Математические структуры и моделирование
2023. № 3 (67). С. 71–87

УДК 004.7
DOI 10.24147/2222-8772.2023.3.71-87

ГЕТЕРОГЕННЫЕ СЕТИ С ЯЧЕИСТОЙ ТОПОЛОГИЕЙ: ТЕХНОЛОГИИ, МОДЕЛИРОВАНИЕ, ПРОТОТИПИРОВАНИЕ СЕТЕВЫХ УСТРОЙСТВ

С.В. Гусс
старший преподаватель, e-mail: sviat@v-guss.ru

Омский государственный университет им. Ф.М. Достоевского, Омск, Россия

Аннотация. В статье рассматривается проблема выбора сетевых технологий, инструментов моделирования и программируемых технических средств реализации гетерогенных, беспроводных, одноранговых компьютерных сетей для инфраструктуры интернета вещей. Разбираются особенности экспериментальной программно-аппаратной разработки сетевых устройств. Даются рекомендации касательно выбора сетевых технологий для организации беспроводных сетей ячеистой топологии, по использованию средств моделирования и подбору аппаратной базы для реализации проектов сетевых устройств.

Ключевые слова: гетерогенные сети, ячеистые сети, интернет вещей, IEEE 802.11s, IEEE 802.15.4.

1. Введение

Значительный интерес в сфере развития компьютерных сетей сегодня направлен в сторону децентрализованных, одноранговых, самоорганизующихся, или как их ещё иногда называют, самовосстанавливающихся (ad-hoc), гетерогенных (HetNet), главным образом беспроводных компьютерных сетей пакетной передачи данных. Такие сети рассчитаны на организацию надёжной связи вычислительных мобильных устройств различного назначения.

Сфера использования таких сетей не ограничивается гражданскими сценариями внедрения технологий интернета вещей и может расширяться на производственные и специализированные области.

Доминирующей топологией здесь является ячеистая сеть (mesh) [1]. Технологии реализации могут быть различными. Из популярных решений доступны Wi-Fi Mesh, который базируется на стандарте IEEE 802.11s, а также Bluetooth Mesh, ZigBee и Thread, базой для которых является стандарт IEEE 802.15.4. Выбор конкретного решения зависит от прикладных потребностей, т. е. того, для каких целей мы организуем сеть ячеистой топологии: для соединения беспроводных датчиков (умный дом, заводские объекты, городская инфраструктура), для поддержания связи мобильных устройств пользователей (в походах, экспедициях, поисковых отрядах, военных целях).

Если говорить об устройствах или объектах инфраструктуры интернета вещей, то здесь можно встретить различные операционные системы и протоколы, с кото-

рыми приходится иметь дело в процессе сетевой коммуникации [2]. Необходимо провести исследование, изучить возможности поведения таких разнородных сетей на различных сценариях в контролируемой и воспроизводимой среде, прежде чем переходить к прототипированию устройств и/или организации обеспечивающей сетевой инфраструктуры для той или иной прикладной области в тех или иных сценариях.

Главные преимущества ячеистой топологии:

- Дальность связи увеличивается за счёт того, что каждый участник сети может пересылать данные своим соседним устройствам, используя последние как промежуточные сетевые устройства.

- Нагрузка на обработку трафика конечных устройств снижается, поскольку каждое устройство в сети может брать на себя роль коммутатора/маршрутизатора.

- Самовосстановление. Нет центрального узла, от которого зависит жизнеспособность всей сети. Есть возможность пересылать информацию по множеству альтернативных маршрутов.

Это потенциальные преимущества. Есть и недостатки, главный из которых – сложность управления такими сетями и их непредсказуемость. Вдобавок, каждая конкретная сетевая технология накладывает свои сложности и ограничения. И всё это выливается в ещё большую неопределённость, справиться с которой, хотя бы частично, можно введением сценариев и профилей использования [3].

2. Технологии реализации беспроводной ячеистой сети

Традиционная компьютерная сеть образуется следующими элементами:

- **Оконечные узлы** (хосты) – это устройства пользователей (смартфоны, портативные трекеры, умные браслеты) или специализированные устройства (интеллектуальные датчики, дроны наблюдения), объединённые общей компьютерной сетью, имеющие сквозное соединение и доступные для взаимного обмена данными.

- **Промежуточные устройства** – это сетевые устройства, основное предназначение которых заключается в пересылке и возможной обработке трафика (фильтрация, маршрутизация).

- **Сетевые среды** (каналы) – это физическая основа передачи данных от одного узла или устройства к другому. В простейшем случае выделяют проводные/кабельные среды (коаксиальный кабель, витая пара, оптоволокно) и беспроводные/мобильные технологии (радиосоединение, сотовые сети, спутниковые коммуникации и т. д.).

Сетевая технология, как некоторый набор стандартов, правил и протоколов, определяет состав необходимых программно-аппаратных средств (оконечных и промежуточных устройств), правила и границы их взаимодействия, задаёт допустимую топологию, релевантную сетевой среде.

Говоря о беспроводных сетях с ячеистой топологией, стоит заметить, что основные объекты таких сетей – мобильные устройства с автономным питанием. Пользовательское устройство (или, если речь идёт о промышленном применении, умный датчик, например) одновременно может стать передающим сетевым, промежуточным для других узлов сети. Это, однако, не исключает наличие дополнительных устройств, таких как базовые станции, предусматриваемые некоторыми сетевыми технологиями. Базовые станции (шлюзы) в таком контексте являются, прежде всего, граничными устройствами для предоставления доступа в Интернет, а пользовательские устройства в таком случае можно называть абонентскими терминалами.

Если речь идёт об интернете вещей, то необходимо ещё наличие сетевого сервера и сервера приложений для разворачивания полноценной инфраструктуры. Но всё это определяется уже конкретной сферой использования.

Технологии для беспроводных компьютерных сетей можно разделить по трём следующим категориям: LPWAN, WLAN и WPAN/WBAN.

Технологии категории **LPWAN** (Low-Power Wide-Area Network – энергоэффективная сеть дальнего радиуса действия) хорошо подходят для передачи на дальние расстояния небольшого объёма данных. Принцип работы близок сетям мобильной связи. Используется топология «звезда», в которой пользовательские устройства подключаются к базовым станциям. Можно разделить на две группы: сотовые сети с лицензируемым диапазоном частот (LTE-M, NB-IoT) и несотовые с нелицензируемым диапазоном частот ISM (LoRaWAN, SigFox, «Стриж»). Абонентские устройства, подключённые к базовым станциям LPWAN, могут служить граничными шлюзами, обеспечивающими доступ в Интернет.

Для передачи больших объёмов данных, но на небольшие расстояния (в пределах нескольких метров) лучше подойдут сети **WLAN** (Wireless Local Area Network – беспроводная локальная сеть). В этой категории очевидный лидер Wi-Fi. Есть ограничения по количеству подключений. Wi-Fi версии 6 уже более приспособлен к интернету вещей. В стандарте IEEE 802.11s Wi-Fi Mesh есть описание того, как организовать сеть с ячеистой топологией. Это наиболее предпочтительный вариант для реализации сети мобильных пользовательских устройств (смартфоны, планшеты, ноутбуки).

Для передачи небольших объёмов данных на близкие расстояния выгоднее, с точки зрения потребления электроэнергии, использовать сети **WPAN** (Wireless Personal Area Network – беспроводная персональная сеть) и **WBAN** (Wireless Body Area Network – беспроводная нательная компьютерная сеть). Bluetooth 5-й версии развивает концепцию Mesh, т. е. можно организовать сеть с полноценной ячеистой топологией. Другая технология ZigBee (и похожая на неё технология Z-Wave) тоже отлично подходит для реализации ячеистой сети. ZigBee не используется в пользовательских устройствах, за редким исключением. ZigBee более характерен для промышленных сетей. Ещё одной перспективной технологией для сетей с ячеистой топологией является 6LoWPAN в связке с протоколом Thread. Технологии данной

категории можно использовать для передачи координат местонахождения, служебной информации небольшого объёма, данные о состоянии объектов или среды.

Используя ту или иную технологию беспроводной связи, необходимо держать в голове многоуровневую модель/архитектуру организации взаимодействия, чтобы точно понимать, задача какого уровня подлежит реализации. В качестве универсальной схемы рекомендуется использовать четырёхуровневую сервис-ориентированную архитектуру (SOA) для интернета вещей [4]:

- **Интерфейсный уровень**. Приложения, пользователи, взаимодействие: соглашения, интерфейсы, API.
- **Сервисный уровень**. Разделение и интеграция сервисов. Репозиторий сервисов. Бизнес-логика.
- **Сетевой уровень**. Сетевая поддержка передачи данных: технологии, стандарты, протоколы.
- **Уровень зондирования**. Аппаратная поддержка. Физическая среда. Технологии, протоколы.

Две наиболее доступные технологии для организации сетей ячеистой топологии – это Wi-Fi Mesh и Bluetooth Mesh. У них разный принцип действия. Wi-Fi Mesh может оказаться затратным решением по потреблению электроэнергии, в сравнении с Bluetooth Mesh, который работает поверх BLE (Bluetooth Low Energy – Bluetooth с низким энергопотреблением).

Wi-Fi Mesh изначально планировался для устранения зон со слабым сигналом, цель – обеспечение непрерывного покрытия в большом помещении и на территории какого-нибудь предприятия. В отличии от Bluetooth Mesh, в Wi-Fi Mesh поддерживается адаптивная маршрутизация в целях самовосстановления и поддержания стабильности сети.

Wi-Fi Mesh больше подходит для пользователей мобильных устройств, чтобы постоянно оставаться на связи друг с другом в пределах mesh-сети. Bluetooth Mesh больше приспособлен для организации связи между автономными устройствами или датчиками, например в беспроводных сенсорных сетях (WSN – Wireless Sensor Network).

Bluetooth Mesh

Стандарт BLE появился в 2010 г. Сегодня решения на базе BLE широко используются в устройствах интернета вещей (умных датчиках, носимых устройствах). В чистом BLE нет возможности организовать сеть с ячеистой топологией, в которой устройства могут напрямую слать друг другу сообщения и в случае необходимости – пересылать данные на другие устройства в сети. В 2017 г. эта проблема была решена появлением стандарта Bluetooth Mesh как надстройки над BLE.

Рассмотрим далее, из чего состоит mesh-сеть Bluetooth.

Узлы (nodes). *Зарегистрированные устройства* (provisioned devices) – участники сети, способные принимать и передавать сообщения. *Незарегистрированные*

устройства (unprovisioned devices) не являются частью mesh-сети. За присоединение к сети отвечают специальные устройства, которые называются *регистраторами* (provisioners). Они отвечают за аутентификацию устройств, назначение адресов и выдачу ключей шифрования. *Клиент конфигурации* (Configuration Client) – часть регистратора или другого любого узла сети. Он выполняет дальнейшую подготовку узла к присоединению в сеть. Выдаёт дополнительные сетевые ключи, настраивает адреса подписки и публикации для каждой модели. У каждого узла должен быть *элемент* (element), их может быть несколько, они могут находиться в различных *состояниях* (условиях). Элементы имеют адреса и созданы для содержания *моделей* (models). Модель определяет базовый функционал узла.

Модели. Сервер (Server), клиент (client), управление (control).

Адреса. Однонаправленные (unicast), групповые (group), виртуальные (virtual). Однонаправленные адреса выдаются элементам в порядке возрастания.

Сообщения. Обмен информацией происходит посредством передачи сообщений (messages) методом публикации и подписки (publish/subscribe). Сообщения определяются специальным кодом операции (GET, SET, STATUS), параметрами и поведением. Сообщения также оперируют состояниями, представляющими положение элемента.

Типы узлов: а) ретранслятор (relay node) – пересылает сообщения по сети, уменьшает значение TTL пакета; б) узел с низким энергопотреблением (LPN – Low Power Node) – может находиться в режиме сна, получает накопленные данные от дружественного узла (friend node), который специально хранит данные для LPN; в) посредник (proxy) – работает с обычными BLE устройствами через GATT; г) узел регистрации (provisioner node) – регистрирует узлы и раздаёт ключи безопасности.

Bluetooth Mesh не является маршрутизируемой сетью (не предусматривается технологией, но можно реализовать при необходимости на прикладном уровне). Для передачи данных внутри сети использует управляемые потоки (managed flooding). Все входящие пакеты кэшируются, чтобы избежать повторной пересылки.

Как уже было сказано, Bluetooth Mesh в своей основе имеет BLE. Работает на частоте 2.4 ГГц. Частотный диапазон разделён на 40 каналов по 2 МГц каждый. Передача сообщений происходит без установки соединения. Процесс обеспечения сети: рассылка маячков, приглашение, обмен открытыми ключами, аутентификация, распространение данных обеспечения.

Достоинства BLE:

- Низкое энергопотребление (оптимизированный протокол передачи, режим сна).
- Бесплатная документация (официальная).
- Доступность (недорогая цена электроники, модулей и чипов).
- Распространённость (смартфоны, планшеты, ноутбуки, персональные устройства).

Ограничения BLE:

- Низкая пропускная способность. Не подходит для передачи больших объёмов данных.
- Небольшая дальность передачи. На передачу влияют заграждения, стены, люди. Наличие корпуса устройства (внутренняя антенна).
- Необходимость шлюза для связи с Интернет.

3. Этапы жизненного цикла экспериментальной программно-аппаратной разработки

Существует классическая модель, пришедшая из сферы разработки программного обеспечения, основанная на каскадном принципе, когда последовательно идут этапы, начиная от анализа и заканчивая внедрением готового продукта. В более современных, гибких моделях, суть этапов не особо изменилась, а добавились некоторые элементы организации процесса, позволяющие возвращаться к предыдущим (вплоть до начального) этапам и вести разработку циклами по инкрементному принципу.

В смешанной разработке, когда разрабатывается программно-аппаратное решение, гибкость желательна. А когда речь идёт об экспериментальной разработке, то она просто необходима. Здесь есть потребность постоянно вносить изменения и коррективы в разрабатываемый продукт. Именно поэтому в последнее время появляется всё больше программируемых электронных модулей для прототипирования и высокоуровневых языков для программирования этих самых модулей, избавляющих разработчиков от рутинных задач и погружения в детали реализации той или иной архитектуры микропроцессорной системы. От выбора соответствующей аппаратной и программной базы зависит скорость создания прототипа или даже минимально жизнеспособного продукта.

Далее будет представлена общая схема разработки программно-аппаратного решения.

Анализ предметной области

Составляющие:

- Описание реальной проблемы в конкретной предметной области и необходимых средств для устранения проблем.
- Обоснование актуальности (в том числе и технико-экономический анализ и обоснование рациональности принятия тех или иных проектных решений или выбора средств разработки: аппаратной базы, программного обеспечения).
- Сравнение доступных и возможных решений, анализ альтернатив.

- Выбор компромиссного решения между имеющимися базовыми и новыми экспериментальными, доступными для тестирования и апробации технологиями.

- Анализ рисков. Учёт альтернативных подходов к решению проблемы и возможных обходных путей, физических препятствий, экономических или технологических ограничений.

Выбор инструментов моделирования и разработки

Частично этот вопрос рассматривается в статье далее. Единственное, что хотелось бы добавить ещё, это случай, когда готовых инструментов не существует. Здесь есть два варианта. Первый – расширить существующий инструмент для поддержки того, что нам надо. Второй – разработать специализированное средство, не содержащее лишних элементов.

Концептуализация

- Подготовка базы для имитационного моделирования сценариев использования технологий для решения задач в предметной области.

- Выдвижение гипотез.

- Планирование общей картины взаимодействия объектов (определение ролей и сценариев взаимодействия, выделение профилей под конкретную ситуацию).

Проведение имитационного моделирования

- Проверка гипотез.

- Выделение паттернов.

- Отбрасывание или отправка на доработку или повторный анализ неработающих сценариев.

- Сбор и обобщение статистических данных и создание на этой основе общей картины взаимодействия устройств.

Выбор базовых сценариев

Поскольку в реальности невозможно проверить абсолютно все сценарии, а для проверки жизнеспособности разрабатываемого решения этого и не требуется, то достаточно выделить ключевые сценарии, определяющие минимум того, что делает систему полезной.

Прототипирование или подбор готовых устройств

После того как было проведено моделирование и получено подтверждение выдвинутых гипотез, нужны ещё физические эксперименты. Чтобы к ним подготовиться, необходимо соорудить устройство или целую сеть устройств, а в нашем случае (гетерогенных сетей с ячеистой топологией) – поддерживающую инфраструктуру сетевого взаимодействия.

Проведение физического эксперимента

В идеале необходимо провести полевые испытания в реальной физической среде со всевозможными ограничениями и возможными непредвиденными обстоятельствами в рамках запланированных сценариев использования.

4. Инструменты имитационного моделирования

4.1. Сетевой симулятор NS-3

NS-3 – симулятор сети с дискретными событиями (для написания сценариев, описания моделей используются языки программирования C++ и Python).

Ключевые абстракции:

- **Узлы** (класс Node). Узел – более общее обозначение хоста или конечной системы, вычислительного устройства. В дальнейшем к узлу добавляется функциональность, стек протоколов, периферийные устройства и приложения. Узлы взаимодействуют друг с другом через каналы посредством сетевых устройств.

- **Каналы** (класс Channel). Среда передачи данных, канал, абстракция для управления взаимодействием объектов, подключения узлов, могут быть специализированы для моделирования как элементарных вещей, типа обычного кабеля, так и сложных устройств, типа сетевого коммутатора Ethernet и даже пространства с препятствиями для моделирования беспроводной коммуникации.

- **Сетевые устройства** (класс NetDevice). Абстракция включает в себя оборудование и необходимые драйверы. Сетевое устройство устанавливается в узел. Один узел может подключаться к нескольким каналам через несколько сетевых устройств. В простейшем случае можно рассматривать как сетевую карту.

- **Приложения** (класс Application). Можно использовать готовые специализиации класса, наборы клиент-серверных приложений для генерации сетевых пакетов.

- **Помощники** (классы с суффиксом Helper или Container: NodeContainer, PointToPointHelper, NetDeviceContainer, InternetStackHelper, Ipv4InterfaceContainer, MeshHelper, MobilityHelper). Для установки и базовой

настройки сетевой технологии (канала, сетевого устройства): настройка физических параметров и ограничений, подключение узлов к сети, назначение адресов, объединение сетей и т. д. Помощники предоставляют такие методы, как, например, Install для установки сетевого устройства в узел или привязывания модели передвижения в пространстве, а также SetDeviceAttribute и SetChannelAttribute для установки необходимых физических значений типа DataRate или Delay. Для Mobility можно установить параметры Mode (Time, Distance), Speed, Bounds, т. е. режим (время или расстояние), скорость, границы.

В NS-3 нет абстракции для операционной системы. В реальном мире на устройствах тоже может отсутствовать операционная система, а функционал реализован на «голом железе». Даже если и есть операционная система, то это может быть не универсальная система типа Windows или Linux, а, например, операционная система реального времени (RTOS). В NS-3 на этот счёт не делается никаких предположений.

Трассировка

Для наблюдения за результатами моделирования используется трассировка, которая запускается через конкретный объект Helper вызовом общего метода EnableTraces, который, в свою очередь, за кадром, согласно моделируемой сетевой технологии, запускает ряд соответствующих трассировок для конкретных протоколов, а те – по конкретным параметрам, которые, при желании, можно запускать отдельно, если нужна только конкретная статистика. Тем самым запущенная через скрипт модель по окончании работы оставляет запись событий, которые можно анализировать. Запись ведётся во множество файлов, название которых начинается, например, на Ul (upload link, передача данных от абонента к базовой станции) и Dl (down link, передача данных от базовой станции к абоненту), а заканчивается на Stats (т. е. статистика). В названии файлов также есть сокращения Rx (приём), Tx (передача) и различные сокращения для характеристик взаимодействия устройств по протоколам физического и канального уровня моделируемой сетевой технологии (Phy, Mac, Rsrp, Rlc, Pdcp).

Содержимое файлов представлено строками таблицы с различными названиями полей, соответствующих особенностям моделируемой среды. Например, для технологии LTE это может быть в случае с файлом для фиксации физического состояния при передаче UlTxPhyStats.txt (для DlTxPhyStats.txt всё будет аналогично): time (время), cellId (идентификатор базовой станции), IMSI (глобальный идентификатор абонента), RNTI (идентификатор абонента в пределах базовой станции), layer (слой), size (размер трансферного блока), ccId (идентификатор несущей). А, например, в файле DlRsrpSinrStats.txt появляются поля для параметров sinr (отношение сигнал/шум с учётом интерференции сигнала) и ComponentCarrierId. Наличие тех или иных полей определяется моделируемой средой. Строки в более специфических файлах, таких как, например, UlInterferenceStats.txt, могут содержать всего три поля: time, cellId и Interference, – описывающие интерференцию сигнала. А в файле DlRlcStats.txt можно найти довольно детальную информацию, типа количества

переданных и полученных блоков данных (nTxPDUs, nRxPDUs), переданных и полученных байтов (TxBytes, RxBytes), задержку в канале (delay) и т. д.

Приведённые выше описания касаются трассировок низкого, физического уровня. Существуют также трассировки более высокого, сетевого уровня, создающие записи формата PCAP, которые можно анализировать в Wireshark. Можно включить полную трассировку методом EnablePcapAll() и задать название файла.

Часть полезной информации можно выводить на экран через LogComponentEnable.

Симуляция

Управление планированием событий моделирования происходит через класс Simulator.

Отдельные аспекты планирования, например истечение срока определённого действия и реакцию на это, можно задать через метод Simulator::Schedule.

С вызова метода Simulator::Run начинается симуляция. Происходит обработка событий в порядке их возникновения. Некоторые события могут стать причиной генерации новых событий. Симуляция автоматически закончится, если очередь событий окажется пустой.

Принудительное завершение симуляции можно выполнить вызовом метода Simulator::Stop с указанием конкретного времени останова. Для компьютерных сетей характерно существование повторяющихся событий, например рассылка объявлений и другой служебной информации протоколов, обновлений таблиц маршрутизации, проверка целостности маршрутов в заданной области и т. д.

Очистка или освобождение ресурсов выполняется вызовом Simulator::Destroy().

Заготовка скрипта модели беспроводной ячеистой сети

В NS-3 для моделирования сетей с ячеистой топологии предлагается использовать технологию Wi-Fi, готовых модулей для Bluetooth нет.

Пример скрипта моделирования входит в официальную сборку и лежит в директории src/mesh/examples. Можно использовать его в качестве отправной точки. Полный исходный текст рассматривать не будем, а разберём основные элементы, которые можно задействовать в своих моделях.

Этот скрипт создаёт топологию квадратной сетки `m_xSize * m_ySize` со стеком IEEE 802.11s, установленным на каждом узле с управлением пирингом и протоколом HWMP. Сторона квадратной ячейки определяется параметром `m_step`. Дополнительные параметры можно задавать через MeshTest::Configure.

Параметры и их значения, заданные по умолчанию:

- **x-size** (`int m_xSize (3)`) – количество узлов в каждой строке сетки;
- **y-size** (`int m_ySize (3)`) – количество узлов в каждом столбце сетки;
- **step** (`double m_step (100.0)`) – размер ребра сетки;

- **start** (`double m_randomStart (0.1)`) – максимальная задержка случайного запуска в секундах;
- **time** (`double m_totalTime (100.0)`) – время моделирования в секундах;
- **packet-interval** (`double m_packetInterval (0.1)`) – интервал между пакетами при UDP-пинге в секундах;
- **packet-size** (`uint16_t m_packetSize (1024)`) – размер пакетов в UDP-пинге;
- **interfaces** (`uint32_t m_nIfaces (1)`) – количество радиоинтерфейсов, используемых каждой точкой сетки;
- **channels** (`bool m_chan (true)`) – использовать разные частотные каналы для разных интерфейсов;
- **pcap** (`bool m_pcap (false)`) – включить трассировку PCAP на интерфейсах;
- **stack** (`std::string m_stack ("ns3::Dot11sStack")`) – тип стека протоколов;
- **root** (`std::string m_root ("ff:ff:ff:ff:ff:ff")`) – Mac-адрес корневой точки сетки в HWMP.

Не приводя лишних деталей и параметров, скрипт моделирования состоит из следующих шагов:

- CreateNodes
- InstallInternetStack
- InstallApplication
- Simulator::Schedule
- Simulator::Stop
- Simulator::Run
- Simulator::Destroy

CreateNodes состоит из следующих шагов:

- Создаются станции в количестве `m_ySize*m_xSize` для формирования топологии сетки: `nodes.Create (m_ySize*m_xSize);`
- Настраивается канал YansWifiChannel:

```
YansWifiPhyHelper wifiPhy = YansWifiPhyHelper::Default ();
YansWifiChannelHelper wifiChannel =
YansWifiChannelHelper::Default ();
wifiPhy.SetChannel (wifiChannel.Create ());
```

- Создаётся вспомогательная сетка и настраивается установщик стека. Установщик стека инициализирует все необходимые протоколы и устанавливает их на устройство точки сетки:

```
mesh.SetStackInstaller(m_stack, ...)
mesh.SetSpreadInterfaceChannels(...)
mesh.SetMacType(...)
```

- Устанавливается количество интерфейсов:
`mesh.SetNumberOfInterfaces (m_nIfaces);`

- Устанавливаются протоколы на узлы:
`meshDevices = mesh.Install (wifiPhy, nodes);`

- Настраивается тип мобильности:

```
MobilityHelper mobility;
mobility.SetPositionAllocator (...)
mobility.SetMobilityModel (...);
mobility.Install (nodes);
```

Суть двух остальных шагов состоит в следующем: InstallInternetStack – настройка параметров IP узлов, InstallApplication – организация небольшого клиент-серверного взаимодействия.

Значение последних четырёх методов (начинающихся с префикса Simulator::) было описано выше. Это типичные для скрипта моделирования методы расписания (Schedule), останова (Stop), запуска (Run) и освобождения ресурсов (Destroy).

Анимация

Имеет смысл, если в проекте есть перемещения. Не самая сильная сторона NS-3. Для анимации используются такие средства, как netanim или PyVis (src/visualizer).

4.2. Программное обеспечение Anylogic

Anylogic – программное обеспечение для имитационного моделирования сложных систем и процессов (для разработки моделей можно использовать язык программирования Java).

Имеет графический язык моделирования. Доступны диаграммы потоков и процессов, карты состояний, блок-схемы и др.

Имеет библиотеку моделей, например Process Modeling Library (библиотека моделирования процессов). Позволяет отследить общую динамику процессов и выявить взаимосвязи между отдельными компонентами. Можно создавать собственные библиотеки.

Можно использовать для моделирования работы телекоммуникационных сетей связи.

4.3. Среда NetLogo

NetLogo – среда многоагентного моделирования и агентно-ориентированный язык программирования. Несмотря на то, что данное средство известно и используется в основном в сфере образования, применимо и для профессионального моделирования. Хорошо справляется с задачами, где есть большое количество независимых объектов.

Можно определить своих агентов, разместить нужным или случайным образом и задать направление. Согласно терминологии NetLogo есть следующие типы агентов: черепашка (turtle), связь (link), пятно (patch), наблюдатель. Для агентов turtle и link можно определить породу (breed). Агенты одного типа объединяются в набор. Синтаксис несложен, порог вхождения низкий.

Имеется встроенная анимация.

В NetLogo есть библиотека моделей. Конкретно для компьютерных сетей готовых моделей нет. Но есть модели общего плана:

- **Network** (сеть). Есть 2D и 3D версии. Здесь можно собрать базовую статистику о сети. Можно задать свои правила соединения узлов, кроме абсолютно случайных. Можно: 1) соединить каждый узел с каждым другим узлом; 2) убедиться, что у каждого узла есть хотя бы одна входящая или исходящая ссылка; 3) соединить только узлы, которые пространственно близки друг к другу; 4) сделать некоторые узлы «концентраторами» (с большим количеством ссылок); 5) создать два типа узлов, отличающихся по цвету, а затем разрешить ссылкам соединять только два узла разных цветов – это делает сеть «двусторонней» (можно расположить два вида узлов по двум прямым линиям).

- **Fully Connected Network** (полностью подключенная сеть). В этой заготовке показано, как создать полносвязную сеть, т. е. сеть, в которой каждый узел связан со всеми другими узлами. Можно понаблюдать, как растёт количество ссылок по мере увеличения количества узлов.

- **Small Worlds** (модель сети малых миров, адаптация модели Уоттса–Строгаца). Сеть малого мира – это математический граф, в котором большинство узлов не являются соседями друг друга, но соседи любого заданного узла, вероятно, будут соседями друг друга. Благодаря этому до большинства соседних узлов можно добраться из любого другого узла за небольшое количество переходов или шагов. В частности, сеть малого мира определяется как сеть, в которой типичное расстояние L между двумя случайно выбранными узлами (количество необходимых шагов) растёт пропорционально логарифму числа узлов N в сети.

- **Random Network** (случайная сеть). Показывает, как создать два разных типа случайных сетей. В сети Эрдёша–Реньи каждой возможной ссылке даётся фиксированная вероятность создания. В простой случайной сети между случайными узлами создаётся фиксированное количество связей.

- **Team Assembly** (сборная команда). Эта модель сетей сотрудничества иллюстрирует, как поведение отдельных лиц при сборе небольших команд для краткосрочных проектов может со временем привести к возникновению разнообразных крупномасштабных сетевых структур.

- **Giant Component** (гигантский компонент). В сети «компонент» – это группа узлов (людей), которые прямо или косвенно связаны друг с другом. Итак, если в сети есть «гигантский компонент», это означает, что почти каждый узел доступен почти из любого другого. Эта модель показывает, как быстро возникает гигантский компонент, если вы наращиваете случайную сеть.

- **Preferential Attachment** (предпочтительное приложение). В некоторых сетях несколько «концентраторов» имеют много соединений, в то время как все остальные имеют только несколько. Эта модель показывает один из способов возникновения таких сетей. Такие сети можно найти в широком диапазоне ситуаций реального мира, начиная от связей между веб-сайтами и заканчивая сотрудничеством между участниками. Эта модель создаёт такие сети с помощью процесса «предпочтительного подключения», при котором новые члены сети предпочитают устанавливать соединение с более популярными существующими членами.

Рассмотренные заготовки носят скорее фундаментальный характер. Могут стать базой для планирования конкретных сценариев взаимодействия устройств. А это, в конечном счёте способно отразиться в настраиваемом профиле.

5. Программируемые технические средства для прототипирования сетевых устройств

Как уже отмечалось в самом начале, устройства могут быть следующих типов: пользовательские, сетевые, универсальные. В контексте гетерогенных, беспроводных сетей под пользовательским устройством может скрываться всё что угодно, от смартфона или планшета до чипа под кожей. Самые простые устройства могут работать без участия операционной системы, под управлением программы, работающей на «чистом железе». Более универсальные устройства, такие как смартфон, могут использоваться как данность, а разработка здесь может ограничиваться созданием прикладного программного обеспечения для конкретной операционной системы. Если говорить о специализированных устройствах, то, скорее всего, здесь речь уже будет идти об использовании программируемых логических интегральных схем (FPGA), которые лучше всего, в смысле скорости, справляются с обработкой цифровых сигналов. Для критических, по реакции на внешние события, систем не обойтись без использования операционной системы реального времени.

Для прототипирования и экспериментов существуют готовые комплекты, наборы, одноплатные компьютеры и оценочные платы для распространённых архитектур, типа ARM. Существует множество решений для мобильных платформ под операционную систему Android.

Для быстрого прототипирования беспроводных устройств предпочтение отдаётся системам типа SoC (System on Chip – система на кристале), например чипы семейства nRF52/53. SoC – это уже не просто микроконтроллер или микропроцессор, это система с интегрированными периферийными модулями Bluetooth и/или Wi-Fi. Также можно использовать одноплатные компьютеры с установленными чипами поддержки беспроводной связи, как это сделано в Raspberry Pi. В качестве доступных по цене и весьма популярных решений можно также рассматривать платы типа ESP-32 и Raspberry Pi Pico W, имеющие поддержку Bluetooth и Wi-Fi.

Наиболее популярные решения для Bluetooth Mesh – это платы на базе STM32WB и nRF52/nRF53 (например, модуль Silicon Witchery S1 с SoC nRF52840 Bluetooth и FPGA Lattice iCE40, предназначенный для обработки цифровых сигналов для граничных вычислений и работы с алгоритмами машинного обучения).

Если цель – создать доступное, популярное решение, необходимо выбирать платформу, которую выбирает массовый пользователь, смартфон на базе современной операционной системы. Если же цель – создать специализированное устройство, то необходимо выбирать более низкоуровневое решение на базе специализированных чипов, с учётом ряда факторов, таких как надёжность поставщика чипов, доступность в будущем, наличие аналогов (взаимозаменяемость) и т. д.

По поводу выбора языка программирования, то, если речь идёт о программировании аппаратной части, выбор следующий (от наиболее низкоуровневого решения к высокоуровневому): Assembler, C, C++, MicroPython, TinyGo. Assembler применяется редко, в основном для анализа поведения и отладки сложных моментов. C/C++ более универсален, и это, по сути, основной язык, если речь идёт об операционных системах реального времени, таких как FreeRTOS, Zephyr, Mbed OS, Keil RTX и т. д. Что касается MicroPython и TinyGo, то эти языки могут быть полезны для проверки работоспособности каких-то простых взаимодействий, для считывания данных с датчиков и работы с индикаторами, дисплеями и т. д.

Если речь идёт о разработке под FPGA, то тут выбор языков небольшой: VHDL, Verilog и SystemVerilog. Всё это универсальные языки описания аппаратуры. В отличие от языков программирования, они манипулируют поведением цифровых логических схем, что позволяет спроектировать программным способом аппаратное решение.

Касательно разработки программного обеспечения. Если это системное программное обеспечение, реализация драйвера или протокола, то это C/C++. Если прикладное, то выбор огромен, начиная от того же C/C++ и заканчивая современными языками, типа Kotlin, Python и Go.

6. Заключение

Основные выводы, которые можно сделать касательно технологий организации гетерогенных беспроводных сетей ячеистой топологии:

- Это сети, которые могут объединять огромное количество разнообразных вычислительных устройств, взаимодействующих по разным протоколам.

- Наиболее универсальные и доступные технологии для организации таких сетей – это Bluetooth Mesh и Wi-Fi Mesh. Bluetooth Mesh более приспособлена для организации сети находящихся неподалёку друг от друга устройств (умный дом, пользовательские мультимедиаустройства, интеллектуальные датчики) и характеризуется пониженным потреблением электроэнергии за счёт того, что работает на базе BLE. Wi-Fi Mesh можно использовать для организации больших сетей предприятия и для полевого развёртывания.

Что касается процесса моделирования сетевого взаимодействия устройств в рамках ячеистой топологии:

- Для проверки общей идеи на высоком уровне доступны такие средства моделирования, как AnyLogic и NetLogo.
- Для проверки особенностей работы конкретных сетевых технологий и трассировки деталей взаимодействия устройств по различным сетевым протоколам доступно мощное средство NS-3, которое можно расширять реализациями своих сетевых технологий в случае необходимости.

И, что касается последнего вопроса, связанного с выбором аппаратной базы, то на рынке доступен широкий диапазон аппаратных средств для прототипирования:

- Популярные экспериментальные платы, типа ESP-32, Raspberry Pi Pico W для проектов небольших устройств.
- Устройства, работающие на базе универсальных операционных систем, такие как Raspberry Pi 2/3/4 (доступны различные дистрибутивы Linux), Onion Omega2+ и другие системы на базе Linux-дистрибутивов типа OpenWrt и LEDE, обычные пользовательские смартфоны на базе Android.
- Если необходимо поработать с технологическими тонкостями и добраться до самых деталей, то здесь доступны такие системы, как STM32WB и nRF52/nRF53 (модуль Silicon Witchery S1 с интегрированным SoC nRF52840 Bluetooth и FPGA Lattice iCE40).

Литература

1. Гусс С.В. Самоорганизующиеся mesh-сети для частного использования // Математические структуры и моделирование. 2016. № 4(40). С. 102–115.
2. Гусс С.В. Имитационное моделирование беспроводных самоорганизующихся сетей с быстроменяющейся топологией // Математическое и компьютерное моделирование : сборник материалов IV Международной научной конференции. Омск, 2016. С. 111–112.
3. Гусс С.В. Реализация протокола многопутевой маршрутизации в самоорганизующихся динамических сетях // Омские научные чтения : материалы Всероссийской научно-практической конференции. Омск, 2017. С. 316–318.
4. Сюй Л.Д., Хе В., Ли С. «Интернет вещей» в промышленности: обзор ключевых технологий и трендов. URL: `https://controleng.ru/internet-veshhej/klyuchevy-h-tehnologij/` (дата обращения: 03.08.2023)

HETEROGENEOUS NETWORKS WITH MESHED TOPOLOGY: TECHNOLOGIES, MODELING, PROTOTYPING OF NETWORK DEVICES

S.V. Guss
Assistant Professor, e-mail: sviat@v-guss.ru

Dostoevsky Omsk State University, Omsk, Russia

Abstract. The article deals with the problem of choosing technologies, modeling tools, software and hardware for the implementation of heterogeneous, wireless, peer-to-peer computer networks for the infrastructure of the Internet of things. The features of experimental software and hardware development of network devices are analyzed. Recommendations are given regarding the choice of network technologies for organizing wireless networks of mesh topology, the use of modeling tools and the selection of a hardware base for the implementation of projects of network devices.

Keywords: heterogeneous networks, mesh networks, IoT, IEEE 802.11s, IEEE 802.15.4.

Дата поступления в редакцию: 14.05.2023