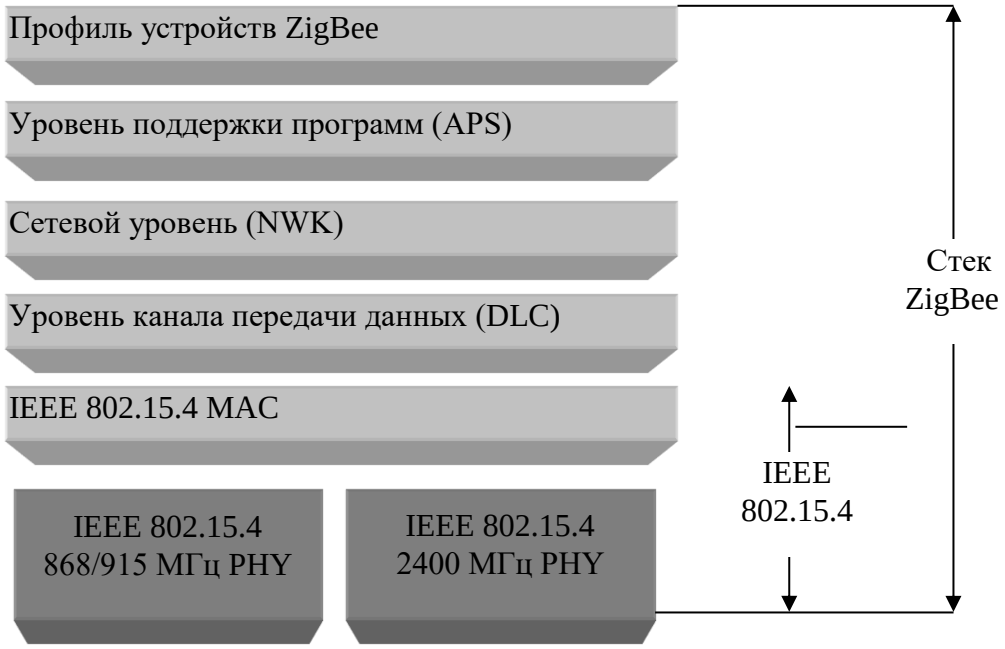


Рисунок 4– Связь стека ZigBee со стандартом IEEE 802.15.4

Характеристики ZigBee:

– частотный диапазон - 2,4 ГГц, 16 частот с шириной 5 МГц;

- DS-SS - прямое расширение спектра сигнала;
- O-QPSK - квадратурная фазовая манипуляция со смещением;
- автоматическое регулирование выходящей мощности в широких пределах для обеспечения энергоэффективности;
- разрешенная мощность - 100 мВт;
- оценка уровня мощности сигнала в эфире - RSSI и подтверждение об успешной доставке для каждого пакета данных;
- Mesh - сетевая технология, которая обеспечивает самовосстановление и самоорганизацию, надежность и гибкость маршрутизации;
- до 65536 узлов (модемов) в сети;
- механизм множественного доступа в эфир с контролем несущей и предотвращением коллизий - CSMA (Carrier Sense, Multiple Access);
- 128-битное шифрование данных по алгоритму AES;
- скорость передачи данных, включая служебную информацию - до 250 кбит/с. [55].

Как в случае с Wi-Fi в ZigBee используется CSMA-CA для существенного снижения количества случаев столкновений, вызванных передачей данных несколькими устройствами одновременно.

Кроме того, отдельный акцент следует сделать на том, что сети ZigBee являются простыми в установке, т.к. они формируются автономно. Также, благодаря, сочетанию маршрутизации на основе таблицы и маршрутизации по дереву обеспечивается гибкость работы и возможность выбора разработчиками широкого спектра соотношений цена/производительность, что в свою очередь способствует формированию масштабируемой и недорогой сетевой инфраструктуры. По сравнению с технологией Wi-Fi ZigBee имеет лучшее соотношение «дальность передачи/скорость передачи/ энергопотребления» благодаря тому, что первичная обработка линейно-частотного импульса выполняется аналоговым способом.

Недостатком ZigBee является недостаточно высокий уровень стандартизации и отсутствие единой программно-аппаратной платформы для разработки сложных программ; невысокая скорость передачи данных, в результате чего большая часть трафика ZigBee расходуется на передачу пакетов; ограниченная совместимость устройств ZigBee различных производителей; отсутствие высокоуровневых профилей.

Технология ZigBee, благодаря своим достоинствам нашла широкое применение на практике. Так, она успешно интегрируется в системы автоматизации жизнеобеспечения строений и зданий (удаленное управление реостатами, выключателями, сетевыми розетками и проч.);

системы автоматического снятия показаний с различных счетчиков (воды, газа, электричества и проч.); системы управления бытовой техникой и электроникой; системы безопасности (датчики доступа и охраны, движения, задымления, утечки газа, воды и т.д.); системы контроля и мониторинга за окружающей средой (датчики давления, температуры, влажности, вибрации и проч.); системы промышленной автоматизации.

Сравнительные характеристики технологий Bluetooth, Wi-Fi и ZigBee приведены в табл. 1.2. Эта информация поможет принять правильное решение при выборе технологии беспроводной передачи данных.

Таблица 1.2 – Сравнительные характеристики технологий Wi-Fi, Bluetooth и ZigBee.

Стандарты	**Технологии**						
	ZigBee 802.15.4			**Bluetooth**	**Wi-Fi**		
					802.11b	**802.11g**	**802.11n**
Частота ГГц	0,868	0,915	2,4	2,4	2,4	2,4	2,4 (5)
Скорость	20 кб/с	40 кб/с	250 кб/с	1 Мб/с	11 Мб/с	54 Мб/с	600 (300)

							Мб/с
Исходящая мощность	0 дБм			0-20 дБм	20 дБм	20 дБм	20 дБм
Радиус действия, м.	10-100			10-100	100	100	150-300
Размер стека кбайт	4-32			>250	>1000	>1000	>1000
Размер сети	216, 264			7+1	64	64	64
Время работы от батареи, ч.	2400-24000			24-240	12-120		
Максимальное количество элементов сети	65536			7	100		
Безопасность	+			Аутентификация, кодирование	68/124 битное шифрование		
Энергопотребление	Невысокое			Невысокое	Высокое		
Цена	Невысокая			Невысокая	Высокая		
Реализуемые стандарты	IEEE 802.14.5			IEEE 802.15.1 IEEE 802.11	IEEE 802.11 a,b,n,g,ah		

Сфера применения	Сенсорные системы	Мобильные устройства	Локальные сети

Таким образом, проведенный анализ позволяет сделать следующие выводы. На сегодняшний день наиболее распространенными беспроводными технологиями является Bluetooth, Wi-Fi и ZigBee. Они пользуются такой популярность за счет их преимуществ, а именно: отсутствие проводов, высокая и безопасная скорость передачи данных, возможность передачи достаточно больших объемов информации и доступность для пользователей.

Технология Bluetooth предпочтительно используется в мобильных устройствах, так как почти каждое из них имеет Bluetooth-адаптер. Технология ZigBee является оптимальной в масштабах предприятий и офисных зданий. Однако анализ развития современного рынка мобильных устройств и сетевых технологии свидетельствует, что в перспективе доминирующей технологией будет Wi-Fi. Об этом свидетельствует тот факт, что большинство современных устройств имеют Wi-Fi адаптер. К тому же с точки зрения создания пользовательских приложений Wi-Fi является более привлекательной технологией для программной реализации беспроводной передачи данных.

В настоящее время одной из острых проблем развития технологии Wi-Fi является увеличение скорости передачи данных. Представляется, что одним из перспективных путей решения данной проблемы является применение методов модуляции цифровых сигналов OFDM (N-OFDM) (ортогональная и не ортогональная частотные дискретные модуляции), технологии пространственно-временной и пространственно поляризационной обработки сигналов, а также MIMO (множественный вход – множественный выход) на базе цифровых антенных решеток. Развитие элементной базы делает возможным использование этих технологий, как в маломощных переносных устройствах, так и в станциях радио, радиорелейной и тропосферной связи. Кроме того, современные достижения процессорной техники позволяют создавать унифицированные комплексы средств связи с возможностью программной реконфигурации оборудования. Уже сейчас разрабатываются новые варианты конфигурации антенн, до 64x64 MIMO . Это позволит добиться еще больших скоростей передачи данных, емкости сети и спектральной эффективности.

Поэтому, с учетом того, что потенциал указанных технологий не исчерпан, исследования в данном направлении являются перспективными и заслуживают дополнительного внимания, как с практической, так и с теоретической точки зрения.

3. Протокол уровня адаптации 6LoWPAN

Проект стандарта IETF 6LoWPAN, рекомендующий передачу сообщений IPv6 поверх 802.15.4, выпущенный в марте 2007 года, изменил все. Потенциал 6LoWPAN для работы с устройствами с низким потреблением энергии делает его привлекательным для использования не только в портативной технике, но и в широком диапазоне промышленных средств. Встроенная поддержка шифрования AES-128 закладывает основу для надежной аутентификации и безопасности. Для того, чтобы быть конкурентоспособным по отношению к другим протоколам, в 6LoWPAN применяется «плати только за то, что используешь» схема сжатия заголовков. При прямой интеграции с IP маршрутизаторами появляется превосходство в использовании наиболее продвинутых сетевых систем безопасности в отличие от тех, что предлагаются шлюзами в adhoc сети.

Рабочая группа IETF 6LoWPAN была сформирована для решения проблемы передачи IP пакетов поверх каналов IEEE 802.15.4 способом, удовлетворяющим открытым стандартам и предоставляющим взаимодействие с другими IP каналами и устройствами в той же мере, как и с устройствами IEEE802.15.4.

Такое решение имеет множество преимуществ. Каждый сенсор в 6loWPAN сети имеет персональный IPv6 адрес. Это позволяет многим компаниям производить LR-WPAN устройства,

которые могут работать вместе в одной сети, позволяет взаимодействовать данным устройствам, работать с сетевыми компьютерами и оборудованием, которое уже существует. Каждый узел сенсорной сети становится доступен из внешних сетей по IP адресу. Это избавляет от необходимости иметь комплексные шлюзы для каждого локального IEEE 802.15.4 протокола, множества адаптеров, используемых существующими приложениями для связи через эти шлюзы, упрощает специфичные для шлюзов процедуры аутентификации и безопасности. Множество устоявшихся, основанных на IP протоколе программных инструментов, таких как ping, traceroute, SNMP может быть сразу же использовано для объединения в сеть и обслуживания LR-WPAN устройств. Также на базе IP могут быть легко реализованы функции NAT (подмена адресов), распределение нагрузки, кэширование. Существующие модели передачи данных на программном уровне и сервисы на базе HTTP/XML/SOAP позволяют упростить процесс разработки приложений для LR-WPAN сетей и унифицировать интеграцию устройств в существующую корпоративную сеть при использовании 6LoWPAN. В новом протоколе маршрутизации RPL поддерживается множественный граф и есть возможность передачи пакетов по соединениям с различными параметрами. RPL (Routing Protocol for Low energy and Lossy networks) – это новый протокол для сетей с низким потреблением энергии и потерями, в основе которого лежат направленные ациклические графы DODAG (Destination Oriented Directed Acyclic Graph). Этот

протокол обеспечивает пути от маршрутизатора к приемному узлу, при этом в маршрутизаторах требуется хранение небольшого объема служебных данных и таблиц маршрутизации, содержащих информацию о родительских узлах в DODAG.

4. Протоколы сетевого уровня

4.1. Протокол динамической маршрутизации DSDV

DSDV (Destination Sequenced Distance Vector) является высокоэффективным динамическим протоколом маршрутизации для ad-hoc сетей. Он классифицируется как протокол дистанционно-векторного класса и основывается на многшаговой маршрутизации и алгоритме Беллмана-Форда.

По определению сеть ad-hoc состоит из мобильных узлов, которые обмениваются данными внутри сети без вмешательства любой централизованной точки доступа. Чтобы сформировать такую сеть, каждому узлу надо функционировать в качестве специализированного маршрутизатора. Когда разрабатывался протокол DSDV, было принято решение учитывать, что топология в ad-hoc сетях очень динамичная и что большинство пользователей не захотят выполнять каких-либо административных действий для создания сети.

Основная цель протокола DSDV – обеспечить решение проблемы петель маршрутизации. Для различия между действительными и недействительными соединениями, протокол использует

(действующее соединение) и нечетные (недействующее соединение) числа. Эти последовательные номера генерируются по назначению.

DSDV избегает проблемы петлеобразования, помечая каждый маршрут с порядковым номером. Основываясь на порядковом номере, узлы в состоянии дифференцировать старые и новые маршруты.

К недостаткам протокола относятся:

−DSDV требует регулярное обновлять таблицы маршрутизации, что требует расхода энергии и небольшого объема пропускной способности, даже когда сеть не используется.

−Когда топология сети изменяется, новый порядковый номер необходимо ввести до повторного входа в сеть.

Исходя из сказанного, DSDV хорошо использовать для не очень динамичных сетей.

4.2. Протокол динамической маршрутизации от источника DSR

Протокол разработан для работы с низкими накладными расходами и возможностью быстро реагировать на изменения в сетевом окружении. Сети с маршрутизацией DSR могут масштабироваться до сотен узлов и поддерживают высокую степень мобильности узлов. Однако, задержки в установлении соединения выше, чем в протоколах, использующих таблицы. DSR (Dynamic Source Routing) - простой и эффективный протокол, который обеспечивает

маршрутизацию без петель и может работать в сетях с однонаправленными связями. Это позволяет иметь низкие накладные расходы и обеспечивать быструю реакцию на сетевые изменения. Протокол определяет и поддерживает все маршрутизации внутри своей сети для самоорганизации и самоконфигруции сети.

К недостаткам протокола относятся:

– Устаревшая информация кэша маршрутов может привести к несогласованности на этапе восстановления маршрут.

– Задержка установления соединения выше, чем в табличных протоколах. Хотя такой протокол хорошо работает в статических и малоподвижных средах, производительность быстро ухудшается с увеличением мобильности. Кроме того, значительные накладные расходы маршрутизации присутствуют вследствие механизма источник-маршрутизации, используемого в DSR. Эти накладные расходы маршрутизации прямо пропорциональ на длине пути.

4.3. Протокол динамической маршрутизации AODV

AODV (Ad Hoc On-Demand Distance Vector) - это протокол динамической маршрутизации, основанный на построении таблиц маршрутизации на каждом узле сети для минимизации времени передачи информации между узлами. Принцип работы протокола показан на рис.3. Протокол находит пути маршрутизации независимо от использования маршрутов. Первым шагом

протокола является построение таблиц маршрутизации на каждом узле. В таблице содержится длина кратчайшего пути к каждому узлу в сети через каждый соседний узел. На каждом следующем шаге каждый узел обменивается с соседними узлами информацией о каждом известном ему кратчайшем пути к каждому узлу сети. После некоторого количества шагов, зависящего от количества узлов в сети, таблицы маршрутизации на узлах перестают изменяться, после чего начинается передача данных по кратчайшему найденному пути **Ошибка! Источник ссылки не найден.**. При сбое узла сети из таблицы выбирается следующий кратчайший маршрут сетей, где узлы-пользователи могут как угодно изменять свое местоположение, тем самым постоянно изменяя топологию без создания каких-либо определенных стационарных путей передачи данных. Обмен сообщениями в протоколе AODV показан на рис.2. Для мониторинга и обнаружения соседних узлов всем узлам сети отправляется сообщение HELLO, после чего происходит рассылка сообщений для запроса маршрута RREQ (Route Request). После получения ответных сообщений RREP (Route Response) от соседних узлов данные (DATA) перенаправляется оптимальным маршрутом между источником S и узлом назначения D, в случае возникновения ошибки узел отправляет сообщений об ошибке RERR (RouteError)**Ошибка! Источник ссылки не найден.**.

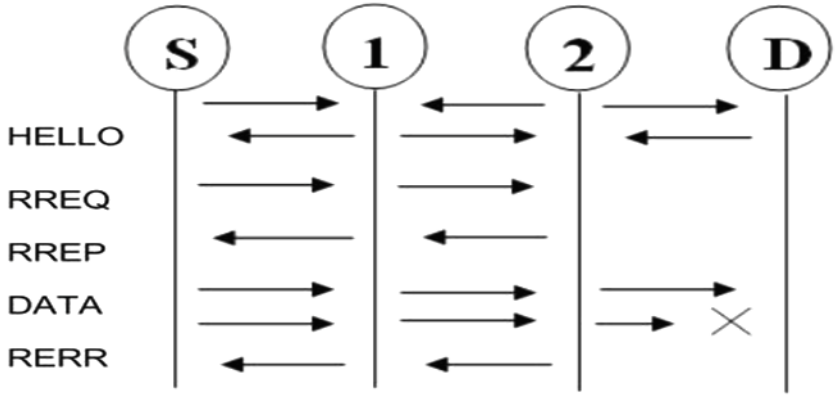


Рисунок 5 – Обмен сообщениями в протоколе AODV

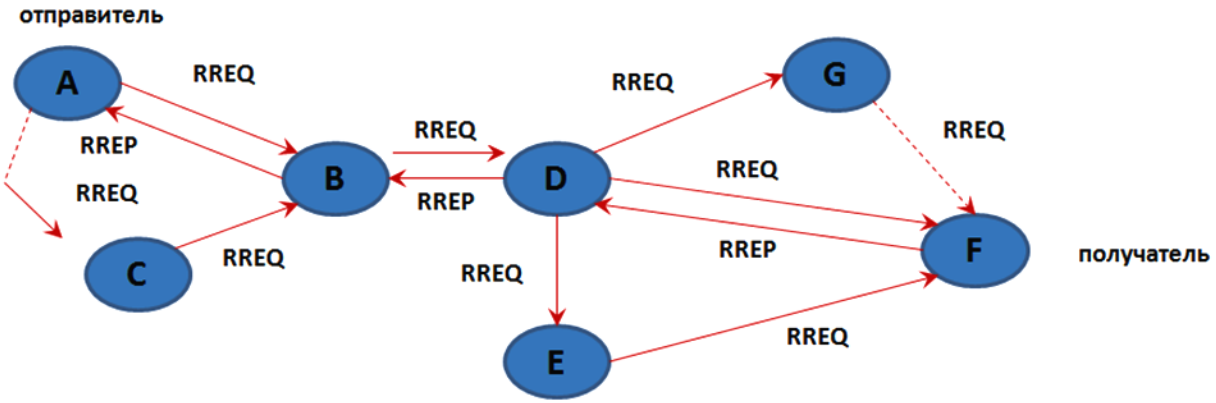


Рисунок 6 – Принцип работы протокола AODV

Преимущество AODV состоит в том, что он не создаёт дополнительного трафика при передаче данных по установленному маршруту.

Один из существенных недостатков протоколов данного типа - процесс разбиения на иерархические структуры или домены, результаты которого сильно сказываются на оптимальности маршрутизации сети. Кроме того, к недостаткам можно отнести наличие постоянного служебного трафика.

4.4. Анализ протокола маршрутизации RPL

Каждое соединение в RPL сети представляется набором показателей, таких как скорость, энергопотребление, поддержка шифрования и т.д., которые являются основой для построения ациклических графов DODAG. Пример графа в RPL представлен на рис. 37, узлы расположены в виде дерева, каждый из них имеет ранг (от 1 до 3 на рисунке), на основе которого они выбирают маршрут передачи сообщений до корневого узла. В конвергентной сенсорной сети каждый RPL маршрутизатор определяет набор родительских узлов, каждый из которых является потенциальной целью для следующего перехода на пути к "корневым узлам" в DODAG, а также предпочтительный родительский узел**Ошибка! Источник ссылки не найден.**.

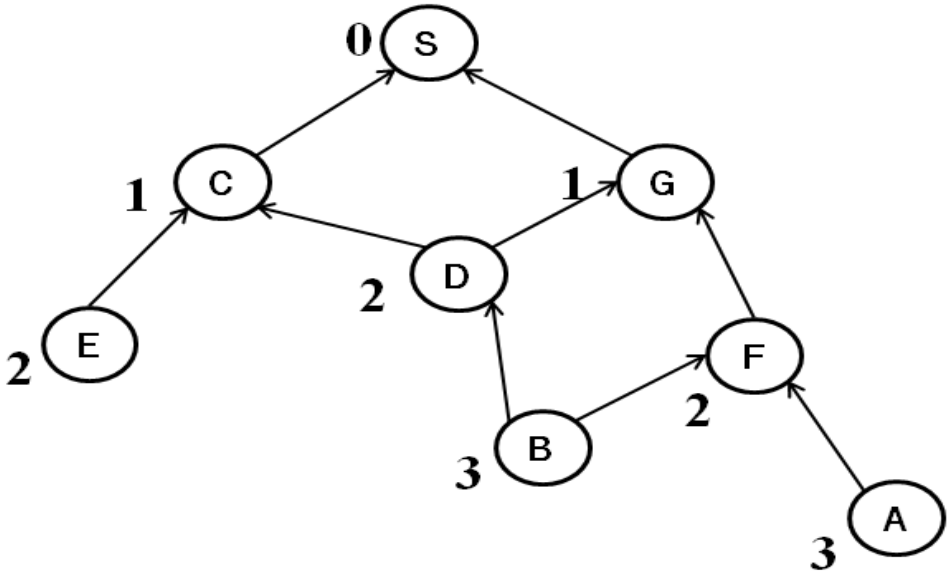


Рисунок 7 – Соединение в RPL

Преимущество RPL состоит в том, что он поддерживает несколько типов сообщений, а именно: точка-точка, точка-многоточка, многоточка-точка, также поддерживается множественный граф. В сети можно создать несколько графов и узел выбирает граф, подходящий для передачи данных в зависимости от типа данных или приложения, для которого они востребованы. Например, в сети на рис. 38 можно организовать граф для передачи сигнальных

сообщений с минимальной задержкой (LBR1- LLNBorderRouter) и граф для телеметрических данных с наиболее энергетически эффективным способом передачи (LBR2).

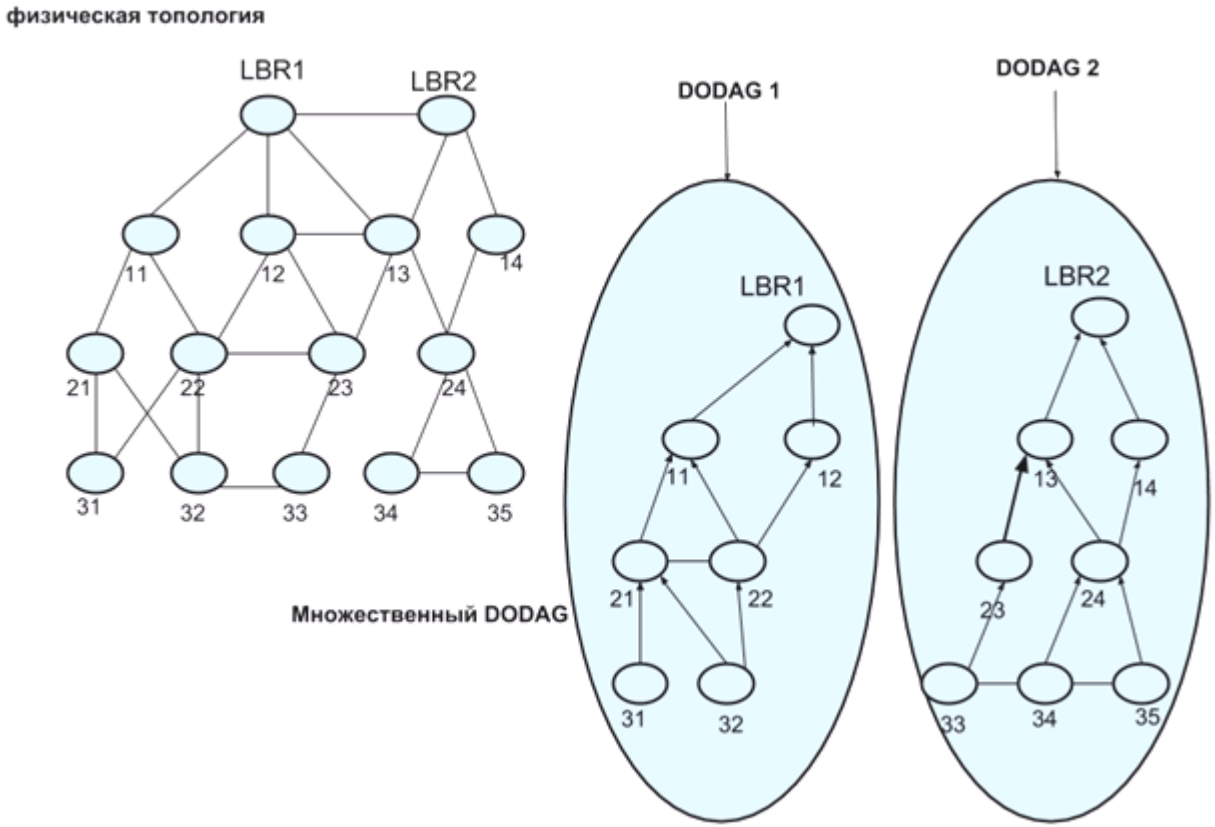


Рисунок 8 – Граф RPL

Сообщения в RPL. RPL использует три типа управляющих сообщений для создания и поддержки топологии RPL и таблицы маршрутизации. Этими сообщениями являются: DODAG Information Object (DIO), DODAG Information Solicitation (DIS) and DODAG Destination Advertisement Object (DAO).

DIO сообщения используются в RPL для формирования, поддержки и обнаружения DODAG. При запуске сети RPL узлы начинают обмениваться информацией о графе DODAG, используя DIO сообщения, которые содержат информацию о конфигурации DODAG. Эти сообщения помогают узлам получить информацию о DODAG и выбрать свои родительские узлы.

DIS Сообщения - любые узлы использует сообщения DIS для того, чтобы запрашивать DIO сообщения от соседних узлов. Последнее требуется для узла в случае, когда он не может получить DIO через заранее определенный интервал времени.

DAO Сообщения - протокол RPL использует DAO сообщения для распространения префикса узла к узлам-предшественникам в поддержку нисходящего трафика. Сообщение DAO также может быть использовано для распространения информации о доступности и для записи маршрутов для областей, включающих в себе несохраненные узлы. На рис. 39 показано использование обратного стека маршрута в DAO сообщениях.

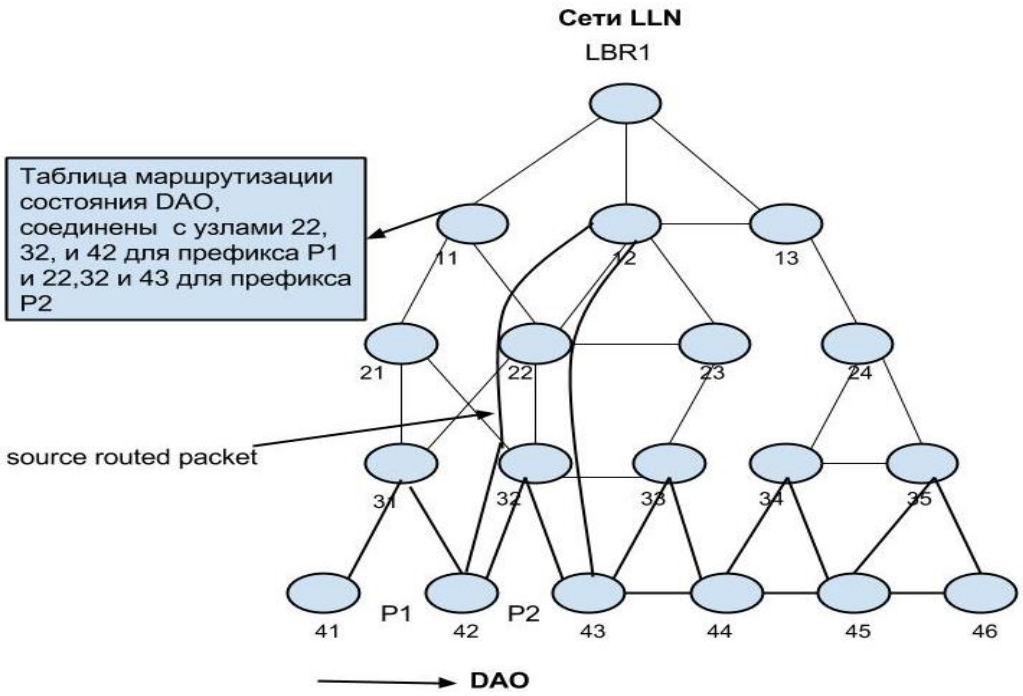


Рисунок 9 – Использование обратного стека маршрута в DAO сообщениях.

Процесс построения графа DODAG для RPL. Каждому узлу ставится в соответствие определенный ранг таким образом, что он возрастает по мере удаления узла от пограничного маршрутизатора. В протоколе RPL это называется направленным ациклическим графом

(Destination Oriented Directed Acyclic Graph – DODAG). Пересылка пакета пограничному роутеру осуществляется с помощью пересылки его соседнему узлу с наименьшим рангом .

Формирование DODAG регулируется несколькими средствами: правилами RPL, используюемыми для защиты от создания петель (на основе рангов DODAG), объектной функцией, метриками пути и настройками узлов. Узел может быть частью нескольких графов, а в экземпляре DODAG может быть несколько корневых DODAG с различным набором узлов.

DIO сообщения отправляются по истечении таймера потока. Основная идея заключается в том, чтобы отправлять собщения DIO при обнаружении несоответствий в DODAG, например, когда узел получает сообщение DIO с новыми параметрами DODAG такими как: объектная функция, новый последовательный номер DODAG или изменение ранга родительского узла и т.д.; обнаружении петель, например когда узел принимает пакет от дочернего узла, который предназначен для пересылки вниз вдоль того же дочернего узла в соответствии с его таблицей маршрутизации, а также, если узел присоединяется к графу с новым идентификатором или перемещается в DODAG. При обнаружении несоответствий узел сбрасывает свой таймер потока, чтобы увеличить частоту объявления сообщений DIO. После стабилизации графа частота отправки DIO сообщений снижается, для того чтобы уменьшить объем сигнального трафика.

Когда узел начинает процесс инициализации в сети, он может приостановить процесс отправки сообщений, пока не услышит объявление DIO от существующего графа. Кроме того, узел может распространить сообщение DIS для опроса соседей и ускорения приема DIO сообщения от них.

Другим вариантом является инициализация своего собственного плавающего DODAG графа и рассылка многоадресных DIO сообщений нового DODAG графа (стоит отметить, что это может быть предпочтительно, если граф необходим для установления и поддержки внутренних соединения между набором узлов при отсутствии целевого DODAG). Однонаправленные сообщения DIO передаются в ответе на однонаправленные сообщения DIS и включают в себя полный набор опций конфигурации DODAG.

После получения сообщения DIO узел должен определить, необходимо ли ему обрабатывать полученное сообщение. Если сообщение DIO имеет ошибки в формате, то узел отбрасывает его без уведомлений. Если нет, то узел должен определить, было ли сообщение DIO отправлено от узла кандидата на подключение. Понятие кандидата в соседи тесно связано с понятием местного доверия, зависит от конкретной реализации и используется для того, чтобы определить, подходит ли узел для выбранного родительского узла. Например, когда узел впервые узнает о соседе, он

может подождать определенный период времени для того, чтобы убедиться, что соединительное звено является достаточно надежным.

Затем узел определяет, связано ли сообщение DIO от DODAG, участником которого он уже является. Если ранг узла источника сообщения DIO меньше суммы ранга узла получателя и некоторого значения, конфигурируемого протоколом RPL называемого DAGMaxRankIncrease (параметр максимального увеличения ранга в графе), то DIO обрабатывается. Это правило называется правилом максимальной глубины.

В случае если сообщение DIO посылается узлом с меньшем рангом, и оно объявляет другой DODAG, обеспечивающий лучший путь согласно целевой функции OF, то сообщение DIO должно обрабатываться. Сообщение DIO также должно быть обработано, если оно исходит от родительского узла, но объявляет другой DODAG, так как родительский узел мог переключиться в другой граф.

Коллизия может произойти, если два узла одновременно передают DIO сообщения друг другу и на их основе принимают решение присоединиться друг к другу. Именно поэтому DIOсообщения, полученные в пределах установленного рискового окна, просто не обрабатываются. Из-за случайного влияния таймера потока ожидается, что последующие DIO сообщения вряд ли будут создавать коллизии.

Безопасность протокола. Безопасность имеет решающее значение для сетей интеллектуальных объектов, но сложность ее обеспечения и размер является основной проблемой LLNs, так как может быть экономически или физически невозможно включить сложные меры безопасности в реализацию RPL. Кроме того, во многих случаях можно использовать механизмы безопасности на канальном уровне, не требуя использования безопасности в RPL. Таким образом, безопасность в RPL доступна в качестве дополнительных расширений. Узлы RPL могут работать в трех режимах безопасности. В первом режиме, называемом "небезопасный" контроль сообщения RPL направляется без каких-либо дополнительных механизмов безопасности. Небезопасный режим означает, что сети RPL могут использовать другие простые средства безопасности (например, канального уровня безопасности) для удовлетворения требований приложений безопасности. Во втором режиме, под названием "предустановка", узлы присоединяются к RPL. Например имеют предустановленные ключи, которые позволяют им обрабатывать и создавать защищенные сообщения RPL. В третьем режиме, под названием "соаутентификация", узлы могут присоединиться в качестве конечных узлов использования предустановленной клавиши на заранее установленный режим или присоединиться в качестве переадресации узла для получения ключа аутентификации. Каждое сообщение имеет безопасный вариант RPL. Уровень безопасности (поддерживаются режимы 32-разрядных и 64-разрядных

MAC и ENC-MAC), а также поддерживаются алгоритмы (СКК и AES-128), используя указанные в протоколе сообщения. Безопасные варианты обеспечивают целостность и воспроизведение защиты, а также конфиденциальность и защиту.

5. Протоколы прикладного уровня

Протокол XMPP. Расширяемый протокол обмена сообщениями и информацией о присутствии Extensible Messagingand Presence Protocol (XMPP). Он был стандартизирован в IETF более десяти лет назад и хорошо зарекомендовал себя, в результате чего до сих пор широко используется в сети Интернет. Однако, будучи достаточно давно разработанным протоколом, XMPP имеет естественные недостатки для реализации новых приложений. По этой причине в прошлом году компания Google прекратила поддержку стандарта XMPP. Тем не менее, в последнее время XMPP вновь получил поддержку инженерного сообщества в качестве протокола для сетей Интернета Вещей. XMPP работает на основе протокола транспортного уровня TCP и обеспечивает асинхронный и синхронный обмен сообщениями. Протокол XMPP предназначен для близких коммуникаций в реальном времени. Таким образом, он поддерживает малый след сообщение и низкую латентность (задержку) при обмене сообщениями. XMPP поддерживает криптографические протоколы TLS/SSL. Тем не менее, не обеспечивают QoS параметры, которые

делают его непрактичным для M2M-коммуникаций. Только унаследованные механизмы обеспечения надежности протокола TC. XMPP поддерживает архитектуру Издатель\подписчик, которая больше подходит для ИВ в отличие от принципа запрос / ответ. Кроме того, это уже установлено, что протокол поддерживается по всему Интернету - это преимущество, в сравнении с относительно новой MQTT. Тем не менее, используется протокол XMPP Сообщений XML, которые создают дополнительные накладные расходы из-за лишних тегов и требуют разбора XML, что требует дополнительных вычислительных способностей, которые увеличивают энергопотребление.

RESTFULSERVICES. Передача состояния представления (REST) на самом деле не протокол, а архитектурный стиль. Он был впервые представлен Роем Филдингом в 2000 году, и широко используется до сих пор. Остальные методы используют HTTPGet, Post, то put и Delete для предоставления ресурсов ориентированной системы обмена сообщениями, где все действия могут выполняться лишь с помощью команд «синхронный запрос/ответ» HTTP-команды. Команда использует встроенный в Accept Заголовок HTTP, для указания формата данных, которые он содержит. Тип Контента может быть XML или json (Нотация объектов JavaScript) и зависит от http-сервера и его конфигурации. Остальное является важной частью ИВ, потому что он

поддерживается всеми коммерческими M2M облачных платформ. Кроме того он может быть легко реализован в смартфонах и планшетных приложениях , потому что он требует только HTTP библиотеки, которые доступны для всех оперативных систем (ОС) распределения. Особенности протокола HTTP могут быть полностью использованы в остальной архитектуре, в том числе обналичивание, проверки подлинности и типа содержимого переговоров. Restful-сервисы используют безопасный и надежный протокол HTTP, который является проверенным во всем мире Интернет-языком. Он может использовать протокол TLS/SSL для безопасности. Однако сегодня большинство коммерческих M2M платформ не поддерживают запросы HTTPS. Вместо этого, они предоставляют уникальные ключи аутентификации, которые должны быть в заголовке каждого запроса, чтобы достичь определенного уровня безопасности. Хотя остальное уже широко используются в коммерческих M2M платформ, маловероятно, что он станет доминирующим протоколом связи, чтобы быть трудновыполнимыми. Он использует протокол http, который означает отсутствие совместимости с ограниченными коммуникационными устройствами. Это оставляет его использование для конечных приложений. Учитывая нынешние тенденции приложений, работающих на смартфонах и планшетах в дополнительных накладных расходах, связанных с функцией запрос/ответ и влияют на использование батареи, а также делает непрерывным опрос или длинным опрос для значений, особенно, когда нет новых обновлений и

накладные расходы становятся бесполезными. Вопросы, которые можно избежать, если издатель / подписка протокол используется, например, MQTT или XMPP. CoAP, с другой стороны, является облегченной версией, несет те же недостатки архитектуры запрос / ответ. Однако он предназначен для запуска через UDP, делая способным использование ограниченных ресурсов устройства.

AMQP.Открытый протокол для передачи сообщений между компонентами системы.Advanced MessageQueuingProtocol(AMQP) - это протокол, который возник из финансовой отрасли. Он может использовать различные транспортные протоколы, но он предполагает базовые надежные транспортные протоколы, такие как TCP. Протокол AMQ Робеспечивает асинхронную Издатель\подписчик связь с сообщениями. Его основным преимуществом является ее функция хранения и пересылки, которая обеспечивает надежность даже после сбоев сети. Это обеспечивает надежность и следующие гарантии доставки сообщений:

- как минимум раз: означает, что сообщение отправлено либо, если оно доставлено или нет.
- По крайней мере один раз: означает, что сообщение будет доставлено, безусловно, один раз, возможно, больше.
- Точно раз: означает, что сообщение будет доставлено только один раз. Безопасность обрабатывается с использованием протоколов TLS / SSL над TCP.

В Последнее время исследованияпоказали, что AMQP имеет низкую вероятность успеха в низкой пропускной способности [10], но он увеличивается по мере увеличения пропускной способности . Другое исследование показывает, что по сравнению с AMQP с вышеупомянутым REST, AMQP может отправить большее количество сообщений в секунду. Кроме того, было сообщено, что AMQP среди 2000 пользователей распределенных по пяти континентам может обрабатывать 300 миллионов сообщений в день. Кроме того, JPMorgan, которая является американской банковской и финансовой компанией использует AMQP и отправляет 1 млрд сообщений в день.

WEBSOCKET.Протокол Websocket был разработан в рамках инициативы HTML по налаживанию каналов связи по протоколу TCP. WebSocket является ни запрос/ответ, ни издатель/подписчик протокол. В WebSocket клиент инициализирует рукопожатие с сервером, чтобы установить WebSocket сессии. Само рукопожатие похоже на HTTP, так что веб-серверы могут обрабатывать сессию WebSocket а также HTTP соединения через тот же порт . Тем не менее, то, что приходит после рукопожатия не соответствует правилам HTTP. На самом деле, во время сеанса, http-заголовки удаляются и клиенты и серверы могут обмениваться сообщениями в асинхронном полнодуплексном соединении. Сеанс может быть прекращен, когда он больше не нужен с серверной или клиентской стороны. WebSocket был создан, чтобы уменьшить накладные расходы связи Интернет, обеспечивая в реальном времени полную дуплексную связь. Существует

также Websocket, называемый суб-протокол Websocket приложения обмена сообщениями протокола (WAMP), которая обеспечивает архитектуру Издатель\подписчик. Систем обмена сообщениями. WebSocket работает над надежной TCP и не реализует никаких механизмов надежности по себе. При необходимости, заседания могут быть обеспечены с помощью WebSocket над TLS / SSL. В ходе сессии сообщения Websocket имеют только 2 байта. Как сообщили соответствующие исследования, HTTP, опрос повторяет информацию заголовка когда скорость передачи данных увеличивается, таким образом увеличивая задержки. WebSocket по оценкам, обеспечивает снижение три-к-одному латентности против полудуплексном HTTP опроса.WebSocket не предназначен для устройств с ограниченными ресурсами, как предыдущие протоколы и архитектуры, основанные на клиент -сервер. его не устраивают приложения ИВ. Однако он предназначен для общения в реальном времени, он безопасен, он минимизирует накладные расходы и с использованием WAMP он может обеспечить эффективные системы обмена сообщениями. Таким образом он может конкурировать с любым другим протоколом, работающим по протоколу TCP.

CoAP протокол.ConstrainedApplicationProtocol(CoAP) разработанный в InternetEngineeringTaskForce (IETF) для устройств с низким употреблением энергии, синхронизируется запросом и ответом на прикладном уровне. Он был разработан на основе HTTP, поэтому совместим с HTTP.

CoAP предназначен для использования в очень простых электронных устройств, которые позволяют им взаимодействовать в интерактивном режиме через Интернет. Подробное описание имеются в технических документах под номером RFC 7252.

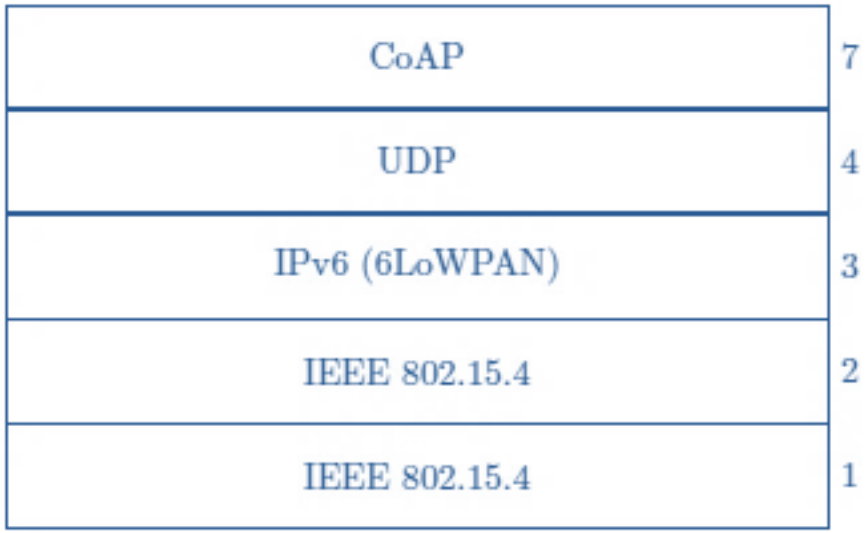


Рис.10. CoAP протокол

CoAP работает поверх UDP. Как и HTTP он использует методы GET,POST,PUT,DELETE для взаимодействия клиент-сервер. Кроме того протокол можно использовать как мультикастовую рассылку, что не позволяют протоколы созданные поверх TCP. Так как CoAP работает поверх UDP, разработаны механизмы, которые позволяют достигать необходимой надёжности. 2 бита в заголовке сообщения позволяют улучшить качество обслуживания отсюда вытекают 4 разновидности сообщений:

1. Confirmable - сообщение которые требует подтверждение (ACK) может быть отправлен синхронно или, асинхронно отдельным последующим сообщением
2. Non-Confirmable-сообщения которые не требуют обратного ответа.
3. Acknowledgment- сообщение-которое отправляется в ответ на прием сообщений типа Confirmable
4. Reset- такие сообщения отправляются в случае если принимающее сообщение повреждена.

Существует также механизм Stop-and-Wait- который задерживает отправление и отправляет повторно сообщения Confirmable. И также в CoAP имеется ID номера размером 16 бит, которые позволяют обнаруживать дубликат.

CoAP C HTTP Mapping позволяет клиенту CoAP через прокси-сервер подключаться к HTTP и обратно передавать от HTTPserver-а клиенту CoAP.

Если CoAP создана на устройствах ИВ и M2M он не включает в себя инструменты безопасности, безопасностью в таких случаях занимается DTLS поверх UDP. Он обеспечивает целостность данных, и безопасно передавать аутентификации, и алгоритмы криптографических ключей. Несмотря на то что DTLS защищает передачу UDP, он не был предназначен для ИВ.

MQTTпротокол. MessageQueueTelemetryTransport(MQTT)-был разработан компанией IBM для упрощения работы M2M. Это асинхронный протокол, построенный поверх TCP. В качестве сервера MQTT играет так называемый брокер. Каждый клиент получает каждый раз обновления на темы которые он подписался. MQTT разработан для низких пропускно-способных узлов и для устройств с низким употреблением энергии.

Реализация уровня QoS для протокола MQTT.

1. Fireandforget:- Сообщение отправляется один раз и не требуется подтверждения.
2. Deliveredatleastonce:- повторно отправленные сообщения требуют подтверждения.
3. Deliveredexactlyonce:-Механизм четырёх-ступенчатого рукопожатия клиент-брокера.

Несмотря на то, что MQTT разработан на TCP протоколе, его заголовки меньше заголовков других протоколов написанных поверх TCP.

Как и HTTP протокол MQTT для обеспечения безопасности использует методы аутентификации TLS/SSL.

6. Операционная система Contiki

В качестве программного обеспечения для микропроцессорных систем сенсорных узлов можно использовать открытые операционные системы :Contiki**Ошибка! Источник ссылки не найден.Ошибка! Источник ссылки не найден.**, TinyOSиFreeRTOS, а также ряд коммерческих операционных систем реального времени. Contiki, Tiny OS и Free RTOS — открытые операционные системы, исходный код который доступен в сети Интернет, разработка их ведется сообществом на публичной основе. Contiki, как ожидается, будет одной из основных операционных систем, которые будут использоваться для соединения триллионов устройств (или “вещей”) Интернета Вещей. Различные функции ОС включают управление программами/процессами, управление ресурсами, управление памятью и коммуникационное управление. Contiki является операционной системой с открытым кодом, она предназначена для встроенных сетевых систем, в частности, для умных устройств. Первая версия Contiki была выпущена командой разработчиков, работавших в сфере промышленности и научных центрах, в 2003 году **Ошибка! Источник ссылки не найден.Ошибка! Источник ссылки не найден.**.

Contiki поддерживает полный стек IP сети со стандартными протоколами, такими как UDP, TCP и HTTP, а также стандарты для маломощных сетей, таких как 6lowpan, RPL и CoAP. Стек протоколов ContikiIPv6 был разработан для Contiki компанией Cisco, полностью сертифицирован

в рамках программы ReadyLogoIPv6. Contiki является первой операционной системой для сенсорных сетей, которая обеспечивает TCP/IP связь (с использованием стека uIP (microIP) — это открытый TCP/IP-стек/модуль, который разработан для микроконтроллеров с 8- и 16-битной архитектурой). Сетевой стек Contiki показан на рисунке 33.

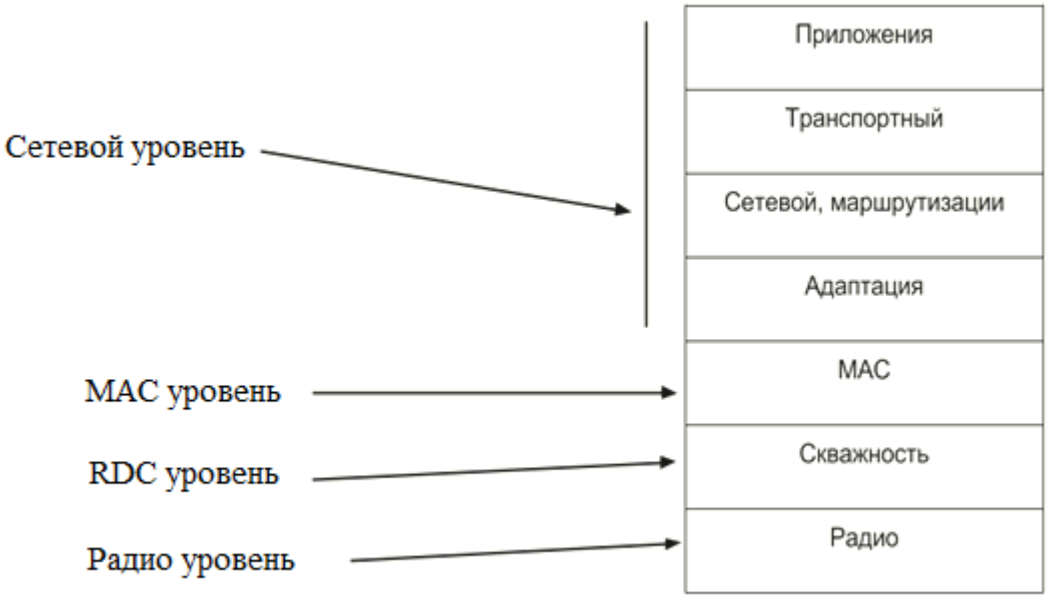


Рисунок 11 – Сетевой стек Contiki

Операционная система Contiki реализована на языке программирования «Си» и поддерживает ряд различных микропроцессоров и аппаратных конфигураций. Операционная система Contiki обеспечивает подключение IPv4 и IPv6 через стеки протоколов uIP и uIPv6. На рисунке 34 показан стек протоколов IPv6. Стек uIPv6 — единственный стек IPv6 для умных устройств, которые получили сертификацию IPv6 Ready. Таким образом, Contiki предлагает модули для решения различных задач на требуемом сетевом уровне.

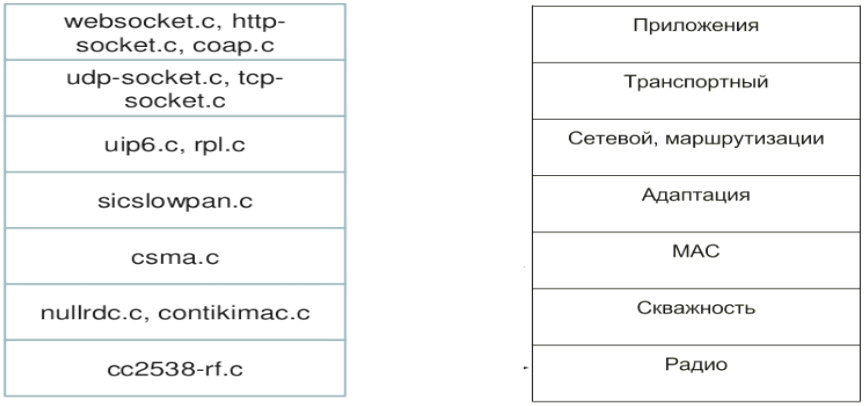


Рисунок 12. Стек протоколов IPv6 Contiki

При конфигурации по умолчанию Contiki использует 2 килобайта ОЗУ и 40 килобайт ПЗУ рис.35. Contiki состоит из ядра, которое управляется событиями, программы во время исполнения

загружаются и выгружаются динамически. Процессы используют облегчённую потоковую модель — протопотоки (protothreads), которые обеспечивают линейный потоковый стиль инициализации ядра.

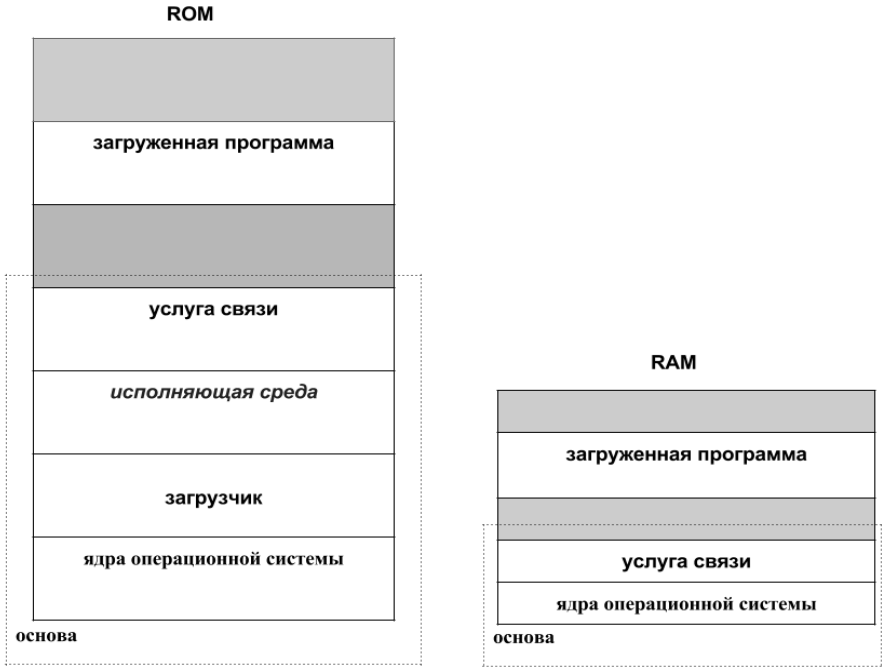
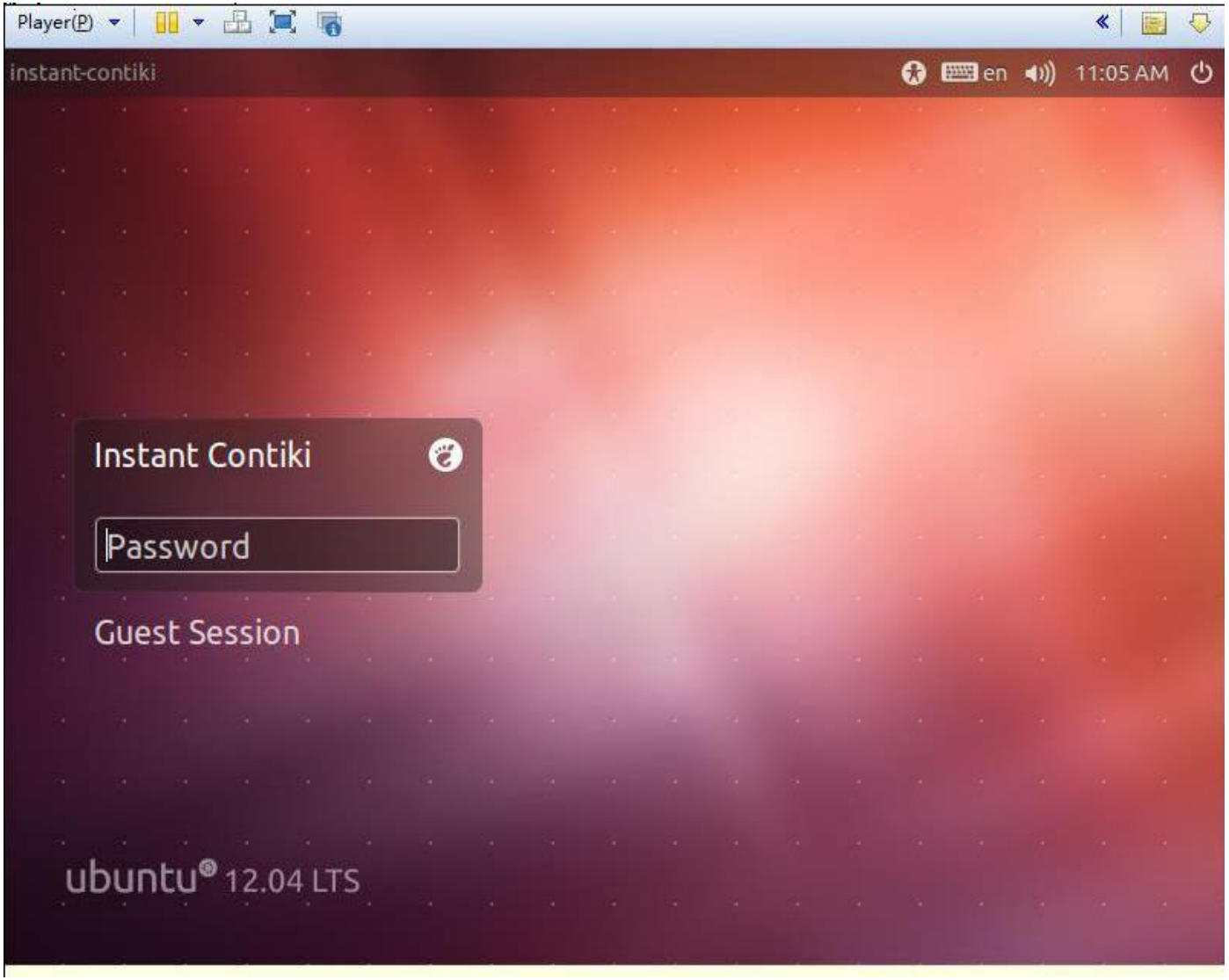


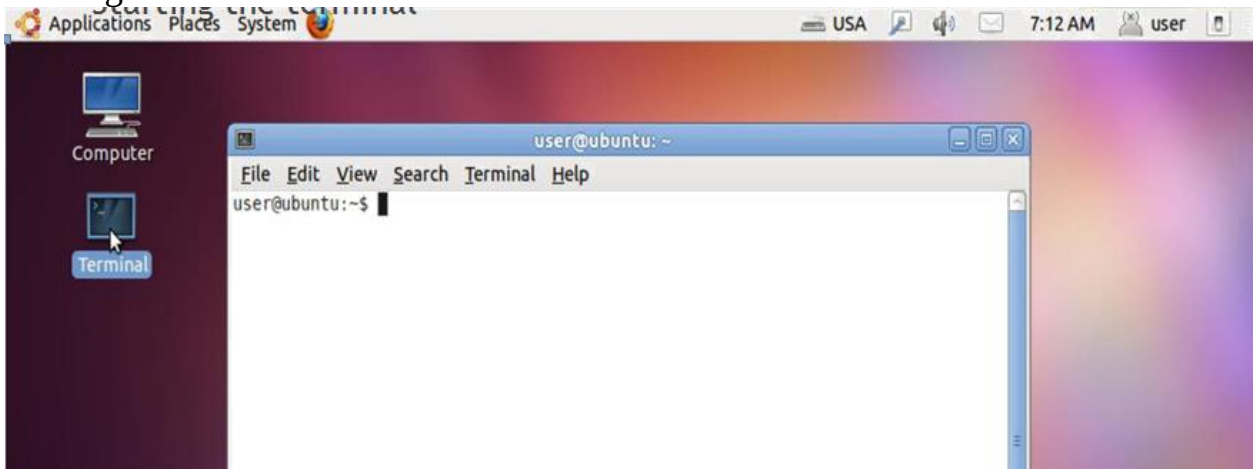
Рисунок 13 – Структура системы

Многопоточный режим с приоритетами в системе Contiki реализован с помощью библиотеки приложений, которые выполняются над ядром, управляемым событиями. Приложения, обеспечивающие многопоточную обработку, компонуются с выполняющимся приложением по мере необходимости, т.е. если оно явно требует многопоточной модели вычислений. Исполнительная система Contiki разделяется на две части – основа (core) и пользовательские программы. Основа (core) состоит непосредственно из ядра операционной системы (kernel), базовых сервисов и фрагментов библиотек поддержки, в том числе исполняющей среды. Разделяемая функциональность реализуется через сервисы как некоторая форма разделяемых библиотек. Эти сервисы можно обновлять или замещать динамически независимо друг от друга во время выполнения, что, по мнению разработчиков, ведет к гибкой структуре системы.


Player(P)
instant-contiki
en
11:05 AM
Instant Contiki
Password
Guest Session
ubuntu® 12.04 LTS

tarting: Open a terminal window

- We will now compile and start Cooja, the Contiki network simulator.
- Starting the terminal

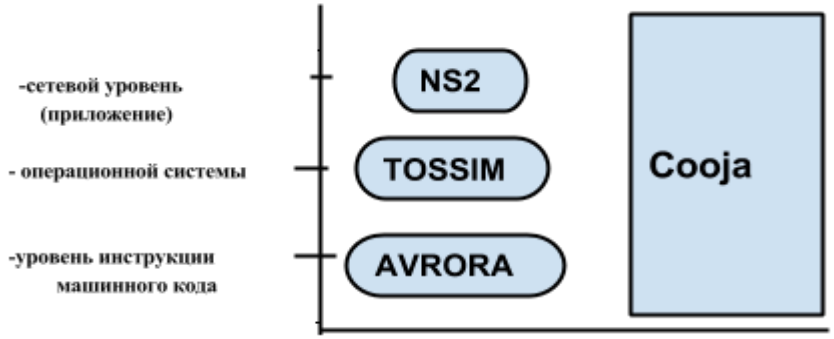


6.1. COOJA симулятор

Устройства с Contiki часто функционируют в качестве узлов крупных беспроводных сетей. Разработка и отладка программного обеспечения для таких сетей является трудоемкой задачей. COOJA — стимулятор сетевых процессов Contiki упрощает данную процедуру, предоставляя функции моделирования устройств и сетей **Ошибка! Источник ссылки не найден.**.

Моделируемый узел в COOJA имеет три основных свойства: данные в памяти, тип узла и его комплектующие. Тип узла определяет общие свойства для узлов. Например, узлы одного типа запускают единый программный код на одинаковой моделируемой аппаратной архитектуре.

Одной из отличительных особенностей COOJA от других симуляторов (NS2, TOSSIM, AVRORA) является то, что COOJA позволяет одновременно контролировать три различных уровня: сетевой уровень, уровень операционной системы и уровень инструкций машинных кодов. COOJA может выполнять программы Contiki двумя различными способами: либо как скомпилированный код прямо на хосте CPU, либо запуск на эмуляторе микропроцессора TIMSP430.



Рисунок– Сравнение COOJA с другими инструментами моделирования

Сетевой уровень используется в основном разработчиками протоколов маршрутизации, где аппаратная отчетность опущена.Уровень, в основном, управляет радиочастью сенсоного узла и специфическими свойствами.

В уровне операционной системы основная задача- смоделировать выполнение собственного кода операционной системы .

И, наконец, на уровне машинного кода команд есть возможностьдля узлов, имеющих различные базовые структуры, моделировать их с помощью java на основе микроконтроллеров вместо скомпилированной системы Contiki.

COOJA поддерживает все описанные уровни и по этой причине может моделировать и операционную системуTinyOS, необходимую для поддержки 6LoWPAN и RPL, и радио обмен пакетами между эмулированными узлами и узлами на основе Java.

Сообщения могут передаваться в различной среде распространения: симулятор позволяет моделировать четыре вида беспроводных каналов. UnitDiskGraphMedium (UDGM) - Constant Loss–канал с постоянными потерями в области ограниченной кругом, UnitDiskGraphMedium (UDGM) –DistanceLoss с потерями, зависящими от расстояния в круге и DirectedGraphRadioMedium (DGRM) – потери зависящие от направления.

Модель NoRadioTraffic не разрешает радиосвязь и, следовательно, не может быть использована для моделирования БСС. Модель UDGM - постоянные потери - это модель беспроводного канала, в которой зона связиописывается как идеальный круг. Все узлы вне этого круга не получают пакеты, а узлы в пределях радиуса связи (круга) получают все сообщения.

Заданная максимальнаядистанция связи умножается на отношение текущей выходной мощности к максимальной выходной мощности моделируемого устройства, а результирующая мощность соотносится с моделируемым расстоянием.

Например, если дальность связи узлов 200 м,а текущая выходная мощность составляет половину максимальной, то радиус зоны связи (круга) составляет 100 м.

Модель UDGM – при потерях, зависящих от расстояния, аналогична предыдущей модели, но она расширена и учитывает два дополнительных фактора. Во-первых, она учитывает интерференцию, в случае которой пакеты теряются на расстоянии итерференции, которое превышает дальность связи.

Так, все передачи пакетов, производимые одновременно удаляются по причине невозможного поведения системы.

Во-вторых, может быть задан коэффициент успешных передач и приемов: успех передачи или приема пакета определяются на основе двух вероятностей, SUCCESS_RATIO_TX (если неуспех, то ни одно устройство не получает пакет) и SUCCESS_RATIO_RX (если неуспех, то только узел назначения не получает пакет).

DGRM - это модель, которая создает топологию сетина основе границ. Она главным образом используется для определения коэффициентов успешных передач и задержки распространения в асимметричных каналах.

Гибкость конфигурации COOJA обеспечивается тем, что многие части модели могут быть легко заменены или расширены с дополнительными функциональными возможностями.

Например, описанная выше модель радиоканала определяется интерфейсами и плагинами. Интерфейсы описывают некоторые свойства узла, такие какрасположение, радиоканал, батарею или последовательный порт.

В отличие от интерфейсов, плагины используются для обеспечения взаимодействия при моделировании. Они обычно предоставляют пользователю графический интерфейс. Но эти инструменты могут быть использованы и без графического интерфейса, при работе COOJA через терминал.

Основными плагинами являются следующие: Simulation Visualizer, Timeline, Log Listener and Contiki Text Editor. SimulationVisualizer, они позволяют легко настроить топологию сети. Пользователь может размещать, и перетащать узлы, изменять настройки среды распространения сигнала (зоны передачи и риема, интерференции), и получать некоторую полезную информацию из узлов, такую как идентификатор, IP-адрес, выходные данные, радио трафик, тип удаления, атрибуты узлов, радио среды и состояние индикаторов.

Временной график отображает состояние радио для каждого узла модели с помощью разных цветов: активен (серый), выключен (нет цвета), передача пакетов (синий), прием пакетов (зеленый) и помех (красный).

Для получения более подробной информации достсточно кликнуть правой кнопкой мыши на шкале интересов. Анализатор лог-записей содержит данные по каждому радио соединению между моделируемыми узлами. Эти данные содержат время, исходный узел, конечные узлы и полезноесодержимое пакета, отправленные во время моделирования.

текстовый редактор Contiki управляет и автоматизирует выполнение моделирования с помощью программ, написанных на Java скрипте.

Они позволяют запускать и останавливать моделирование, установить время ожидания моделирования, динамически добавлять и удалять узлы, захватить выходы, датчики узлов и взаимодействовать с плагинами COOJA.

После остановки моделирования, пользователь может сохранить его (в том числе используемые плагины) в файлы с форматом CSC. Конфигурационный файл представляет собой XML-структуру, что облегчает изменение некоторых частей модели.

6.1.1. Архитектура COOJA

В этом разделе анализируется структура симулятора COOJA. Программное обеспечение симулятора написано на языке Java и он включен в проекте Contiki; исходный код COOJA расположен по этому пути: contiki/tools/cooja/java/se/sics/cooja

Так как имеется исходный код симулятра, то есть возможнось изменять любую его часть, ниже приводится краткое описание java-классов, которые используюится для взаимодействия с плагином.

interfaces/Radio: Эти интерфейсы позволяют моделировать поведение трансивера, в них определены методы, которые управляют состоянием, положением и мощностью радиопередатчика, и приемникасообщений.

Замечание: файл описываетсвойства узлов. Его методы возвращают идентификатор узла, тип узла, т. е. telosb, telosa и micaz и память узла.

Positioner:: класс используется, для определения исходныого положения узлов.

RadioConnection: описывапет радиосоединение между одним источником и узлом назначения, а также узлами, создающими помехи. Этот файл используется классом среды распространения для создания приложений описания трафика.

ConvertedRadioPacket: это наиболее часто используемый пакет в COOJA. В целях поддержки связи между уровнями, все трансиверы должны поддерживать передачу и прием этого пакета.

Simulation: файл управляет выполнением каждого моделирования. Он является видимым классом и поэтому если моделирование проходит обновления (например, вставки, удаления узлов и смена состояния моделирования),каждый плагин COOJA может его видеть.

RadioMedium: этот класс является абстрактным и так каждая среда распространения для реализации COOJA использует его. Зарегистрированные в среде распостранениятрансиверы могут отправлять и получать радио пакеты.

radiomediums/AbstractRadioMedium: обеспечивает базовую функциональность для реализации среды распростанения, он обрабатывает регистраций радио, активное соединение между узлами и записывает логи всех зарегистрированных радио интерфейсов.

Зарегистрированная мощность радиосигнала обновляется всякий раз, при изменении параметров среды распространения; есть три фиксированных уровней, т. е. нет трафика, шум и прием данных.

radiomediums/UDGM: UnitDiskGraphRadioMedium зона связи в виде кругов. Он использует два разных диапазона параметров, один для передачи и один для помех от других радиостанций; оба прараметира растут с ростом индикатора выходной мощности. Кроме того, этот класс имеет два разных коэффициента для моделирования удачных передач, приемов и случайных ошибок на канале.

radiomedium/DirectedGraphMedium: среда распространения реализуется через границы, где каждое звено поддерживает задержки распространения и коэффициенты успешной передачи.

plugins/analyzers/IPHC Packet Analyzer: анализирует каждый пакет, отправляемый через среду распространения, обеспечивает получение информации о нагрузке, размере и многих других параметрах.Все анализаторы могут расширить этот класс.

plugins/analyzers/IEEE802154Analyzer: анализатор позволяет изучать пакеты переданные через уровни 802.15.4, различая сообщения данных и сообщения ACK, которые образуют кадр: для каждого кадра - класс определяет адрес источника и адрес назначения.

plugins/analyzers/IPHCPacketAnalyzer: класс сканирует каждую датаграмму (пакетов IPv6), отправляемую от клиента к узлу назначения. В этом случае анализ проводится с помощью протокола 6LoWPAN, определяя следующие данные для каждого сообщения: IPv6-адрес источника, адрес назначения протокола IPv6, udp-порт источника и udp-порт назначения.

plugins/analyzers/ICMPv6Analyzer: имеет функцию анализа маршрутизации трафика. Поскольку трафик маршрутизируется по протоколу RPL, класс разделяет сообщения на следующие категории: пакеты DIS, пакеты DIO, пакеты DAO и пакетов DAOACK. Плагин коллекцииметрик вклается в каталог плагинов.

6.1.2. Файл формата CSC

Этот файл определяет способ выполнения и сбора показателей в соответствии с изменяющимися трафиком передачи данных при моделировании COOJA. Для достижения этой

посталенной цели, на этапе тестирования используется инструмент сбора показателей через оболочку shell, без использования GUI. В этом случае, для запуска моделирования COOJA, используется команда ant run_nogui -Dargs = / path / fils.csc. В отличие от моделирования в GUI, в качестве параметра команды ant требуется файл конфигурации моделирования (.csc). Данный файл в формате XML содержит структуру моделирования, который определяет что сбор показателей. Другими словами, документ в виде CSC определяет:

используемую среду распространения и соответствующие линии между узлами;

Программа двоичное изображение, работающая на узлах;

расположение узлов в сети;

среда сбора показателей COOJA, получаемых в процессе моделирования;

программа javascript для управления длительностью симуляции и запуском методов плагина для расчета показателей и получения статистических данных. Практически, для проверки команда ant запускается несколько раз, с равными долями трафика передачи данных. Эта процедура повторяется три раза напри моделировании, для того, чтобы усреднить статистические значения, и удалить ложные значения. Скорости передачи данных варьируется от 1 пакета/с до 11 пакетов/с, увеличивась каждый раз на 0,5 пакетов/с.

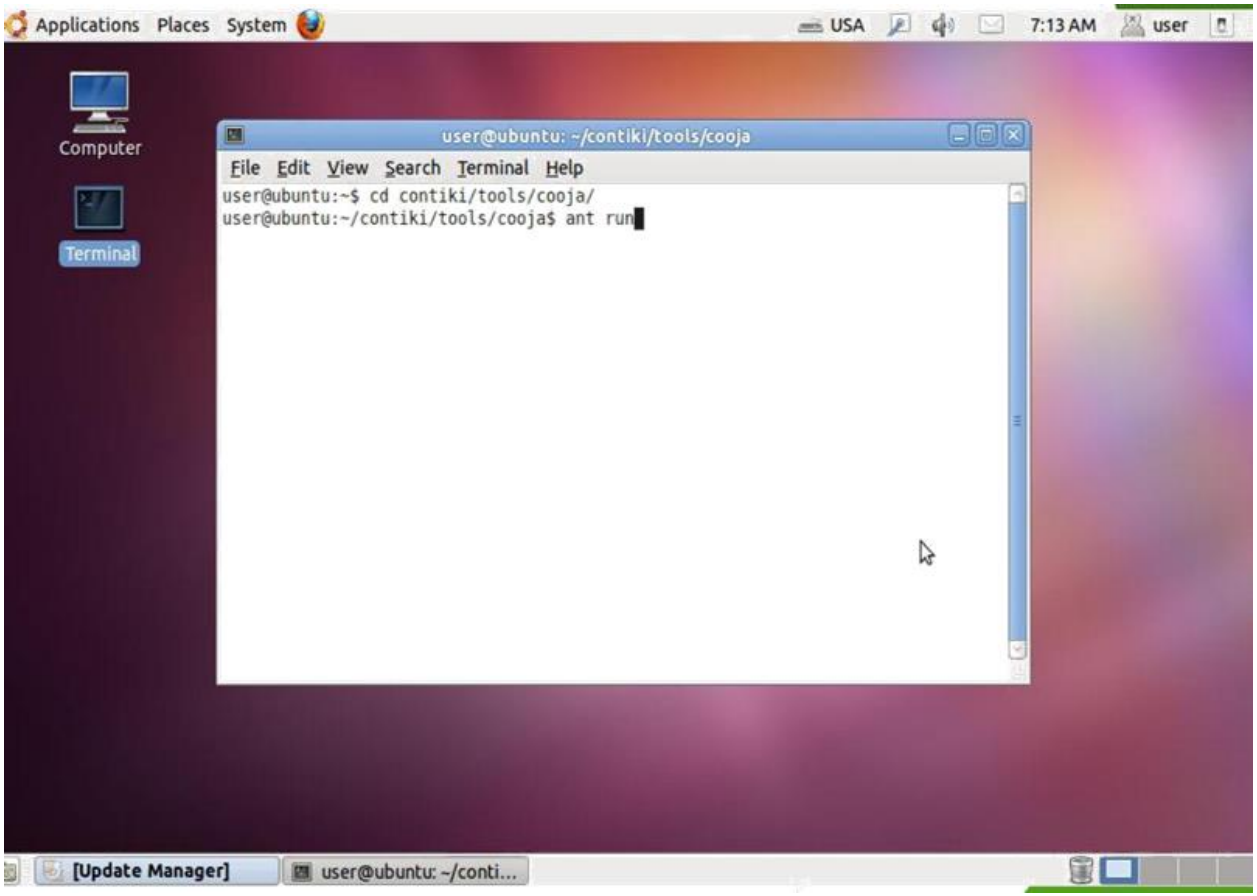
Для того, чтобы изменить скорость передачи данных, среда сбора показателей будет изменять константу PACKET_INTERVAL, определенную в Makefile приложения Test_RPL. Вместо того, чтобы управлять объемом трафика, генерируемого клиентами приложений, этап тестирования требует, чтобы каждый сенсорный узел отправлял 500 датаграмм на узел назначения. Чтобы установить это значение, среда сбора показателей будет изменять константу PACKET_NUMBER, размещенную в том же файле PACKET_INTERVAL.

6.1.3. **Интерфейсы cooja симулятора**

Можно запустить симулятор Cooja с помощью следующих команд:

cd contiki -2.6/tools/cooja

ant run


Applications
Places
System
USA
7:13 AM
user
Computer
Terminal
user@ubuntu: ~/contiki/tools/cooja
File Edit View Search Terminal Help
user@ubuntu:~$ cd contiki/tools/cooja/
user@ubuntu:~/contiki/tools/cooja$ ant run
[Update Manager]
user@ubuntu: ~/conti...

сначала появится большой текст, потом в коне можно увидеть окно интерфейса Cooja, как показано на рисунке 5

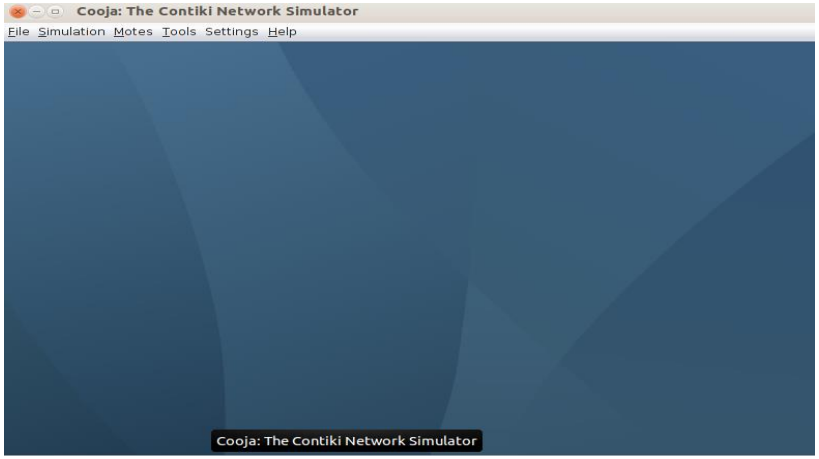


Рис 5.Окно интерфейса симулятора Cooja

Creating Simulation

- Click the **File** menu and click **New simulation**

Прежде всего, нам нужно будет создать новое моделирование с помощью кнопку “File > New Simulation”. при нажатии этой кнопки мы увидим окно, показанное на рисунке 6.

После этого можно выбрать более подходящее название, а затем выбрать радио среды, которая лучше всего подходит для типа моделирования. вполне боле подходящий для всех это (Unit Disk Graph Medium - UDGM) .

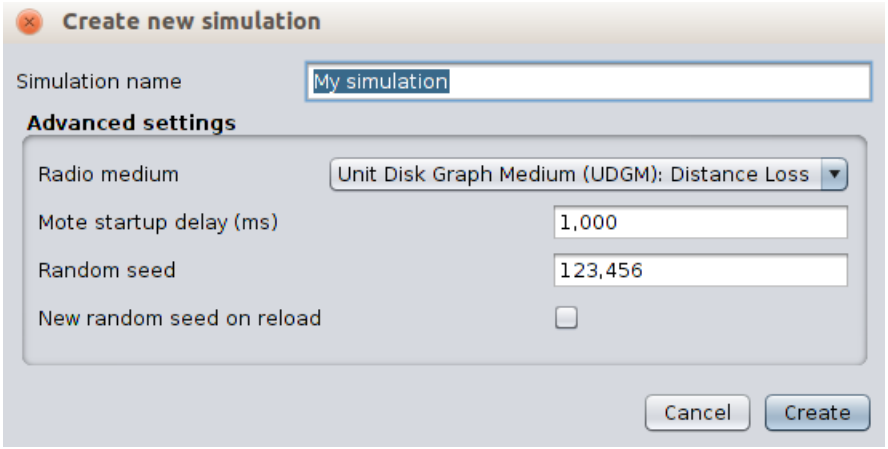


Рис . Создание нового интерфейса моделирования

Когда уже все готово можно нажать кнопку "Create", и появится новое окно но уже с интерфейсом моделирования рисунок 7 который состоит из 5 окон :

- окно сети показывает физическое расположение сети, т.е. можно разместить узлы по необходимости для того, чтобы сформировать топологию.

- окно управления моделирования позволяет начать, остановки и перезагрузки моделирование. Она также позволяет контролировать скорость, с которой осуществляется моделирования.

-Окна результатов можно фильтровать показанный результат в виде строки.

- Окно Timeline показывает события, которые происходят на каждом узле над Timeline моделирования. Эти события могут быть радио трафик, LED или что-либо еще.

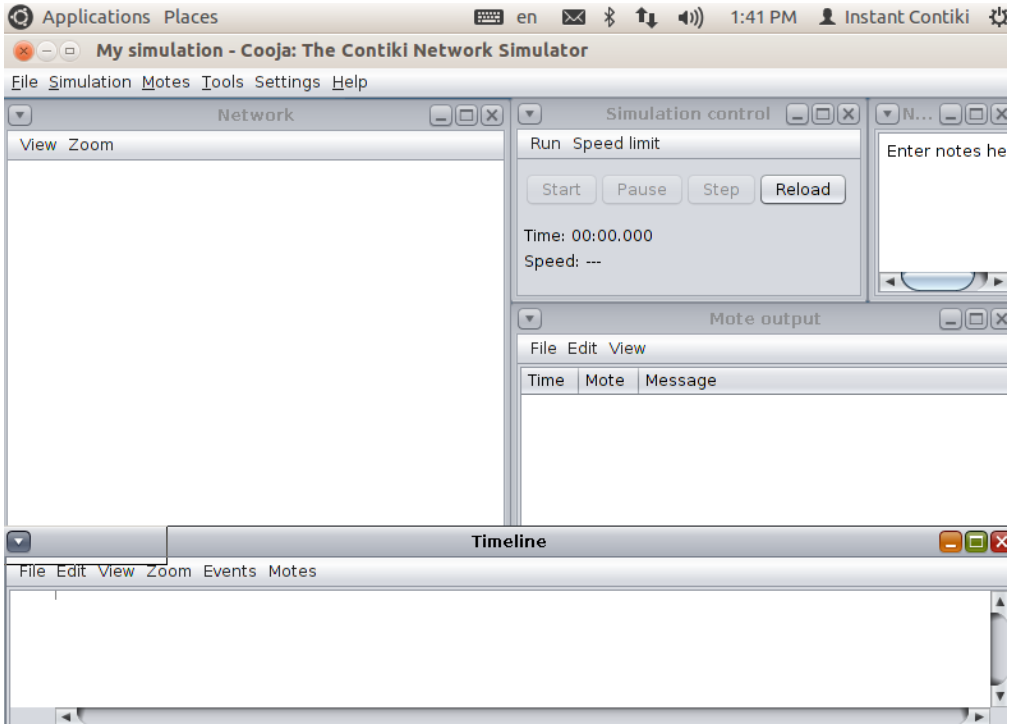
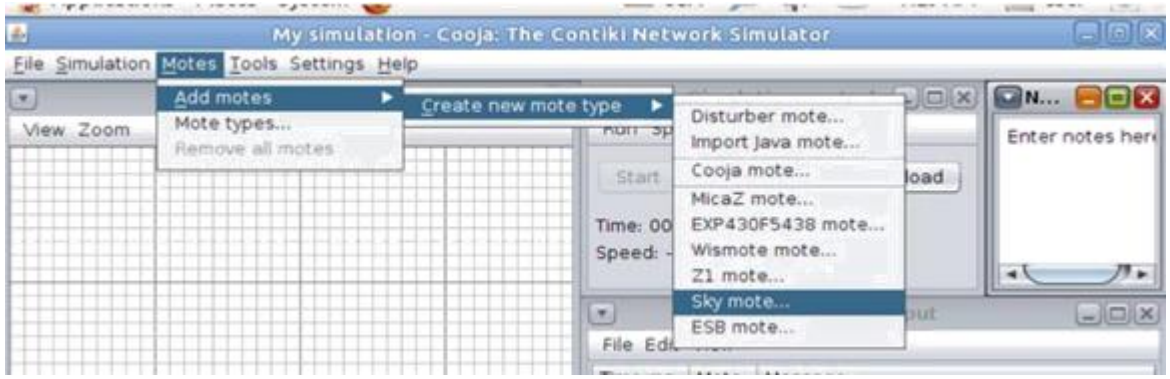


Рис . Интерфейс моделирования

Следующий этап это установить тип узлов . В Cooja можно создать различные типы узлов, определить их параметры и назначить их программный код. Нажать на“Motes > Add Motes > Create New Mote Type > Sky Mote”

Тут можно дать тип узла полезное описание. Например пусть будет этот тип пограничный маршрутизатор Border Router как показано на рисунке 8



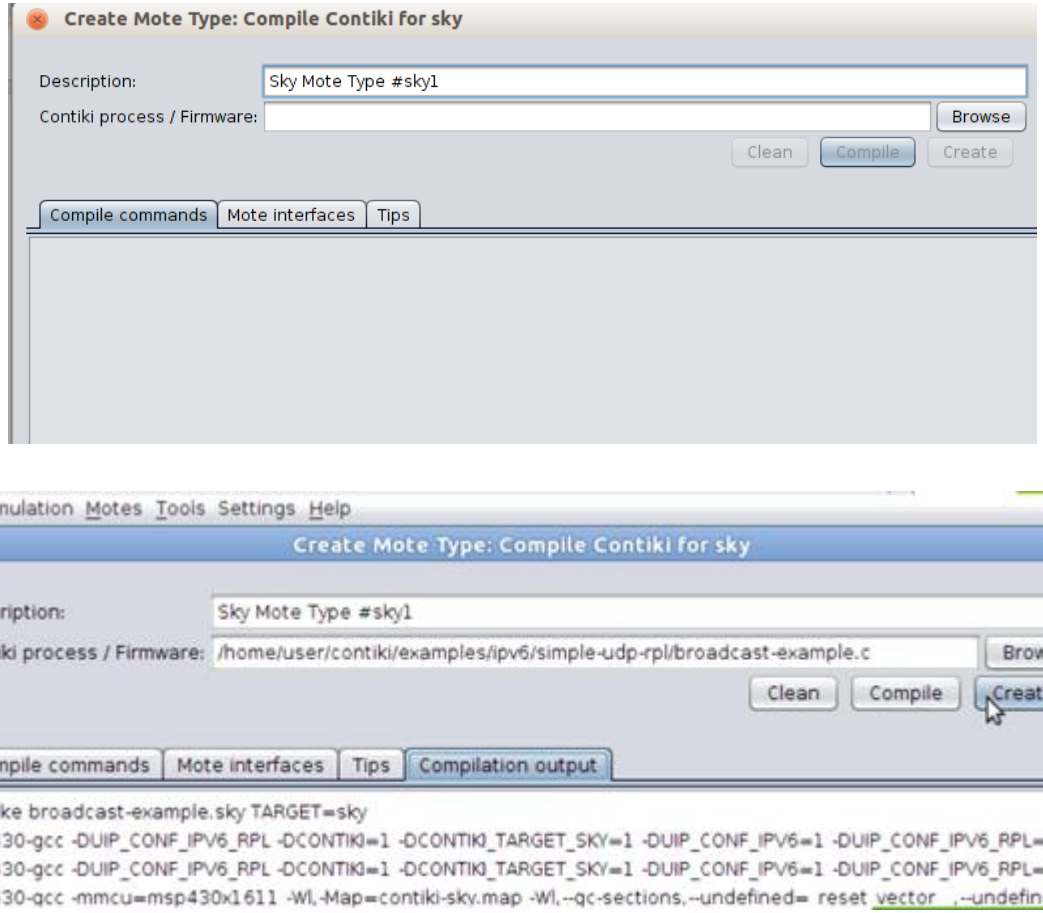
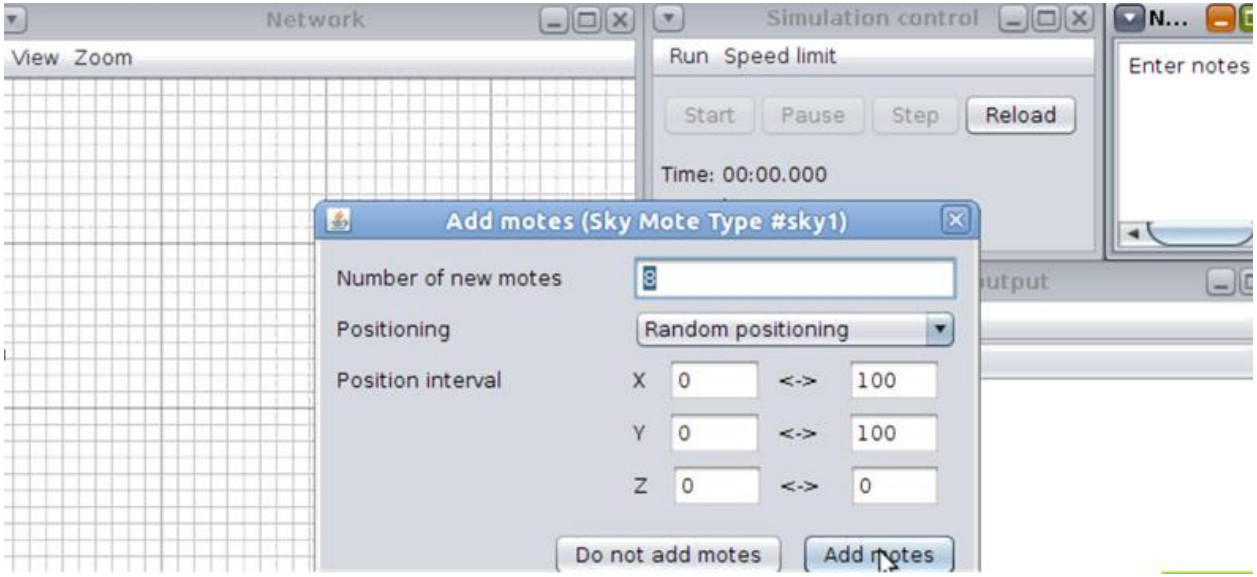
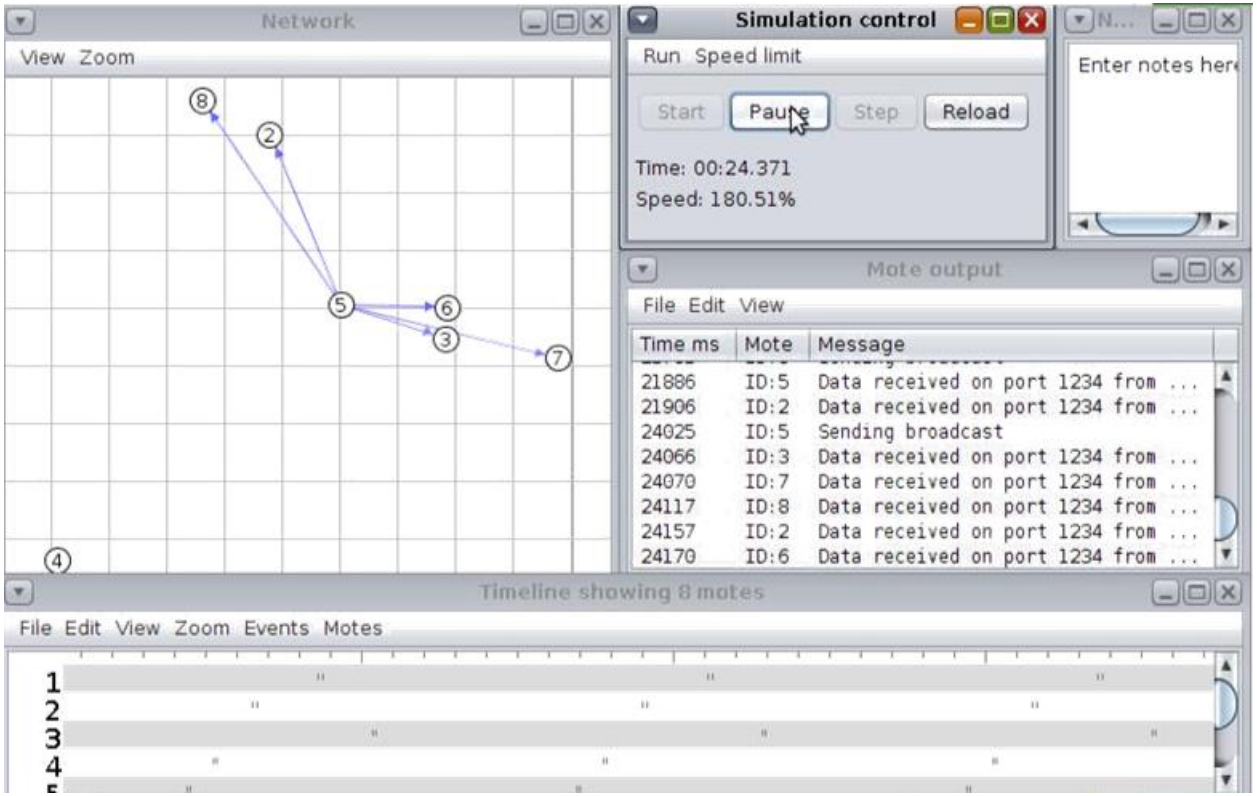


Рис. Интерфейс создания типа узла

Интерфейс симуляции

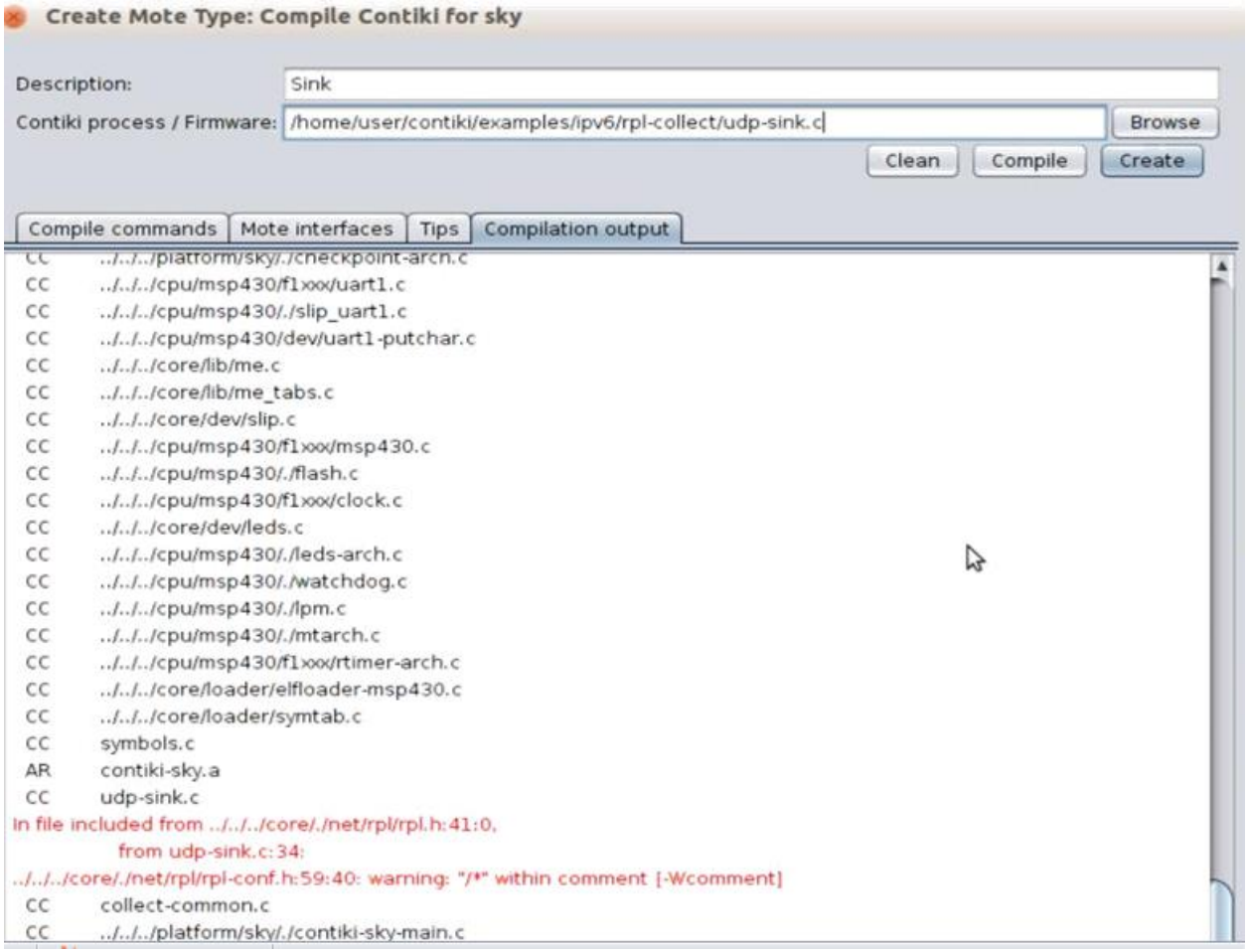

Network
View Zoom
Simulation control
Run Speed limit
Start Pause Step Reload
Time: 00:00.000
N...
Enter notes
Add motes (Sky Mote Type #sky1)
Number of new motes
8
Positioning
Random positioning
Position interval
X 0 <-> 100
Y 0 <-> 100
Z 0 <-> 0
Do not add motes
Add motes



Пример моделирования (RPL)

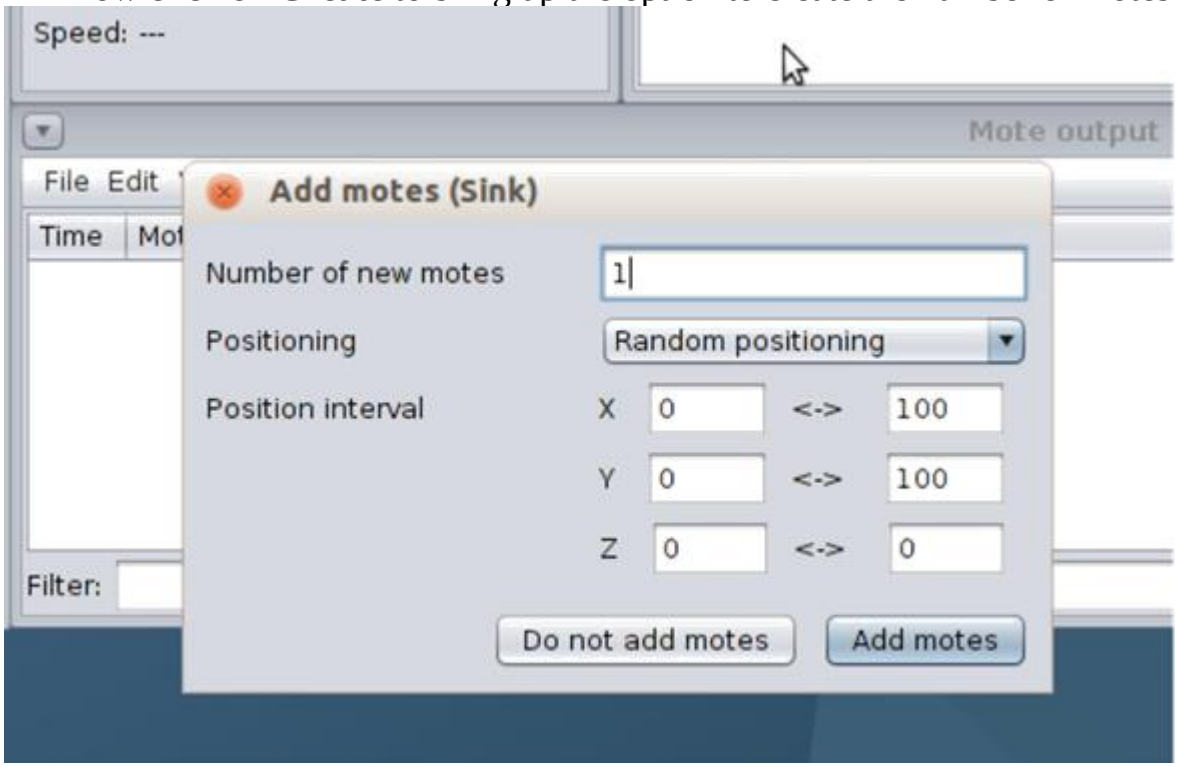
As this example involves the use of RPL the path selected is

- **/home/user/contiki/examples/ipv6/rpl-collect/udp-sink.c**.
- udp-sink.c is the actual C language firmware of the mote that will now be created.
- Click **Clean** to erase any previous compile of the mote and then **Compile**.



Создание узлов Cooja

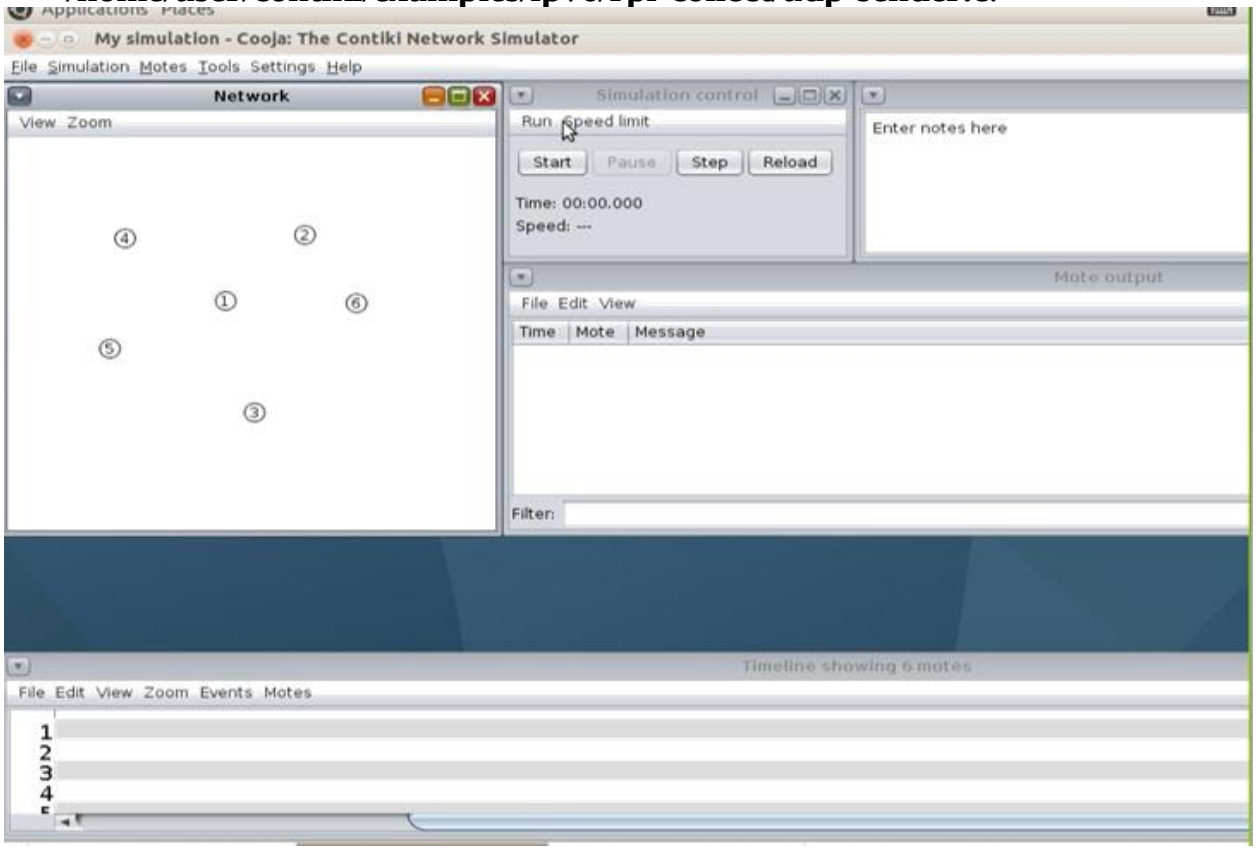
- Now click on **Create** to bring up the option to create the number of motes required.



Создания узлов отправления Cooja

- This process should be repeated for the Sender motes with the path for the mote firmware now

- **/home/user/contiki/examples/ipv6/rpl-collect/udp-sender.c**.



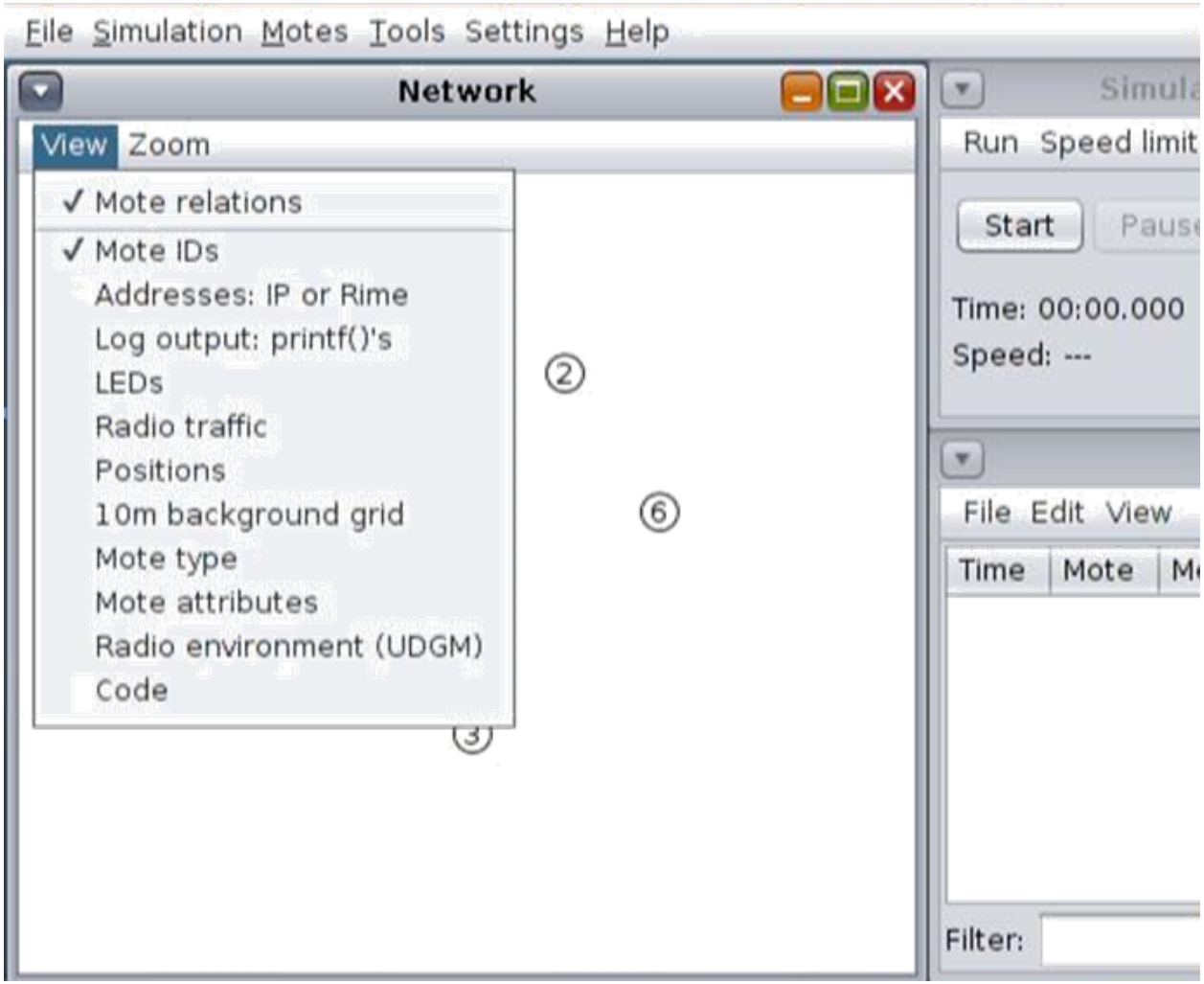
Сетевые настройки

- The **Timeline** window shows events over a period of time. This view can be filtered using the drop down menu and the output can be saved to a file if necessary.

- The **Mote Output** window shows the output window will display any printouts from the motes.

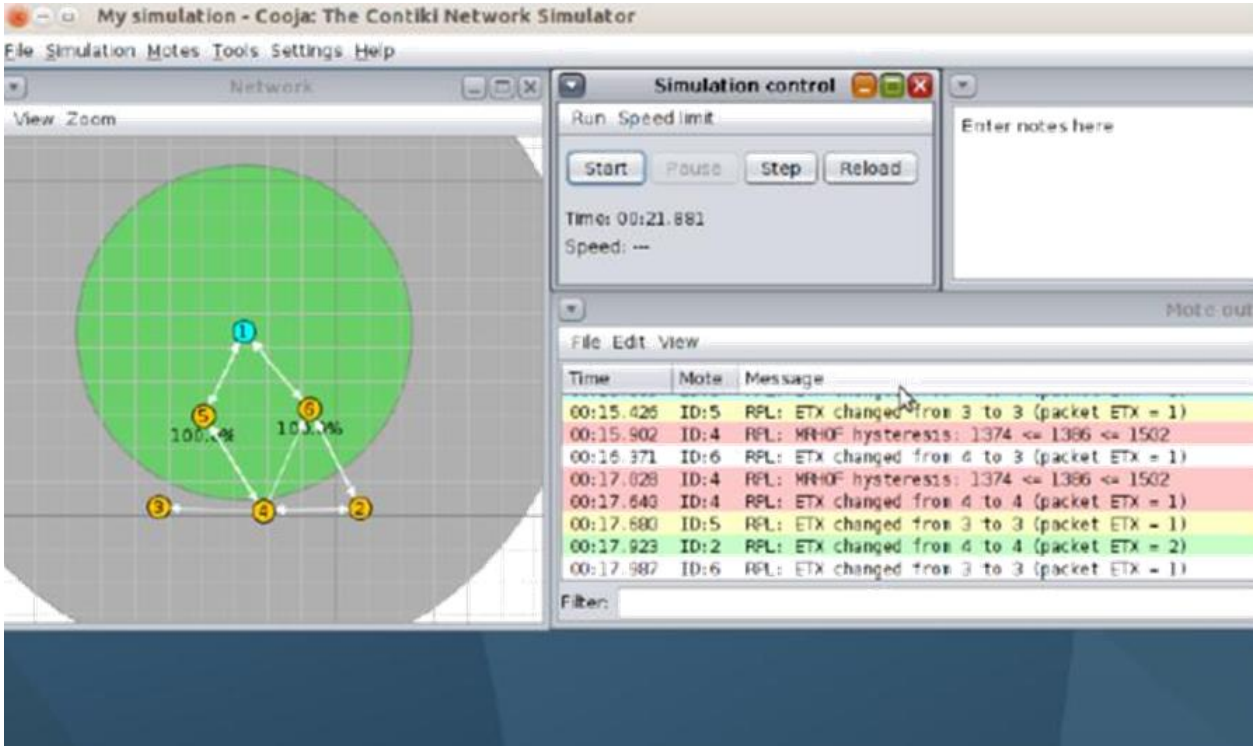
- The **Simulation Control** window is where the simulation is started, paused and stopped. The simulation can also be completely reloaded from this window although there are other options in this regard.

- The **Network** window displays the layout of the network motes. he **View** dropdown menu shows the various options



- The **'Addresses: IP or Rime'** option displays the addresses, in this case IPv6, of each mote.

- **‘Log output: printf()’s’** displays printf messages from the mote code inside the actual view window.
- **‘LEDs’** is useful in order to observe the LED lights on the simulated motes.
- The **‘10m background grid’** is extremely useful to give a sense of scale to the network layout. This displays a grid of 100m2 squares rather than a blank background.
- The **‘Radio Traffic’** option is extremely useful in animating the network once the simulation is running.
- **‘Mote type’** employs a colour scheme to show the difference between motes of different types.
- **‘Mote attributes’** allows the use of code to alter the colorizing of motes as well as other options and is not relevant to this discussion.
- **‘Radio environment (UDGM)’** displays the transmission range of any particular mote and is extremely useful when deciding upon the optimal position of motes within a simulated network.
- **‘Code’** merely displays code as it is being run and is not of relevance to this discussion.



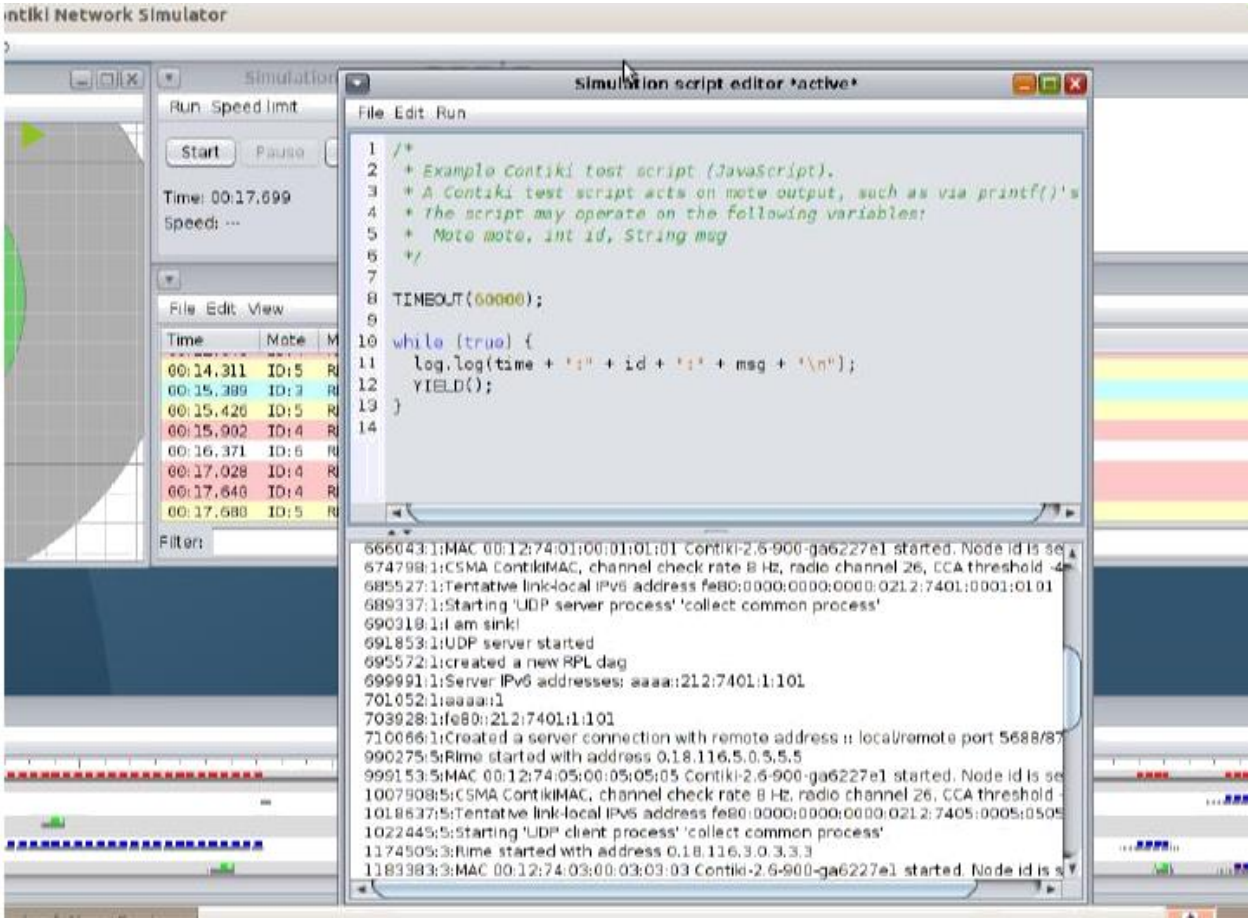
Узлы и Script Editor

Collect

- select the **‘Tools’** dropdown menu and the **‘Simulation Script Editor’**.

- the Script Editor can be used merely to display messages and to set a timer on the simulation.

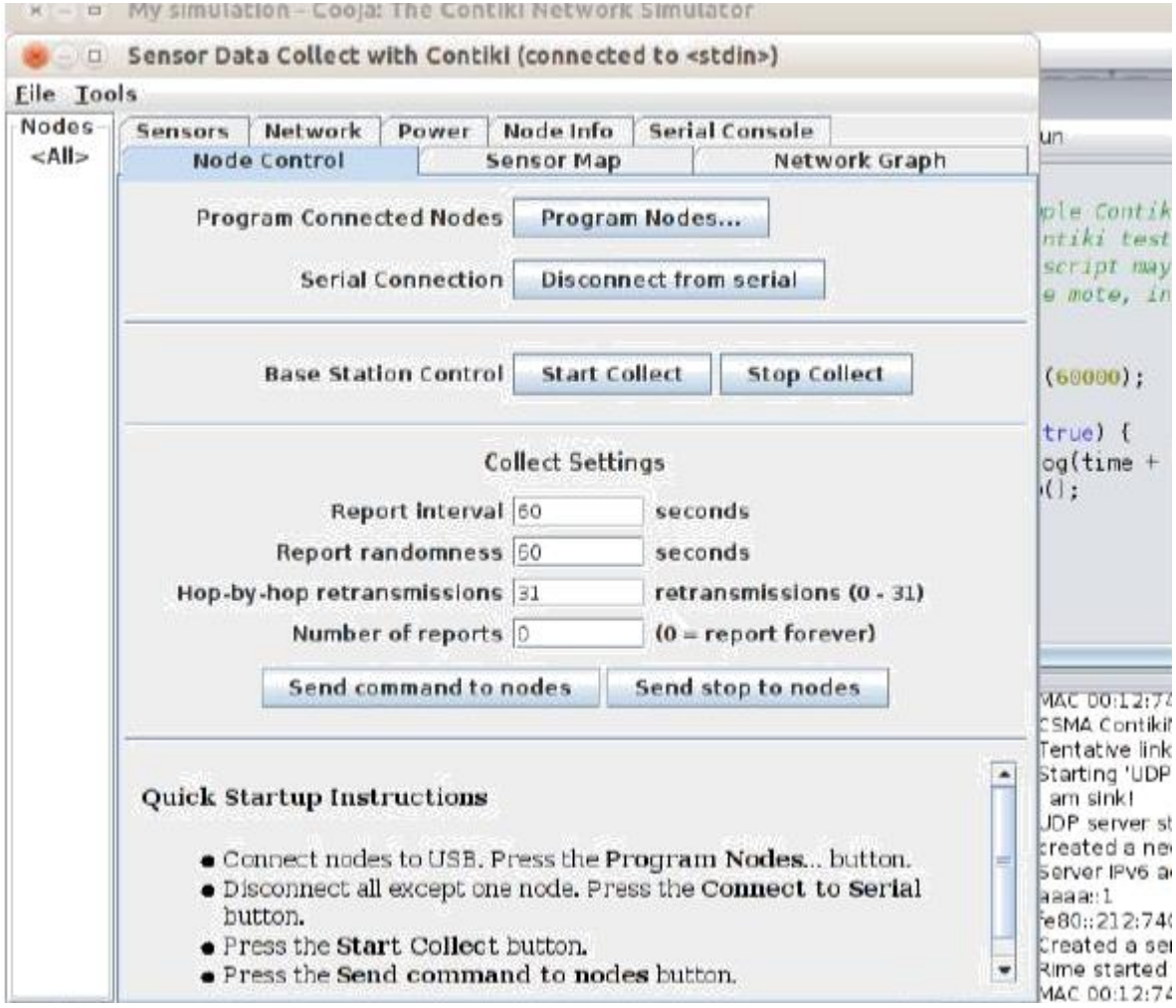
- In order to do this, within the Script Editor menu select the **‘File’** dropdown menu of which **‘Load example script’** is the only option. Then select **‘Just log all printf()’s and timeout’**.

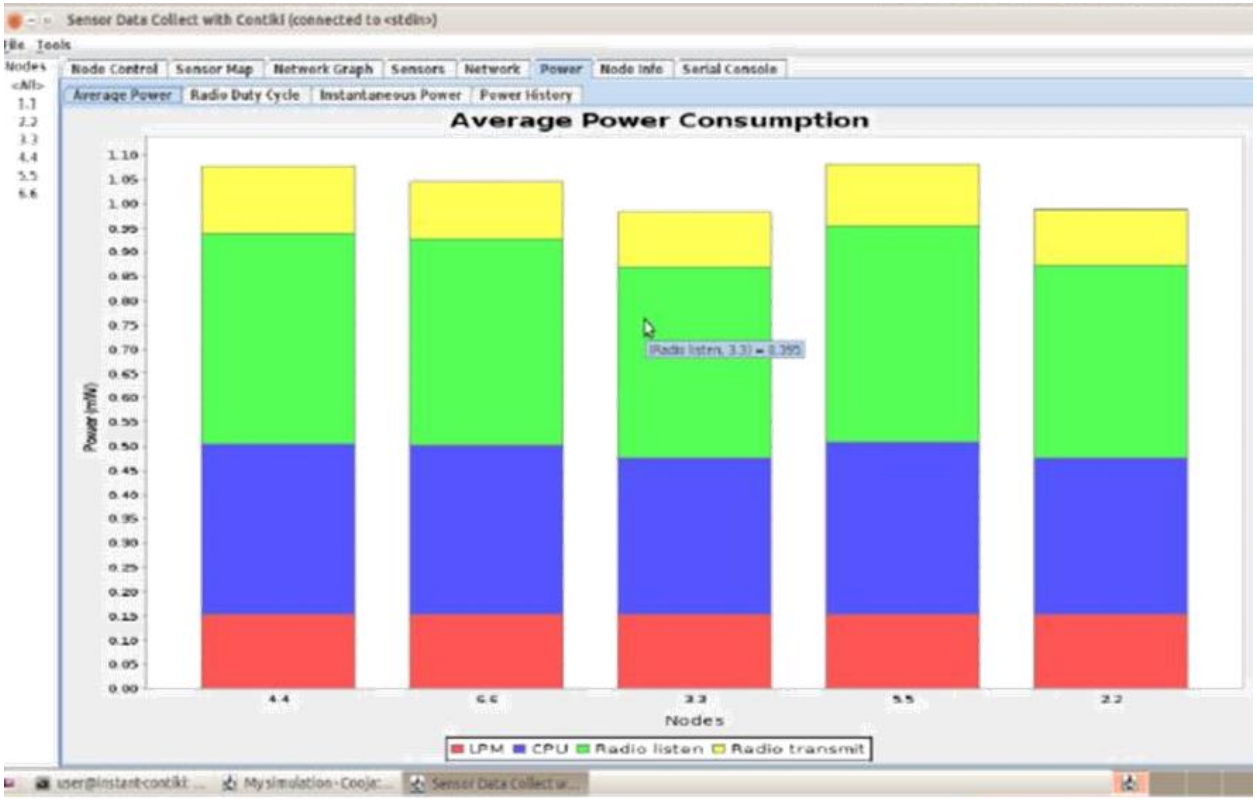


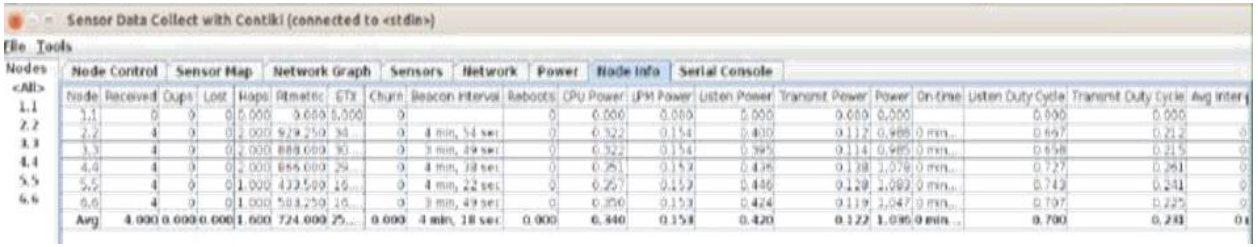
Collecting data

- select **‘Mote tools for Sky 1’** and then **‘Collect View’**. Alternatively use the **‘Tools’** dropdown menu, then

  ‘Collect View’ and finally **‘Sky 1’**.


My simulation - Cooja: The Contiki Network Simulator
Sensor Data Collect with Contiki (connected to <stdin>)
File Tools
Nodes
<All>
Sensors
Network
Power
Node Info
Serial Console
Node Control
Sensor Map
Network Graph
Program Connected Nodes
Program Nodes...
Serial Connection
Disconnect from serial
Base Station Control
Start Collect
Stop Collect
Collect Settings
Report interval
60
seconds
Report randomness
60
seconds
Hop-by-hop retransmissions
31
retransmissions (0 - 31)
Number of reports
0
(0 = report forever)
Send command to nodes
Send stop to nodes
Quick Startup Instructions
Connect nodes to USB. Press the Program Nodes... button.
Disconnect all except one node. Press the Connect to Serial button.
Press the Start Collect button.
Press the Send command to nodes button.


Sensor Data Collect with Contiki (connected to <stdin>)
File Tools
Nodes
<All>
1.1
2.2
3.3
4.4
5.5
6.6
Node Control
Sensor Map
Network Graph
Sensors
Network
Power
Node Info
Serial Console
Average Power
Radio Duty Cycle
Instantaneous Power
Power History
Average Power Consumption
Power (mW)
1.10
1.05
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
4.4
6.6
3.3
5.5
2.2
Nodes
LPM
CPU
Radio listen
Radio transmit



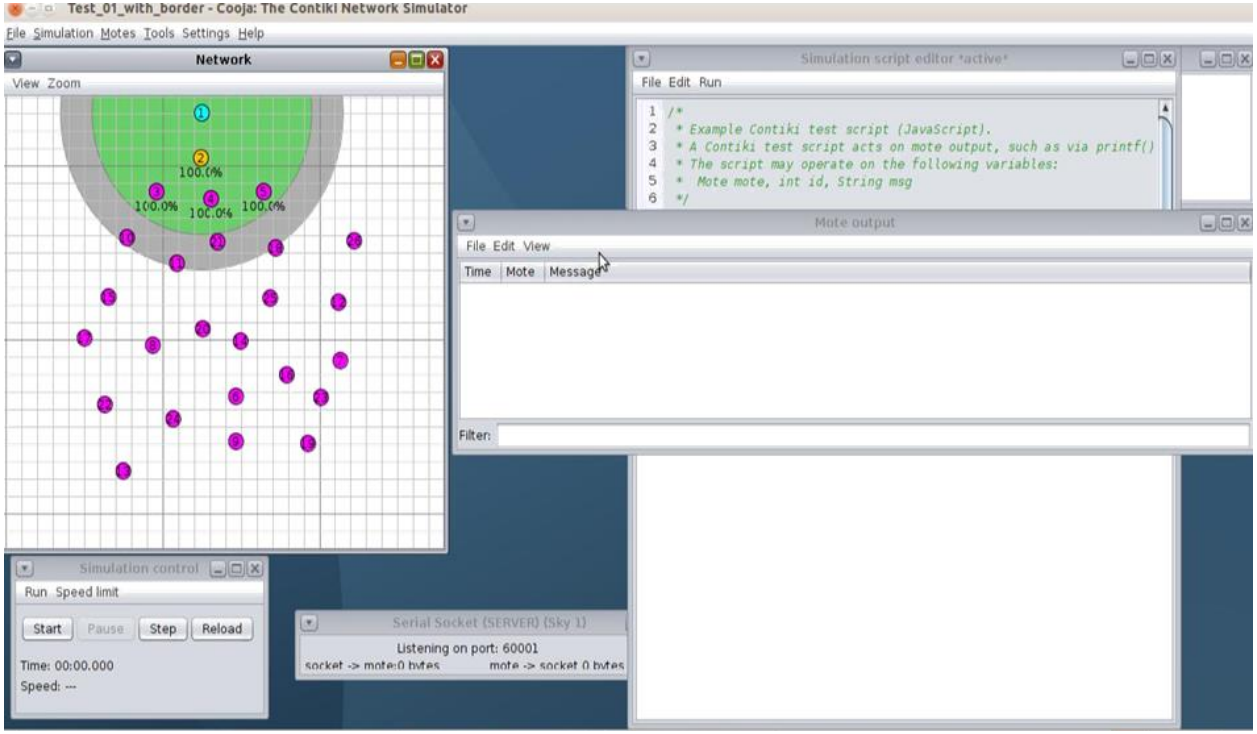
Маршрутизация IPv6

- This has a Border Router bridged to the RPL network in order to assign IPv6 addresses and routing within the network.

- Border Router mote is compiled using firmware to be found in

  /contiki/examples/ipv6/rpl-borderrouter/ border-router.c.

- To establish the bridge with the Border Router, first right click the Border Router mote 1 and select **‘Mote tools for Sky1’** then **‘Serial Socket (SERVER)’**. This will create a serial port on the Border Router which is accessible via UDP port number 60001 on the local machine[



- The simulation can now be started, however, a Terminal window should be open ready to enter the following commands which will establish the bridge to the Border Router:

user@instant-contiki:~/contiki$ cd tools user@instant-

contiki:~/contiki/tools$ make tunslip6 make: `tunslip6' is up to

date.

user@instant-contiki:~/contiki/tools$ sudo ./tunslip6 -a 127.0.0.1 aaaa::1/64

[sudo] password for user:

- This will the return the next slide, meaning a bridge has been established and IPv6 addresses assigned with prefix **aaaa::/64**. The Border Router's IPv6 address of **aaaa::212:7401:1:101** can be seen.
  - **slip connected to ``127.0.0.1:60001''**

opened tun device ``/dev/tun0'' ifconfig tun0 inet

`hostname` up ifconfig tun0 add aaaa::1/64

- **ifconfig tun0 add fe80::0:0:0:1/64**

ifconfig tun0

- **tun0 Link encap:UNSPEC HWaddr 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00**

inet addr:127.0.1.1 P-t-P:127.0.1.1 Mask:255.255.255.255

- **inet6 addr: fe80::1/64 Scope:Link**

inet6 addr: aaaa::1/64 Scope:Global

UP POINTOPOINT RUNNING NOARP MULTICAST MTU:1500 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0

TX packets:0 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:500

RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)

***** Address:aaaa::1 => aaaa:0000:0000:0000**

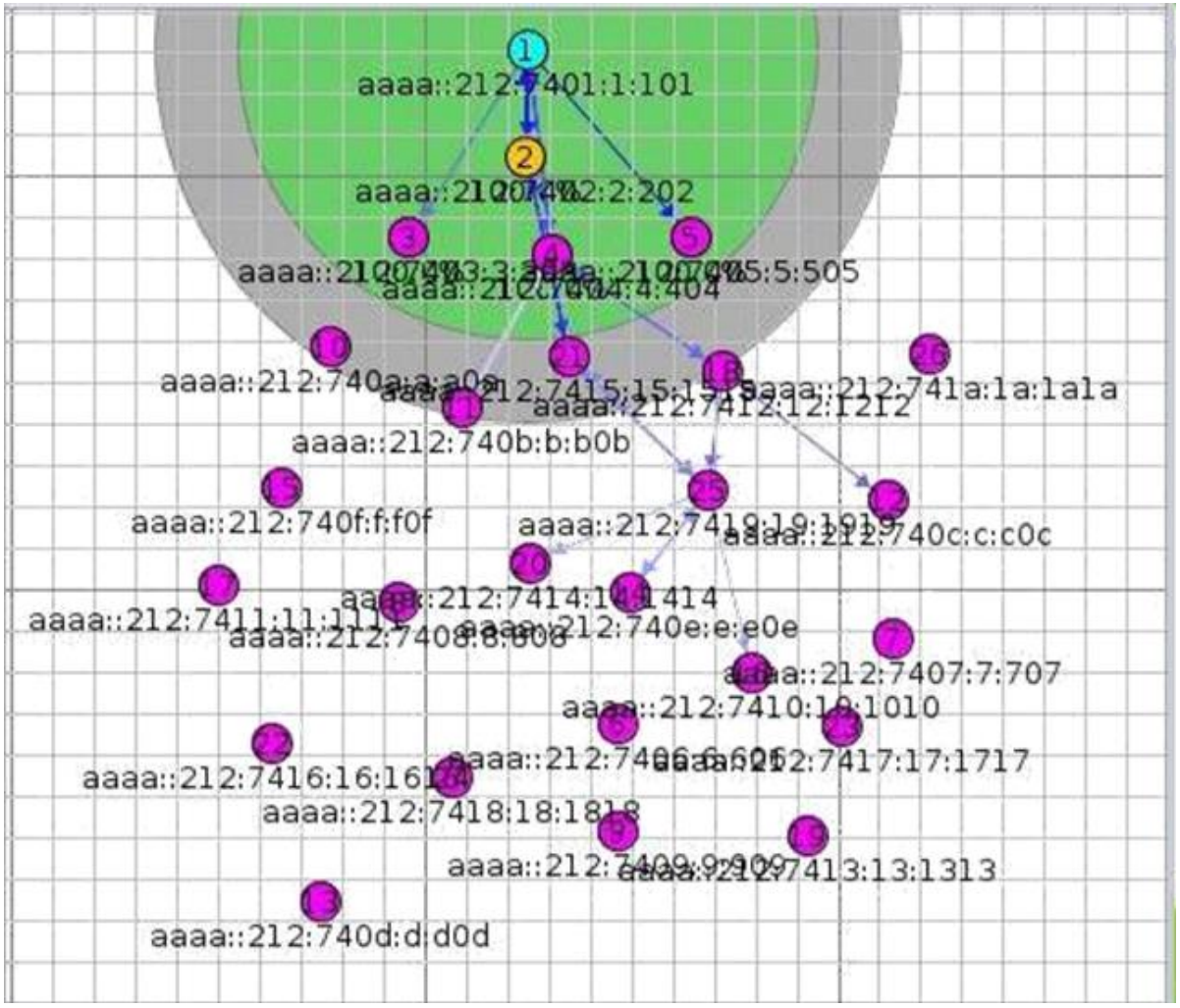
Got configuration message of type P

Setting prefix aaaa::

Server IPv6 addresses:

aaaa::212:7401:1:101

fe80::212:7401:1:101


aaaa::212:7401:1:101
aaaa::212:7402:2:202
aaaa::212:7403:3:303
aaaa::212:7404:4:404
aaaa::212:7405:5:505
aaaa::212:740a:a:a0a
aaaa::212:7415:15:151
aaaa::212:741a:1a:1a1a
aaaa::212:7412:12:1212
aaaa::212:740b:b:b0b
aaaa::212:740f:f:f0f
aaaa::212:7419:19:1919
aaaa::212:740c:c:c0c
aaaa::212:7411:11:1111
aaaa::212:7414:14:1414
aaaa::212:740e:e:e0e
aaaa::212:7408:8:808
aaaa::212:7407:7:707
aaaa::212:7410:10:1010
aaaa::212:7416:16:1616
aaaa::212:7417:17:1717
aaaa::212:7418:18:1818
aaaa::212:7409:9:909
aaaa::212:7413:13:1313
aaaa::212:740d:d:d0d

- While the simulation is running, the bridge can be used to Ping motes in the network as follows. Although not all are successful due to the volatility of the network:

user@instant-contiki:~$ ping6 -c 5 -s 8 aaaa::212:7401:1:101 PING

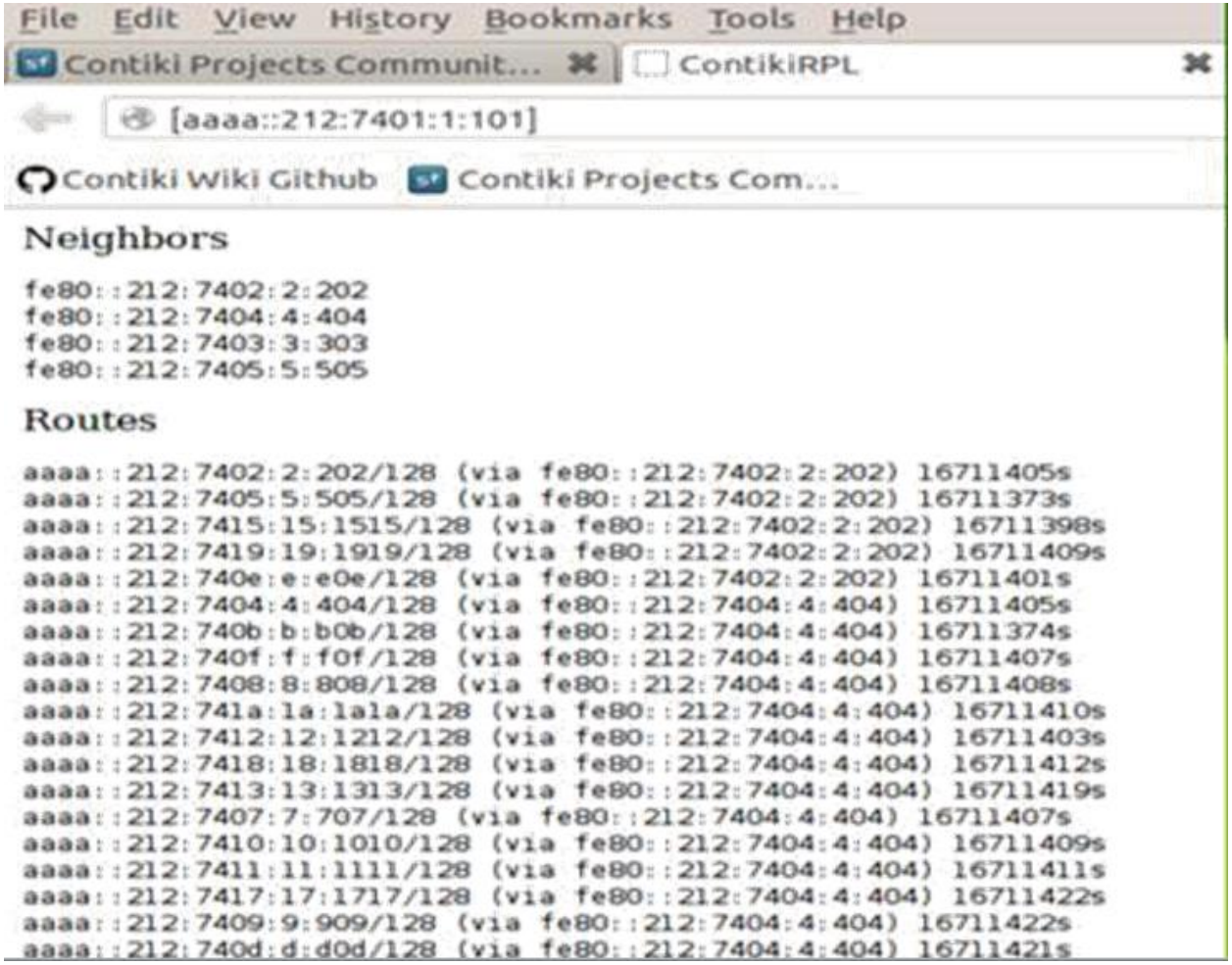
aaaa::212:7401:1:101(aaaa::212:7401:1:101) 8 data bytes

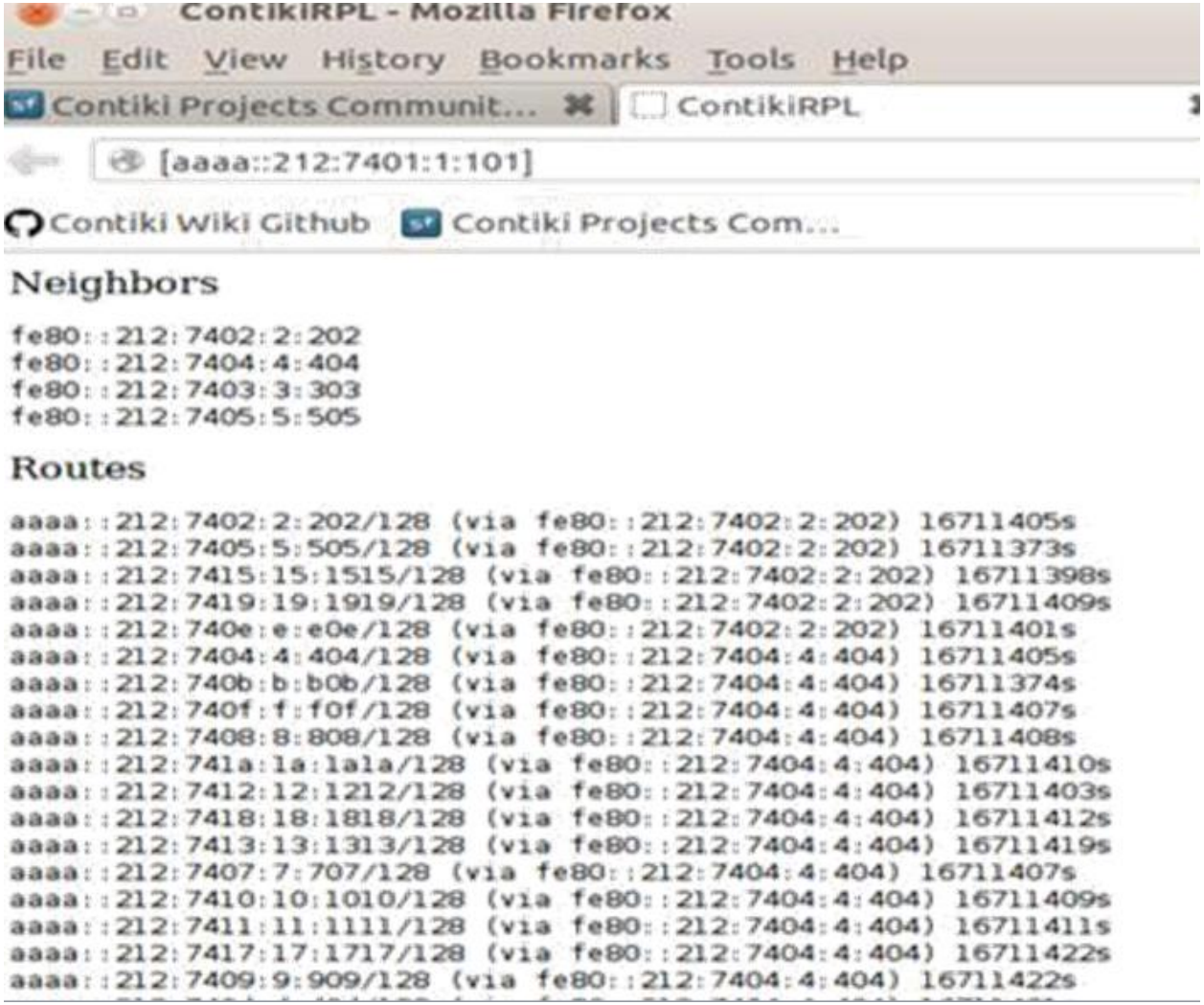
16 bytes from aaaa::212:7401:1:101: icmp_seq=1 ttl=64 time=26.8 ms 16 bytes from aaaa::212:7401:1:101: icmp_seq=2 ttl=64 time=21.2 ms 16 bytes from aaaa::212:7401:1:101: icmp_seq=3 ttl=64 time=19.1 ms 16 bytes from aaaa::212:7401:1:101: icmp_seq=4 ttl=64 time=31.8 ms 16 bytes from aaaa::212:7401:1:101: icmp_seq=5 ttl=64 time=15.9 ms

--- aaaa::212:7401:1:101 ping statistics ---

5 packets transmitted, 5 received, 0% packet loss, time 4008ms rtt

min/avg/max/mdev = 15.969/22.996/31.819/5.660 ms


File Edit View History Bookmarks Tools Help
Contiki Projects Communit...
ContikiRPL
[aaaa::212:7401:1:101]
Contiki Wiki Github
Contiki Projects Com...
Neighbors
fe80::212:7402:2:202
fe80::212:7404:4:404
fe80::212:7403:3:303
fe80::212:7405:5:505
Routes
aaaa::212:7402:2:202/128 (via fe80::212:7402:2:202) 16711405s
aaaa::212:7405:5:505/128 (via fe80::212:7402:2:202) 16711373s
aaaa::212:7415:15:1515/128 (via fe80::212:7402:2:202) 16711398s
aaaa::212:7419:19:1919/128 (via fe80::212:7402:2:202) 16711409s
aaaa::212:740e:e:e0e/128 (via fe80::212:7402:2:202) 16711401s
aaaa::212:7404:4:404/128 (via fe80::212:7404:4:404) 16711405s
aaaa::212:740b:b:b0b/128 (via fe80::212:7404:4:404) 16711374s
aaaa::212:740f:f:f0f/128 (via fe80::212:7404:4:404) 16711407s
aaaa::212:7408:8:808/128 (via fe80::212:7404:4:404) 16711408s
aaaa::212:741a:1a:1a1a/128 (via fe80::212:7404:4:404) 16711410s
aaaa::212:7412:12:1212/128 (via fe80::212:7404:4:404) 16711403s
aaaa::212:7418:18:1818/128 (via fe80::212:7404:4:404) 16711412s
aaaa::212:7413:13:1313/128 (via fe80::212:7404:4:404) 16711419s
aaaa::212:7407:7:707/128 (via fe80::212:7404:4:404) 16711407s
aaaa::212:7410:10:1010/128 (via fe80::212:7404:4:404) 16711409s
aaaa::212:7411:11:1111/128 (via fe80::212:7404:4:404) 16711411s
aaaa::212:7417:17:1717/128 (via fe80::212:7404:4:404) 16711422s
aaaa::212:7409:9:909/128 (via fe80::212:7404:4:404) 16711422s
aaaa::212:740d:d:d0d/128 (via fe80::212:7404:4:404) 16711421s



- Cooja can be manipulated to use either of MRHOF or OF0 quite easily

- **/home/user/contiki/core/net/rpl**

- Change **Makefile.rpl** as follows:

CONTIKI_SOURCEFILES += rpl.c rpl-dag.c rpl-icmp6.c rpl-timers.c \

rpl-mrhof.c rpl-ext-header.c

- This should be left unchanged to use **MRHOF** or changed to **rpl-of0.c** in the event that **Objective Function Zero** is to be used.

Change **rpl-conf.h** as follows:

*/

* The objective function used by RPL is configurable through the

* RPL_CONF_OF parameter. This should be defined to be the name of an

* rpl_of object linked into the system image, e.g., rpl_of0.

/*

#ifdef RPL_CONF_OF

#define RPL_OF RPL_CONF_OF #else

/* ETX is the default objective function. */ #define RPL_OF

rpl_mrhof

#endif /* RPL_CONF_OF */

- Currently, the only alternative to ETX available in Cooja is the Energy metric. To utilise this, change
- **rpl-conf.h** as follows:

*/

* Select routing metric supported at runtime. This must be a valid

* DAG Metric Container Object Type (see below). Currently, we only

* support RPL_DAG_MC_ETX and RPL_DAG_MC_ENERGY.

* When MRHOF (RFC6719) is used with ETX, no metric container must

* be used; instead the rank carries ETX directly.

/*

#ifdef RPL_CONF_DAG_MC

#define RPL_DAG_MC RPL_CONF_DAG_MC #else

#define RPL_DAG_MC **RPL_DAG_MC_NONE** #endif

/* RPL_CONF_DAG_MC */

*/

- In the event that the Energy metric is to be used this should be changed to **RPL_DAG_MC_ENERGY**.

6.2. Генерация сценариев для Cooja с помощью языка программирования Python

Для реализации новых протоколов в диссертации используется новые инструменты моделирования. Для генерации скрипов в формате xml используется язык программирования Python.

В ходе предыдущей исследовательской работы были разработаны инструменты моделирования которые включают в себя базовые сценарии БСС на язык программирования C++, со всеми необходимыми функциями для генерации трафика, и автоматизации, скриптами написанными на языке программирования Python и использованными для генерации случайных координат, случайных отметок времени, для запуска и остановки, и генерации других переменных. Одним из преимуществ использования этого инструмента является возможность создания полностью автоматического моделирования окружающей среды.

Функция для добавления плагинов

```
def add Plugin(xmlRoot, name, width, height, location_x, location_y, zet):
plug = ET.SubElement(xmlRoot, "plugin")
plug.text = name
w = ET.SubElement(plug, "width")
w.text = width
```

```
z = ET.SubElement(plug, "z")
z.text = zet
h = ET.SubElement(plug, "height")
h.text = height
lx = ET.SubElement(plug, "location_x")
lx.text = location_x
ly = ET.SubElement(plug, "location_y")
ly.text = location_y
return plug
```

Функция для добавления узла со случайными координатами

```
def addRandomMote(rootElem, moteIdNum, typeId, maxX, maxY):
randX = random.random()* maxX
randY = random.random()* maxY
simMote = ET.SubElement(rootElem, "mote")
bp = ET.SubElement(simMote, "breakpoints")
locConf = ET.SubElement(simMote, "interface_config")
    locConf.text = "se.sics.cooja.interfaces.Position"
```

```
cordX = ET.SubElement(locConf, "x")
    cordX.text = str(randX)
cordY = ET.SubElement(locConf, "y")
    cordY.text = str(randY)
cordZ = ET.SubElement(locConf, "z")
    cordZ.text = "0.0"
idConf = ET.SubElement(simMote, "interface_config")
    idConf.text = "se.sics.cooja.mspmote.interfaces.MspMoteID"
idNum = ET.SubElement(idConf, "id")
    idNum.text = str(moteIdNum)
moteTypeConf = ET.SubElement(simMote, "motetype_identifier")
    moteTypeConf.text = str(typeId)
```

Функция для добавления узла с заданными кординатами

```
def addStaticMote(rootElem, moteIdNum, typeId, posX, posY):
simMote = ET.SubElement(rootElem, "mote")
bp = ET.SubElement(simMote, "breakpoints")
locConf = ET.SubElement(simMote, "interface_config")
```

```
    locConf.text = "se.sics.cooja.interfaces.Position"
cordX = ET.SubElement(locConf, "x")
    cordX.text = str(posX)
cordY = ET.SubElement(locConf, "y")
    cordY.text = str(posY)
cordZ = ET.SubElement(locConf, "z")
    cordZ.text = "0.0"
idConf = ET.SubElement(simMote, "interface_config")
    idConf.text = "se.sics.cooja.mspmote.interfaces.MspMoteID"
idNum = ET.SubElement(idConf, "id")
    idNum.text = str(moteIdNum)
moteTypeConf = ET.SubElement(simMote, "motetype_identifier")
    moteTypeConf.text = str(typeId)
```

Генерациясценария

```
    # Simulation Title
simTitel = ET.SubElement(sim, "title")
    simTitel.text = "My simulation"
```

```
    # Random seed
simRand = ET.SubElement(sim, "randomseed")
    simRand.text = "123456"
    # Simulation delay
simDelay = ET.SubElement(sim, "motedelay_us")
    simDelay.text = "1000000"
```

Задание параметров среды распространения

```
simRadioMedium = ET.SubElement(sim, "radiomedium")
    simRadioMedium.text = "se.sics.cooja.radiomediums.UDGM"
trRange = ET.SubElement(simRadioMedium, "transmitting_range")
    trRange.text = "30.0"
inRange = ET.SubElement(simRadioMedium, "interference_range")
    inRange.text = "40.0"
ratioTx = ET.SubElement(simRadioMedium, "success_ratio_tx")
    ratioTx.text = "1.0"
ratioRx = ET.SubElement(simRadioMedium, "success_ratio_rx")
    ratioRx.text = "1.0"
```

Константы для описания типа и сценариев запускаемых на узлах

```
MOTE_TYPE = "se.sics.cooja.mspmote.SkyMoteType"
    MOTE_INTERFACE_LIST = [
        "se.sics.cooja.interfaces.Position",
        "se.sics.cooja.interfaces.RimeAddress",
        "se.sics.cooja.interfaces.IPAddress",
        "se.sics.cooja.interfaces.Mote2MoteRelations",
        "se.sics.cooja.interfaces.MoteAttributes",
        "se.sics.cooja.mspmote.interfaces.MspClock",
        "se.sics.cooja.mspmote.interfaces.MspMoteID",
        "se.sics.cooja.mspmote.interfaces.SkyButton",
        "se.sics.cooja.mspmote.interfaces.SkyFlash",
        "se.sics.cooja.mspmote.interfaces.SkyCoffeeFilesystem",
        "se.sics.cooja.mspmote.interfaces.Msp802154Radio",
        "se.sics.cooja.mspmote.interfaces.MspSerial",
        "se.sics.cooja.mspmote.interfaces.SkyLED",
        "se.sics.cooja.mspmote.interfaces.MspDebugOutput",
```

```
        "se.sics.cooja.mspmote.interfaces.SkyTemperature"
    ]
receiverMote = {
        'identifier' : "sky1",
        'description' : "Sky Mote Type #sky1",
        'source' : "[CONTIKI_DIR]-2.7/examples/ipv6/simple-udp-rpl/unicast-receiver.c",
        'commands' : "make unicast-receiver.sky TARGET=sky",
        'firmware' : "[CONTIKI_DIR]-2.7/examples/ipv6/simple-udp-rpl/unicast-receiver.sky"
    }
senderMote = {
        'identifier' : "sky2",
        'description' : "Sky Mote Type #sky2",
        'source' : "[CONTIKI_DIR]-2.7/examples/ipv6/simple-udp-rpl/unicast-sender.c",
        'commands' : "make unicast-sender.sky TARGET=sky",
        'firmware' : "[CONTIKI_DIR]-2.7/examples/ipv6/simple-udp-rpl/unicast-sender.sky"
    }
```

6.3. Реализация протоколов в COOJA симуляторе

Реализацияпротокола RPL. ContikiRPL это реализация протокола RPL с открытым исходным кодом, RPL представляет собой стандарт протоколов маршрутизации IPv6 для сети с потерями и низким энергопотреблением

ContikiRPL работает на уровне над уровнем беспроводной линии с низким энергопотреблением и потерями. Протокол RPL был успешно запушен с радио настенде беспроводной сети. Реализация полностью интегрирована с операционной системой Contiki, и включена по умолчанию в платформу TmoteSky.

Требуемый обхем памяти менее, чем 5 Кбайт ПЗУ и 0,5 Кбайт оперативной памяти. ContikiRPL реализует протокол RPL, согласно версии 18 из спецификации RPL, и два объективной функций - OF0 и минимального ранга целевой функции с гистерезисом Minimum Rank Objective Function with Hysteresis (MRHOF). ContikiRPL был успешно протестирован на совместимость с помощью программы-взаимодействия IPSOAlliance, где он был использован на трех различных платформах и запушен на двух разных соединенных уровнях, IEEE 802.15.4 и Watteco коммуникационный модуль питания линии с низким энергопотреблением.

ContikiRPL также была использована в двух сетевых датчиках развертывания. Реализация ContikiRPL отделяет логику протокола, конструкция и разбор сообщения и объективные функции в различных модулях. Модуль логики протокола управляет информацией DODAG, поддерживает набор кандидатов родителей, и связывающую их информацию, общается с модулями объективных функций и проверяет сообщения RPL на логическом уровне в соответствии со спецификацией RPL. Конструкция сообщения и модуль синтаксического анализа преобразуют форматы сообщения RPLICMPv6 в собственные абстрактные структуры данных ContikiRPL. Наконец, модули целевой функции реализуют целевую функцию API.

При экспериментировании с целевыми функциями, их внутренние операция недоступны для ContikiRPL. ContikiRPL производит переадресации решения в пакете, но устанавливает переадресацию (tables) для uIPv6 и оставляет фактическое перенаправление пакетов на uIPv6. Исходящие пакеты IPv6 направляются от уровня uIPv6 к уровню 6LoWPAN для сжатия заголовков и фрагментации. Уровень 6LoWPAN пересылает пакеты MAC уровню.

По умолчанию уровень ContikiMAC является механизмом CSMA/CA, который ставит исходящие пакеты в очередь. Пакеты из очереди передаются в порядке поступления через уровень radiodutycycling (RDC). Уровень RDC, в свою очередь передает пакеты через уровень радиолинии.

Уровень MAC отправляет пакеты, пока не получит подтверждения канального уровеня от приемника. Если происходит коллизия, MAC уровень делает откат и повторно передает пакет. Исходящие пакеты имеют конфигурируемый порог для максимального числа передач.

Модуль информации о внешних соседях обеспечивает обновление оценок стоимости пути с помощью функции обратного вызова. В течение одной секунды обновления, ContikiRPL пересчитывает стоимость пути к шлюзу через обновленную линию, и проверяет выбранную целевую функцию, следует ли сделать переключение. Стоимость путиописывается метрикой ETX, которая рассчитывается с помощью экспоненциально-взвешенного скользящего среднего функции по числу канального уровня передач с a = 0,2. Устройство ETX используется для вычисления ранга целевой функции MRHOF и выбора материнской целевой функции OF0. Записи о новых соседяхимеют первоначальную оценку ETX равную 5. Плитика выбора соседей ContikiRPL основана на удержании соседей, которые имеют хорошие оценки ETX и низкие ранги.

Стек протокола RIME. Стек протокола RIME включает набор коммуникационных примитивов, начиная от широковещательных лучшим локальным соседям и однонаправленной доставкой лучшему локальному соседу и заканчивая передачей всем узлам в сети с наилучшими показателями и надежной однонаправленной пошаговой передачей. Приложения и протоколы,

работающие выше стека RIME, могут использовать 1 или несколько примитивов, предоставляемых данным стеком.

Протоколы в стеке RIME имеют уровневую организацию, где более сложные протоколы реализуются, используя более простые. Мы выбрали коммуникационные примитивы в Rime стеке, основываясь на том, какие типовые протоколы сенсорных сетей используются. Приложения и протоколы, работающие поверх стека RIME соединяются с любым уровнем стека и используют любые коммуникационные примитивы. Стек протоколов RIME поддерживает примитивы с передачей на соседний узел(одношаговые) и с последовательной передачей через несколько узлов(многошаговые). Многошаговые примитивы не определяют способ маршрутизации в сети. Вместо этого, в процессе передачи пакета через сеть приложение или протокол верхнего уровня вызывается на каждом узле, и определяет следующий узел. Это позволяет реализовать произвольные протоколы маршрутизации, используя многошаговые примитивы.

Для того чтобы показать (подчеркнуть) универсальность коммуникационных примитивов, мы реализуем 4 типа стандартных протоколов для сенсорных сетей поверх стека RIME: 1 протокол сбора данных, такой как MintRoute и CTP, 2 протокола распространения данных на основе Trickle и Deluge и 1 протокол маршрутизации для меш-сетей, основанный на AOVD.

Протокол маршрутизации для mesh-сетей, основанный на AOVD. Каждый узел хранит список адресов узлов назначения вместе с адресом следующего узла. Чтобы установить путь через сеть, протокол посылает всем ближайшим соседям пакеты “Запрос маршрута”. Когда один из узлов получает такой пакет, он устанавливает обратный путь к отправителю данного пакета и передает этот же пакет дальше к ближайшим соседям. Если один из узлов обнаружил, что “Запрос маршрута” адресован ему, то он посылает отправителю пакет “Ответ на запрос маршрута”. Узлы на пути следования уже знают обратный путь к отправителю начального запроса. “Ответ на запрос маршрута” посылается используя многошаговую однонаправленную передачу. Узлы, которые передают “Ответ на запрос маршрута”, устанавливают маршрут к получателю, который изначально посылал “Запрос маршрута”. Пакеты данных посылаются, используя многошаговую однонаправленную передачу.

RIME реализация протокола маршрутизации для меш сети использует примитив передачи всем узлам в сети (распространение по сети) (nf) и примитив многошаговой передачи с наилучшими результатами (mh) для многошаговой и однонаправленной передачи.

6.3.1. Размещение узлов

Для исследования и сравнения протоколов маршрутизацииRPLи AODVв БСС использовали типовые плоскостные (2D) модели.

Во всех сценариях шлюз располагался в центре сенсорного поля, измерялось количество промежуточных узлов для оценки задержек и потерь пакетов. Сенсорное поле представляет собой квадрат, ширина которого увеличивалась от 10 до 100 метров по мере увеличения количества узлов.

На рис.40 приведен пример для одного из сценариев, где размещены 100 узлов случайным образом на поле с размером 100 на 100, шлюз размещен в центре поля.

Для передачи информации используются широко распространенный в самоорганизующихся сетях протокол AODV и протокол сети с потерями и низким энергопотреблением RPL .

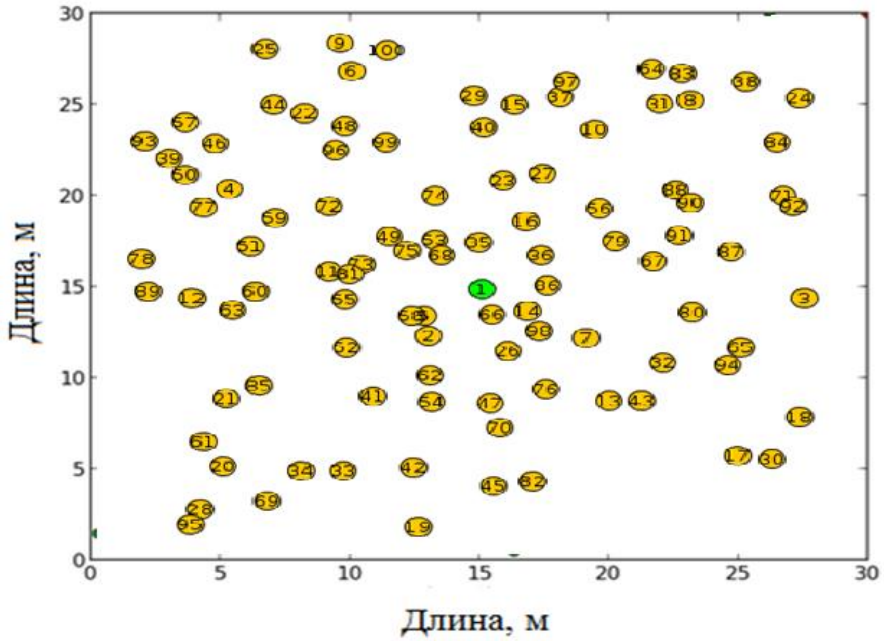


Рисунок 2 –Сенсорное поле

6.3.2. Результаты моделирования для сравнения протоколов AODV и RPL для разных плотностей сенсорных узлов

Интересно отметить, что в то время как RPL построил только двунаправленные пути, некоторые пути в AODVсети являются однонаправленными из-за стохастического характера процесса распространения.Точнее, большинство узлов добавляют маршрут к шлюзу при

первомдостижении шлюза RREQ. Тем не менее, обратный путь отыскивается в более позднее время, когда приемник должен адресовать узел. Обратим внимание на то, что маршруты RPL короче, чем у AODV. Одновременно отметим, что AODV предоставляет наиболее оптимальные маршруты в случае коротких маршрутов. Есть две основные причины для этого: 1) из-за операции распространения канал насыщен и некоторые сообщения RREQ отбрасываются, потому что они достигают максимального числа попыток протокола CSMA, 2) у сенсорных узлов ограничена память, они имеют возможность хранить только два пакета, поэтому часть сообщенийRREQ также может быть отброшена. При моделировании протокола AODV было замечено, что существенное число пакетов может пройти более, чем четыре шага. С другой стороны, как видно на рисунке 4.6 маршруты, построенные с RPL оптимальны: ни один пакет не затрачивает более 4 шагов.

Результаты моделирования для двух моделей для протокола RPL показаны на рис. 41 и 42, первая модель содержит 100 узлов, размещенных случайным образом на поле 100 на 100 метров, шлюз размешается в центре поля, а вторая модель содержит 50 узловпри тех же параметрах поля.

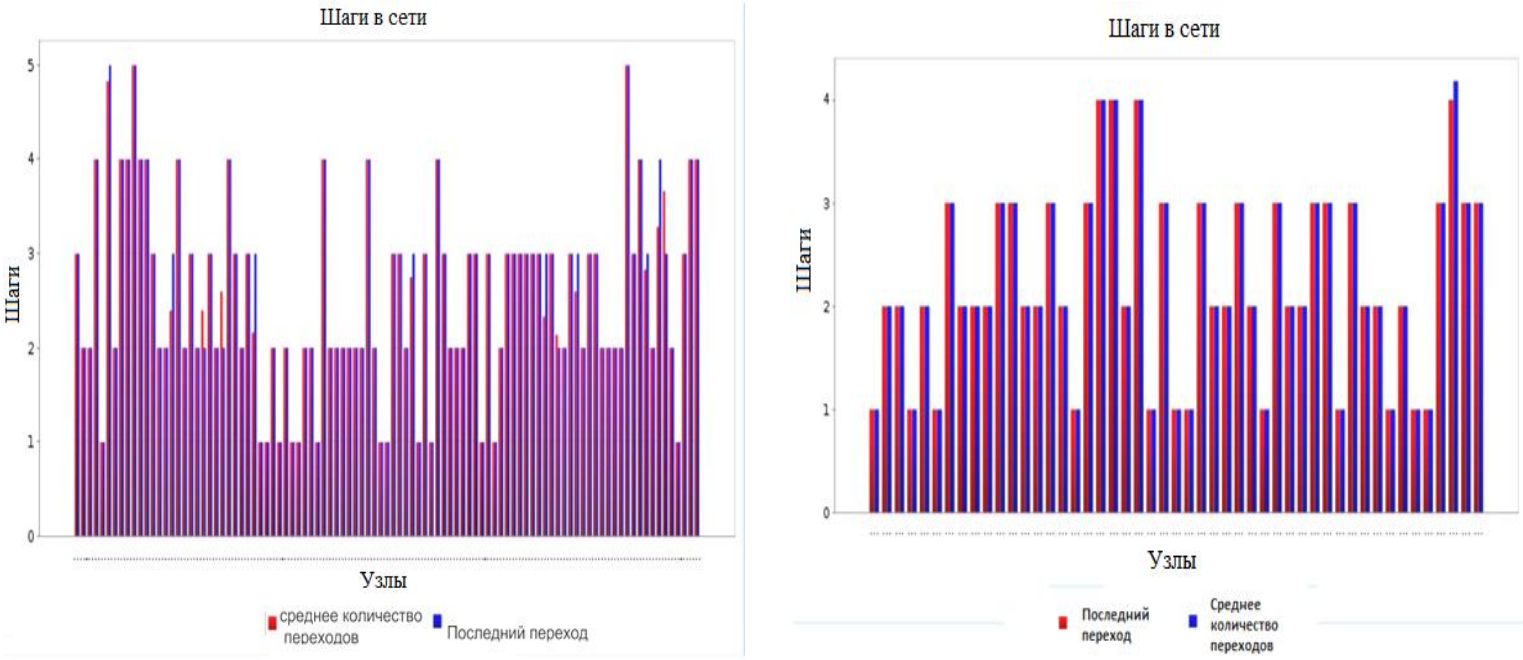


Рисунок – Зависимость числа шагов от числа узлов в сети для 100 и для 50 узлов

Реализация протокола CoAP в Cooja симуляторе

Для реализации протокола авторы использовали инструменты разработки поставляемые с ОС Contiki[7,9]. Данная операционная система поддерживает современные протоколы связи для сетей с потерями и низким энергопотреблением: 6LoWPAN, RPL, CoAP. Моделирование производилось с помощью инструмента COOJA. Для построения сети на базе CoAP

использовалась реализация протокола из ОС Contiki.

На рисунке 2 показано сенсорное поле представляет собой квадрат с шириной 100 метров, модель содержит 22 узлов, размещенных случайным образом на поле 100 на 100 метров.

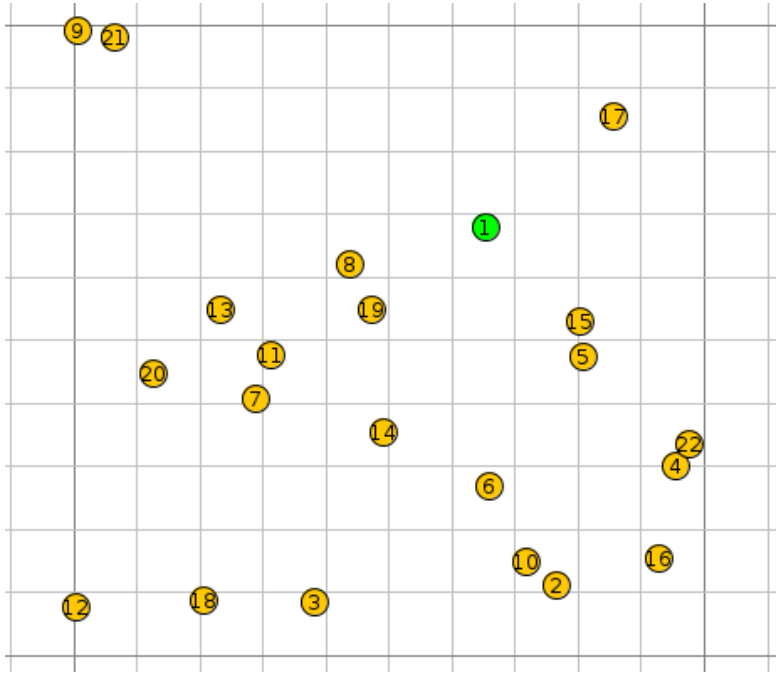


Рис.. Сенсорное поле

Для реализации работы coap сервера (er-rest) на cooja нужно маршрутизировать запрос\ответ пакеты между ОС contiki ubuntu и cooja. В качестве такого маршрутизатора использовали как и внешние платформы так и localhost. Для этого надо задать интерфейс 127.0.0.1 узлу на котором

скомпилирован er-rest (server coap) и потом объявить маршрутизацию между сенсорным полем и ОС. Для данной симуляции авторы использовали сеть с протоколом RPL. В этой директории contiki/examples/ipv6/rpl-border-router запускаем маршрутизацию make connect-router-cooja. после этого было получено маршрутизированный сетевой адрес er-rest узла рисунок 3.

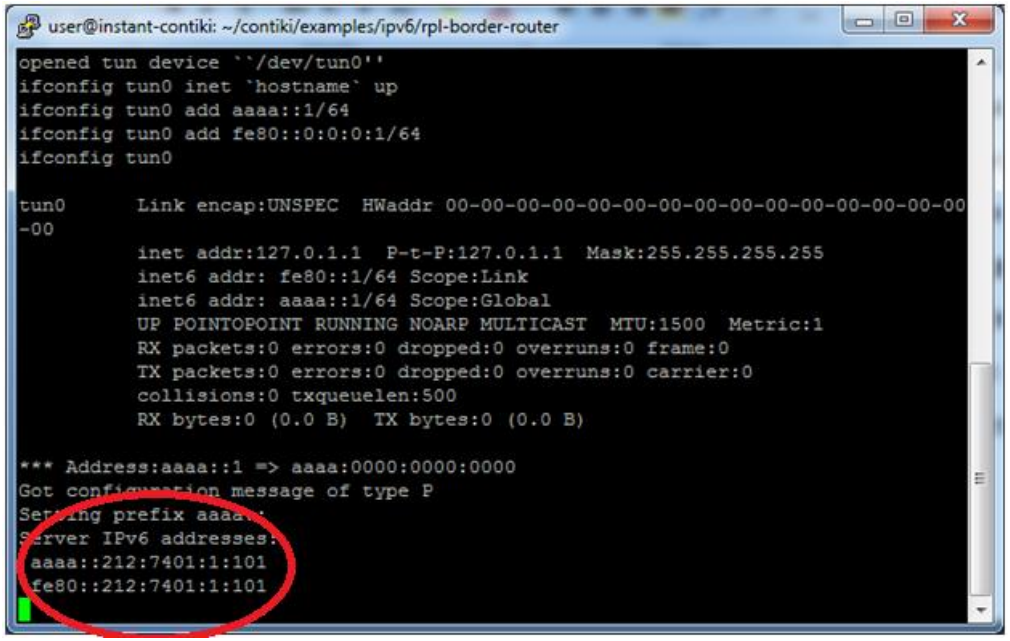


Рис. маршрутизированный сетевой адрес er-rest узла

Протокол CoAP в WoT. Для реализации протокола в качестве клиента, в браузерах используют дополнительный инструмент Copper mozila firefox (CU). CoAP клиент (CU) позволяет

взаимодействовать с сервером CoAP через встроенный WEB-ресурс рис. 4. Для подключения надо ввести в адресной строке URL адрес удалённого узла [IPv6]:5683. В открывшейся страничке можно увидеть методы GET,PUT, POST, и DELETE. Он регистрирует обработчик протокола для схем COAP URI (UniformResourceIdentifier) унифицированный идентификатор ресурса, которые легко объединяет URI в браузере. Пользователи могут просматривать устройства, закладки свои ресурсы, как обычные веб-страницы, и по ссылкам в документах HTML, для обнаружения новых устройств.

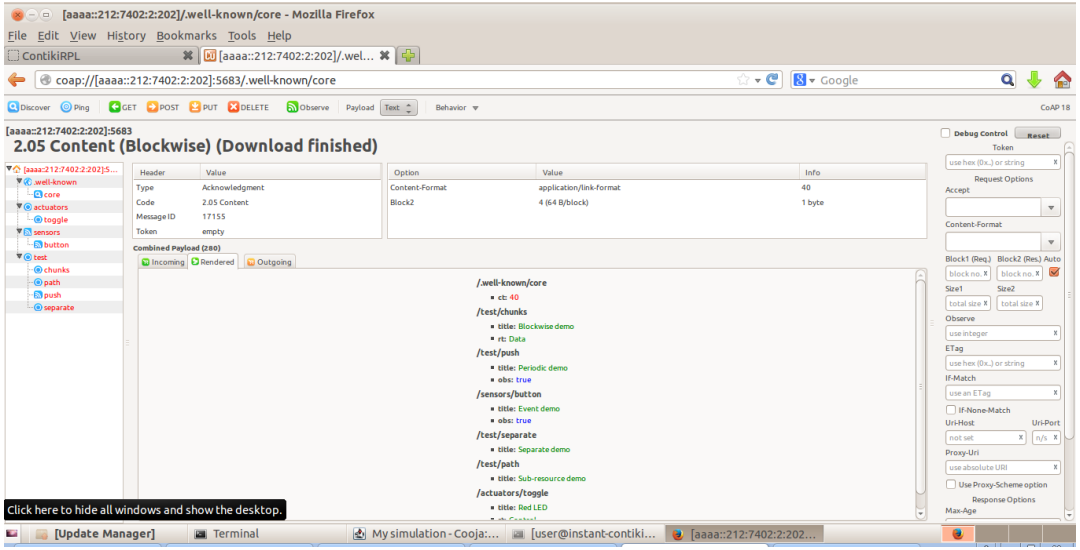


Рис.. Протокол CoAP в WoT

По сравнению с другими протоколами, CoAP работает поверх UDP(транспортного уровня).

Основным преимуществом протокола CoAP- является то, что он работает поверх UDP, что уменьшает объём передаваемого пакета и не требует дополнительной синхронизации, что очень хорошо для сетей с низким энергопотреблением. Также это даёт возможность мониторинга удалённого объекта в реальном времени, где невозможно обеспечивать узел постоянной энергией.

ТАБЛИЦА. Сравнение протоколов WoT

Протокол	Транспортный Уровень	QOS	Архитектура	безопасность
CoAP	UDP	ДА	Запрос\ответ	DTLS
MQTT	TCP	ДА	издатель\подписчик	TLS/SSL
XMPP	TCP	нет	издатель\подписчик Запрос\ответ	TLS/SSL
AMQP	TCP	да	издатель\подписчик	TLS/SSL
Web socket	TCP	Нет	Клиент\сервер издатель\подписчик	TLS/SSL