Новые возможности технологии Ember для сетей ZigBee

Татьяна КРИВЧЕНКО, к. т. н.
tkr@efo.ru

Разработчики беспроводных средств связи малого радиуса действия уже знакомы с продукцией компании Ember [1]. Цель настоящей статьи обсудить новые возможности встраиваемых библиотек EmberZNet, рассмотреть технологию программирования и основные характеристики новых приемопередатчиков ZigBee EM250 и EM260, а также познакомить читателей с новыми отладочными средствами Ember, которые позволяют не только вести отладку программного обеспечения отдельного узла беспроводной сети, но и настраивать приложение ZigBee на сетевом уровне.

Поддержка ячеистых беспроводных сетей

Компания Ember для своих приемопередатчиков предоставляет встраиваемую библиотеку EmberZNet, которая реализует все уровни стека протоколов ZigBee, а также обладает рядом дополнительных преимуществ.

Одним из таких преимуществ является то, что пакет EmberZNet позволяет строить как сети с топологией «дерево», так и ячеистые сети, которые являются более эффективными, надежными и могут содержать большее количество узлов [1].

Спецификация ZigBee 1.0 анонсирует возможность построения сетей с обеими топологиями, но конкретный механизм реализации сети прописан только для единственного стандартного профиля стека HC (Home Control) и соответствующего профиля приложения HCL (Home Control Lighting), которые предписывают использовать топологию «дерево».

По информации Ember в течение 2006 года рабочие группы альянса ZigBee продолжали работу над совершенствованием спецификации Zigbee, и, в частности, велась разработка нового профиля стека CII (Commercial, Industrial, Institutional), который будет предназначен для создания ячеистых сетей. Для профиля стека CII разрабатываются сразу несколько профилей приложения. Ожидается, что появятся такие новые профили приложения, как Commercial Building Automation (CBA), Industrial Plant Monitoring (IPM), Wireless Sensor Applications (WSA) и другие.

Компания Ember уже сегодня заложила в свои библиотеки поддержку ячеистых сетей.

Поддержка мобильных и батарейных узлов

Несмотря на то, что уже много сказано [1, 2] о том, какие типы узлов могут присутствовать в сетях ZigBee, хочется еще раз обсудить эту тему, учитывая возможности, предоставляемые пакетом EmberZNet.

Координатор несет ответственность за образование сети. В сети с топологией «дерево» координатор и после образования сети играет ключевую роль, так как хранит в своей памяти топологию всего «дерева» и, следовательно, все маршруты для передачи сообщений в сети. В ячеистой сети после ее образования координатор играет роль обычного маршрутизатора, если профиль приложения не накладывает на него какие-либо специфические функции. Следует, правда, отметить, что в случае реализации защиты данных в сети с использованием удостоверяющего центра (trust center) таким центром может быть только координатор.

Маршрутизаторы обеспечивают ретрансляцию сообщений в сети, а также могут посылать и принимать свои собственные сообщения. В отличие от конечных узлов маршрутизаторы никогда не могут «засыпать», так как не известно заранее, в какой момент потребуется выполнять ретрансляцию.

Конечные узлы могут взаимодействовать с сетью только через свои «родительские» узлы и не могут ретранслировать сообщения других узлов. В зависимости от настроек конечные узлы, выполненные на базе технологии Ember, могут быть следующих типов: «засыпающие» (sleepy), «незасыпающие» (non-sleepy) и «мобильные» (mobile).

Все конечные узлы не имеют возможности ретранслировать сообщения. «Засыпающие» узлы также обладают тем свойством, что переходят в режим пониженного энергопотребления и выключают свой приемопередатчик в периоды между сеансами связи.

Поддержка «мобильных» узлов в библиотеках Ember появилась лишь в этом году. «Мобильные» узлы, так же как и «засыпающие», имеют возможность отключать свой приемопередатчик, но, кроме этого, они еще могут перемещаться в пределах сети и постоянно менять «родительские» узлы.

Присоединяясь к сети, конечное устройство сообщает «родительскому» узлу свой тип, в зависимости от которого «родительский» узел по-разному обслуживает конечный узел. Обслуживание «засыпающих» и «мобильных» узлов очень похоже. Когда в сети передается

сообщение на такой узел, то это сообщение сохраняется в буфере «родительского» узла и считывается затем конечным узлом во время очередного его пробуждения. «Засыпающие» и «мобильные» узлы, сообщая «родительскому» узлу свой тип, также сообщают интервал времени, в течение которого они хотя бы один раз обратятся к «родительскому» узлу для считывания данных. Если «засыпающий» узел не обратится к «родительскому» узлу в течение этого интервала времени, то «родительский» узел все равно сохраняет для него буфер в своей памяти, но уже не гарантирует, что узел получит все адресованные ему сообщения без потери информации. Если же «мобильный» узел не выйдет на связь вовремя, то «родительский» узел считает его ушедшим из радиуса действия и вычеркивает из своих таблиц, а «мобильный» узел в свою очередь, не получая ответа от прежнего «родительского» узла, начинает поиск нового, что он делает в среднем за 600 мс.

Хочется еще раз отметить, что маршрутизаторы ZigBee-сети все время должны слушать эфир и, следовательно, обязаны иметь стационарное питание. Это не должно разочаровывать читателей, которые, например, задумали оснастить беспроводными батарейными устройствами офис или коттедж. В зависимости от толщины и материала стен, а также размеров здания на каждом этаже потребуется разместить всего два–три маршрутизатора со стационарным питанием, а все остальные узлы (датчики, пульты управления, исполнительные устройства) могут быть батарейными. Кроме того, очень часто исполнительные устройства для осуществления своей основной функции должны иметь стационарное питание (вентиляторы, обогреватели, осветительные лампы). Если в сети имеются такие устройства, то целесообразно именно их и выбрать в качестве маршрутизаторов.

Механизм агрегатирования

Ранее [1] уже обсуждались вопросы формирования сети ZigBee, поиска маршрутов и передачи сообщений. Механизм агрегатирования является сравнительно новым и более эффективным механизмом маршрутизации, дополнительно введенным в библиотеки EmberZNet для сетей сбора данных, в которых множество удаленных узлов пересылают сообщения на один центральный узел.

Напомним, что, говоря о маршрутизации, мы имеем в виду только ячеистые сети, поскольку в ZigBee-сети с топологией «дерево» маршруты жестко заданы структурой самого «дерева», хранятся в памяти координатора, и, следовательно, маршрутизация в таких сетях не требуется.

В ячеистой сети каждый узел прежде, чем он первый раз передаст сообщение на какой-нибудь адрес, должен при помощи соответствующей API-функции сделать запрос сетевому уровню библиотеки EmberZNet на поиск оптимального маршрута между двумя узлами. Стандартный алгоритм поиска маршрута между узлами A и B осуществляется следующим образом. Узел A рассылает широковещательное сообщение вида: «Я — A! A кто знает B?». Все узлы, которые слышат это сообщение, ретранслируют его, добавляя информацию о том, с каким уровнем сигнала они получили сообщение. Таким образом, узел B получает сообщение от A много раз различными путями и имеет возможность выбрать оптимальный маршрут. При этом минимизируется количество ретрансляций и учитывается качество сигнала при каждой пересылке. Ответное сообщение узел B посылает уже по выбранному оптимальному маршруту. Каждый промежуточный узел при этом сохраняет в своей таблице маршрутизации соответствующую запись о том, что он является участником данного маршрута. Из сказанного выше видно, что процедура поиска маршрута, представляющая собой широковещательную рассылку, сильно загружает сеть. Для сетей сбора данных, в которых множество удаленных узлов пересылают сообщения на один центральный узел, механизм агрегатирования позволяет заменить множество широковещательных рассылок от удаленных узлов, предназначенных для поиска маршрута к центральному узлу, на всего одну широковещательную рассылку, инициируемую самим центральным узлом. В результате такой широковещательной рассылки, вызываемой API-фунцией emberCreateAggregationRoutes() , каждый удаленный узел получает маршрут к центральному узлу (рис. 1).

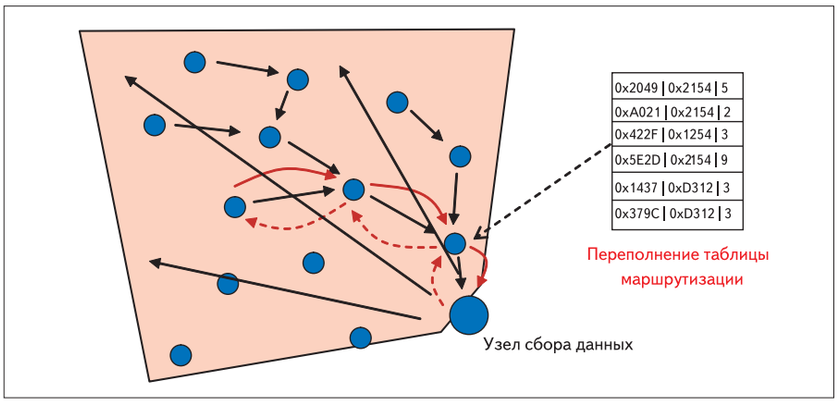

Рис. 1. Агрегатирование — новый механизм маршрутизации для систем сбора данных

Таблица. API-функции библиотеки EmberZNet для обмена сообщениями

	уровень APS	Транспортный уровнь Ember
Адресные сообщения	emberSendUnicast()	emberSendDatagram() emberSendSequenced()
Широковещательные сообщения	emberSendBroadcast()	emberSendMulticast() emberSendLimitedMulticast()

Еще одной проблемой сетей сбора данных является то, что узлы, близко расположенные к центральному и участвующие в большом количестве маршрутов (рис. 1), расходуют много памяти на хранение записей в своих таблицах маршрутизации. Механизм агрегатирования позволяет сократить объем требуемой памяти за счет того, что сохраняются только записи для маршрутов в направлении от удаленного узла к центральному. Предполагается, что в системах сбора данных сообщения от центрального узла к удаленным осуществляются реже и для них создаются только временные (всего на одно сообщение) маршруты в тот момент, когда сообщение от удаленного узла направляется к центральному узлу. Таким образом, центральный узел для того, чтобы послать свое сообщение, должен сначала дождаться сообщения от удаленного узла, принимая которое, центральный узел получает также временный маршрут для передачи одного своего сообщения в обратном направлении. Центральный узел может использовать этот временный маршрут вместе с подтверждением послать какую-либо дополнительную информацию. После того, как ответное сообщение будет доставлено удаленному узлу, промежуточные узлы удаляют из своей памяти записи о временном маршруте.

Доставка сообщений

Библиотека EmberZNet позволяет приложению обмениваться сообщениями, непосредственно взаимодействуя с уровнем поддержки приложений APS [1, 4] стека протоколов ZigBee, или дает возможность использовать дополнительный транспортный уровень Ember [1], который является надстройкой над уровнем APS и обеспечивает дополнительный сервис.

В таблице представлены API-функции библиотеки Ember для обмена адресными и широковещательными сообщениями. Видно, что уровень APS предоставляет программисту не такой уж большой выбор. При выполнении адресной передачи в этом случае программист должен в качестве параметра функции указать 16-разрядный адрес узла в сети. Для получения этого адреса приложение, зная 64-разрядный идентификатор узла, должно предварительно послать запрос стеку и получить соответствующий 16-разрядный адрес. Широковещательная рассылка с использованием уровня APS не позволяет ввести ограничения на количество адресатов и поэтому очень сильно загружает сеть.

Транспортный уровень Ember берет на себя заботу об отслеживании соответствия между 64-битными идентификаторами и 16-разрядными адресами в сети, и поэтому программист, использующий транспортный уровень Ember, может не задумываться о 16-разрядных адресах. При пересылке адресных сообщений транспортный уровень Ember предоставляет программисту две возможности. Если требуется отсылать одиночные пакеты, то удобно использовать функцию emberSendDatagram(), которая представляет собой аналог UDP-сообщений в сетях Ethernet только с той разницей,

что обеспечивается подтверждением доставки сообщения. Функцию emberSendSequenced() следует использовать в том случае, когда требуется послать большое количество пакетов и важно, чтобы на приемной стороне сохранился порядок поступления пакетов. Для выполнения этой задачи функция emberSendSequenced() добавляет к сообщению номер пакета. Эта функция подобна обмену TCP в сетях Ethernet и требует предварительной установки соединения между двумя адресатами и закрытия соединения после того, как обмен будет завершен.

Транспортный уровень Ember также предоставляет более развитые функции и для широковещательных рассылок. В частности, при помощи функций emberSendMulticast() и emberSendLimitedMulticast() можно выполнить групповую рассылку, то есть рассылку сообщений группе узлов без ограничения или с ограничением на количество ретрансляций соответственно.

Для использования транспортного уровня Ember программист должен предварительно создать таблицу связей (bindings), в которой сохраняются записи для всех одиночных адресатов и групповых адресатов, с которыми данный узел предполагает общаться.

Процесс реальной отправки сообщения в эфир (так же, как и обработка входящих сообщений) зависит от используемого кристалла.

При выполнении приложения и стека на одном кристалле (приемопередатчики EM2420 или EM250) программный код приложения и стек разделяют между собой процессорное время. В программе это организуется таким способом, что в основном цикле программы как можно чаще вызывается функция emberTick(), которая собственно и выполняет все задачи, которые должен выполнить стек к настоящему моменту на всех уровнях. Таким образом, после того, как в основном цикле будет выполнена, например, функция emberSendDatagramm(), соответствующее сообщение будет поставлено в очередь на отправку, но реально оно будет отправлено в эфир тогда, когда очередной раз получит управление функция emberTick().

В случае приема сообщения приемопередатчиком это сообщение попадает в очередь на обслуживание входных сообщений, и при очередном вызове функции emberTick() стек инициирует вызов callback-функции emberIncomingMessageHandler() для того, чтобы сообщить приложению о поступлении нового сообщени.

При использовании сетевого сопроцессора EM260 стек и приложение выполняются на разных процессорах, поэтому сопроцессор сразу начинает передачу сообщения в эфир после поступления от хоста транзакции emberSendDatagram() по SPI-интерфейсу. При приеме нового сообщения сопроцессором он генерирует сигнал прерывания, который сообщает хосту, что необходимо инициировать прием данных из сопроцессора EM260.

Приемопередатчики Ember

Компания Ember сегодня выпускает три различных кристалла для реализации устройств ZigBee: EM2420, EM250, EM260. Они различаются между собой по функциональному назначению, составу внутренних узлов и характеристикам собственно приемопередатчика ZigBee. Цветовая диаграмма на рис. 2 поясняет функциональное назначение приемопередатчиков Ember. Напомним, что обмен данными в беспроводных сетях ZigBee регламентируется многоуровневым стеком протоколов, который изображен на рис. 2 в правой части.

Исторически первый кристалл EM2420 является наиболее простым. Он не содержит процессорного ядра и реализует на аппаратном уровне собственно приемопередатчик и часть уровня доступа к среде (MAC). Для реализации протоколов более высоких уровней ZigBee Ember предлагает библиотеку EmberZNet, разработанную в среде IAR для AVR-микроконтроллеров компании Atmel. Библиотека предоставляется в объектных кодах и компилируется совместно с кодом приложения. Пакет EmberZNet для AVR также содержит исходные коды демонстрационных проектов, один из которых реализует ячеистую сеть сбора данных с датчиков, а другой является примером реализации сети со стандартным профилем ZigBee HCL для управления осветительным оборудованием.

Кристалл EM250 предназначен для устройств, требующих минимизации габаритных размеров и энергопотребления беспроводного узла. Этот кристалл содержит одновременно и приемопередатчик ZigBee и 16-разрядный микроконтроллер XAP2b, встроенную Flash-память, аналого-цифровой преобразователь с разрешением 12 разрядов, таймеры, порты ввода/вывода. Таким образом, на базе кристалла EM250 можно строить компактные однокристальные устройства ZigBee. Отличительной особенностью кристалла EM250 является его малое энергопотребление в спящем режиме, которое составляет менее 1 мкА при работающем встроенном таймере, предназначенном для выведения кристалла из спящего режима.

Преимуществом EM250 по сравнению со своим предшественником также являются увеличенные максимальная мощность передачи (3,2 мВт по сравнению с 1 мВт у EM2420) и чувствительность приемника (-98 дБм по сравнению с -94 дБм у EM2420), что позволило увеличить дальность связи между узлами. Так, по сообщению компании Ember, на открытом пространстве с использованием керамических антенн была достигнута дальность связи 250 м.

Новый кристалл EM260 разработчики Ember называют сетевым сопроцессором. Он также содержит на кристалле процессорное ядро XAP2b, но это ядро предназначено только для обслуживания стека ZigBee, в то время как основное приложение должно выполняться на внешнем хост-процессоре. Таким образом, при использовании кристалла EM260 мы снова возвращаемся к двухкристальному решению. Однако теперь имеется множество преимуществ. Во-первых, в качестве хост-процессора можно использовать любой микроконтроллер; во-вторых, не требуется совместная компиляция кода приложения и кода, реализующего стек, что позволяет избежать множества трудно выявляемых ошибок. В-третьих, два процессора работают быстрее, чем один. Если при использовании кристаллов EM2420 и EM250 код приложения и код стека разделяли между собой процессорное время, то теперь выполнение приложения и обслуживание стека могут выполняться параллельно, что, конечно, будет приводить к повышению скорости отправки и приема сообщений.

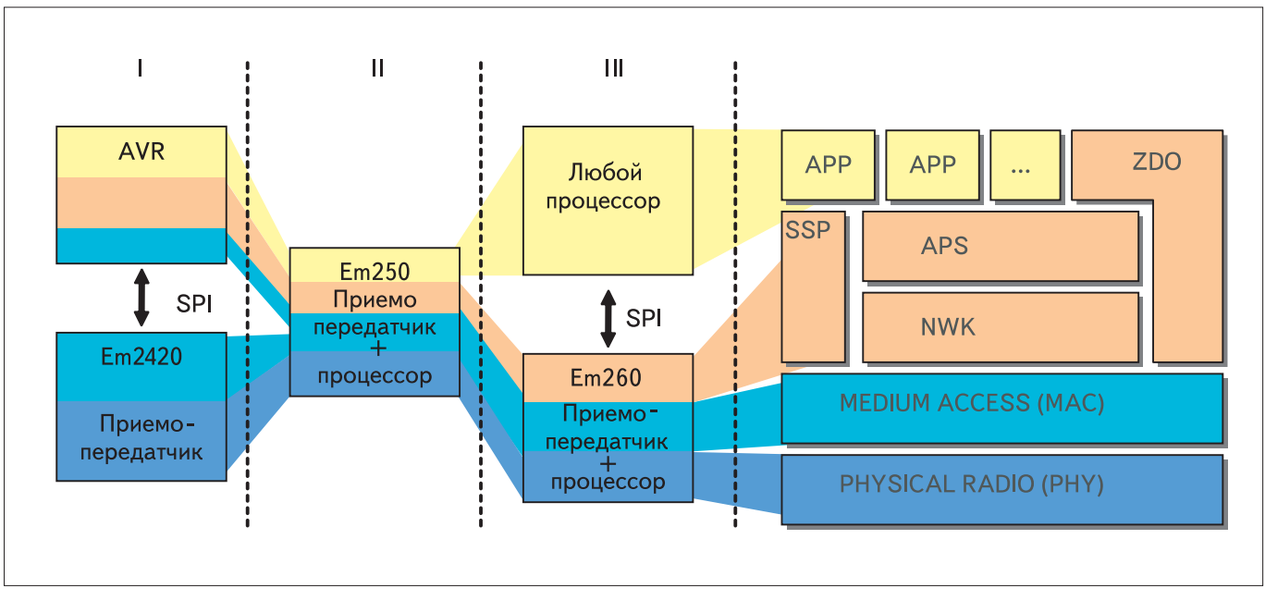

Рис. 2. Функциональное назначение приемопередатчиков Ember для сетей ZigBee

Следует отметить, что для всех трех типов кристаллов используется одна и та же библиотека EmberZNet. В первом случае эта библиотека портирована для микроконтроллеров AVR и среды разработки IAR, во втором — для кристалла XAP2 и среды разработки "xIDE for EM250" и, наконец, в третьем, откомпилированный программный код, реализующий стек Zigbee, непосредственно «прошивается» в кристалл EM260. В первых двух случаях, когда приложение и стек обслуживаются одним процессором, приложение взаимодействует со стеком при помощи API-функций и callback-функций [1].

В случае применения сопроцессора вместо API-функций используются транзакции интерфейса SPI (рис. 3). Каждая транзакция состоит из трех фаз: команды, которую приложение посылает стеку; фазы ожидания, во время которой сопроцессор обрабатывает команду; фазы ответа сопроцессора. Таким образом, транзакция SPI подобна вызову API-функции, которая так же сначала передает некоторое сообщение стеку, затем ждет, когда стек отработает, и только потом получает ответ о результате выполненной операции.

Кроме интерфейса SPI сопроцессор EM260 связан с хостом еще двумя сигнальными линиями — HOST_INT и WAKE. Первая из них выполняет роль callback-функции. Кристалл EM260 формирует на ней сигнал прерывания для хоста, когда имеет какие-либо данные для него, например, в случае поступления очередного сообщения из сети. Вторая линия, наоборот, предназначена для того, чтобы хост мог вывести кристалл EM260 из режима пониженного энергопотребления.

Кристалл EM260 не имеет встроенного АЦП и большого количества портов ввода/вывода, как у EM250, так как ориентирован только на поддержку стека ZigBee. В то же время характеристики приемной и передающей части кристалла EM260 совпадают с характеристиками EM250.

Подводя итог краткому сравнению приемопередатчиков Ember можно сделать вывод, что сегодня оба новых кристалла — Ember EM250 и EM260 — кажутся интересными. Кристалл EM250 позволяет создавать более компактные решения, а микросхема EM260 удобна для более сложных приложений, в которых тип основного процессора уже задан и разработчикам не хочется иметь дело с компиляцией стека ZigBee и осваивать малоизвестный процессор XAP2b.

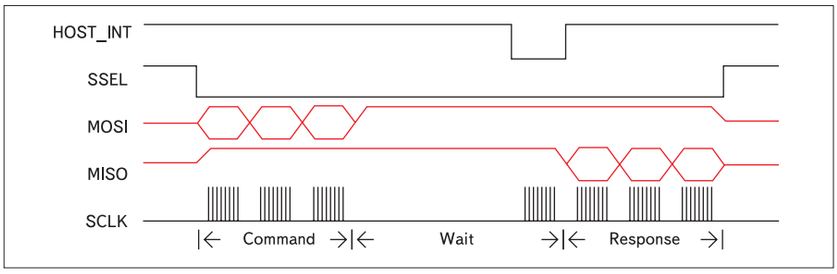

Рис. 3. Взаимодействие хоста и сопроцессора EM260 по интерфейсу SPI

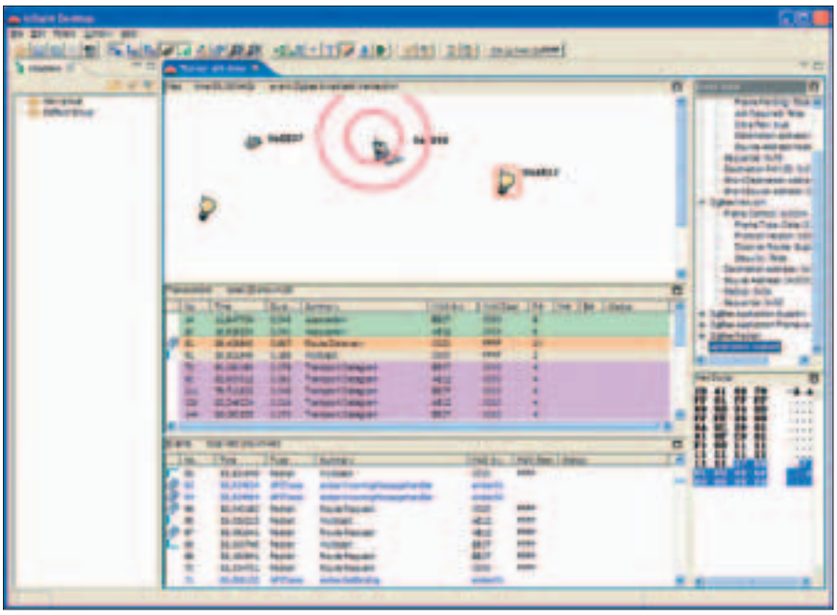

Рис. 4. Анализатор трафика ZigBee-сети InSightDeskTop

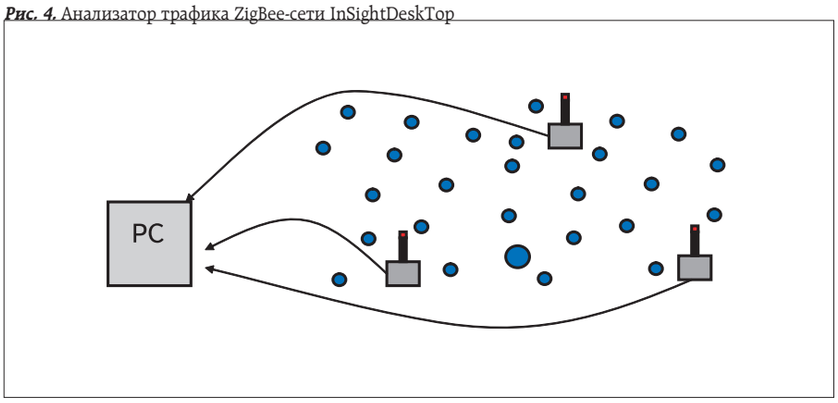

Рис. 5. Сбор трафика большой сети ZigBee при помощи нескольких снифферов

Отладочные средства Ember

Компания Ember уделяет большое внимание разработке отладочных средств, обеспечивающих профессиональный уровень настройки беспроводной сети. Основной задачей, которую ставят при этом разработчики Ember, является разработка таких средств, которые позволяли бы не только отлаживать программный код отдельного узла, но и осуществлять настройку и отладку приложения на сетевом уровне.

Отладочные средства Ember позволяют сегодня на одном персональном компьютере наблюдать все присутствующие в сети узлы и отслеживать взаимодействие между ними. Основной сетевой отладочной программой при этом является программа-анализатор трафика InSightDeskTop, рабочее окно которой представлено на рис. 4.

Эта программа имеет графическое окно, в котором можно визуально наблюдать взаимодействие узлов в сети. Широковещательные сообщения здесь отображаются расходящимися кругами, а адресные сообщения — направленными стрелками. Разработчик может загрузить в это окно план здания или местности и наблюдать взаимодействие беспроводных узлов с привязкой к объекту.

Кроме визуального отображения сети, программа InSightDeskTop осуществляет запись трафика в файл журнала. Программист имеет возможность просматривать последовательность пакетов низкого уровня, которые непосредственно слышны в эфире, а также наблюдать транзакции более высоких уровней (MAC-уровня, сетевого уровня или транспортного). Программа InSightDeskTop не только отображает все пакеты, но и анализирует их, выделяя в последовательности принятых байт различные поля, соответствующие различным уровням стека протоколов ZigBee.

Программа InSightDesktop взаимодействует с сетью через резервный канал Ethernet. Первые версии отладочных комплектов Ember предполагали, что все беспроводные узлы должны быть подключены во время отладки к резервному каналу Ethernet. Каждый узел, посылая сообщение в эфир, дублировал это сообщение и в резервный канал, обеспечивая, таким образом, сбор трафика сети.

Такой подход часто был затруднителен, так как не все узлы можно было подключить к резервному каналу Ethernet. В настоящее время эта проблема решается путем использования снифферов. Сниффер — это беспроводное устройство, выполненное на одном из кристаллов Ember, которое имеет выход на резервный канал Ethernet и в которое занесен программный код sniffer. Сниффер постоянно

слушает эфир и посылает в резервный канал все пакеты, которыми обмениваются беспроводные узлы, находящиеся в радиусе действия сниффера. В случае настройки большой сети можно иметь несколько сниффероов, например, в различных частях одного здания (рис. 5).

Хочется отметить, что программа InSightDeskTop может не только собирать трафик беспроводной сети, но также позволяет разработчику осуществлять перепрошивку встроенного программного кода узлов сети через резервный канал Ethernet или через радиоканал. Эта возможность позволяет разработчику распределенной сети все время находиться на своем рабочем месте и заниматься собственно разработкой кода для различных узлов сети, а не бегать с программатором от одного узла к другому.

На рис. 6 представлено многофункциональное устройство InSight Adapter. Его можно использовать в качестве программатора для кристаллов EM250 и EM260. Оно также позволяет подключить беспроводной узел, выполненный на базе кристаллов EM250 или EM260, к резервному каналу Ethernet. Наконец, это устройство совместно с целевым узлом, в который загружен код sniffer, становится сниффером беспроводной сети.

В заключение хочется отметить, что сети ZigBee анонсированы как высокоинтеллектуальные, самоорганизующиеся, самовосстанавливающиеся и простые в использовании беспроводные сети. Но эти качества были обещаны лишь конечным пользователям, а до этого должно быть потрачено много усилий программистов и системотехников, чтобы «вдохнуть» в конечные изделия эту самую интеллектуальность. Известно, что чем проще конечному пользователю, тем труднее программисту. Специалисты Ember сделали для достижения поставленной цели первые шаги: разработали библиотеку EmberZNet, приемопередатчики ZigBee и отладочные средства.

Рис. 6. Многофункциональное устройство InSight Adapter

Завершить начатую работу по созданию hi-tech систем предстоит вам, нашим читателям. Опыт последних лет показывает, что от разработчика конечного приложения на базе библиотек Ember также требуются серьезные усилия по программированию. Если перед вами стоит задача разработать ZigBee-сеть быстро и не тратить много времени на разработку программного обеспечения, то можно использовать для своей сети ZigBee-модемы ETRX2 компании Telegesis [2], которые построены на базе нового кристалла EM250, имеют встроенную антенну и уже «прошитое» программное обеспечение, которое значительно облегчит работу программиста конечного приложения. БТ

Литература

1. Кривченко Т. И. Беспроводные сети компании Ember: ZigBee и EmberNet // Беспроводные технологии. 2005. № 1.
2. Кривченко Т. И. ZigBee-модемы ETRX компании Telegesis // Беспроводные технологии. 2006. № 2.
3. Кривченко Т. И. Технология ZigBee // Коммунальный комплекс России. 2006. № 4.
4. EmberZNet Application Developer's Guide, 21 July, 2006, 120-0066-000J
5. EM260 ZigBee/802.15.4 Network Processor