ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ №4(35), 2017, с. 263–283
УДК 004.45

Ю. В. Шевчук, Е. В. Шевчук, А. Ю. Пономарёв, И. А. Фохт,
А. В. Елистратов, А. Ю. Вахрин, Р. Е. Яровицын

Etherbox: протокол для управления модульной сенсорной сетью

Аннотация. Протокол прикладного уровня Etherbox предназначен для взаимодействия управляющего компьютера с узлами сенсорной сети в форме исполняемых программ виртуальной машины, что позволяет обеспечить гибкость в управлении, необходимую для сенсорных сетях с модульной конструкцией узлов. В статье описаны принципы функционирования сенсорной сети, использующей протокол Etherbox, в сравнении с сетями, использующими протоколы MQTT-SN и CoAP. Рассматривается архитектура программного обеспечения снесорных узлов и управляющего компьютера.

Ключевые слова и фразы: сенсорная сеть, IoT, Etherbox, MQTT, MQTT-SN, CoAP, SNMP.

Введение

Мы рассматриваем сенсорную сеть, построенную по технологии TCP/IP по схеме, показанной на рис.1. Сеть объединяет сенсорные узлы, к которым подключены датчики и исполнительные механизмы. Пользователи получают доступ через Интернет к данным от датчиков: сырым или обработанным, живым или ретроспективным. Пользователи также имеют возможность управлять подключенными к сенсорным узлам исполнительными механизмами — вручную или автоматически.

Сенсорные узлы используют технологию TCP/IP и теоретически могут быть подключены к Интернет непосредственно — через шлюз 1...3 уровня, без преобразования протокола прикладного уровня. Но есть ряд причин, по которым в настоящее время общепринятой является разделение протоколов: один протокол используется для общения с сенсорными узлами, другой для доступа к сенсорной сети из Интернет. Эти причины следующие:

Работа частично поддержана программой ОНИТ РАН «Архитектурно-программные решения и обеспечение безопасности суперкомпьютерных информационно-вычислительных комплексов».

DOI: 10.25209/2079-3316-2017-8-4-263-283

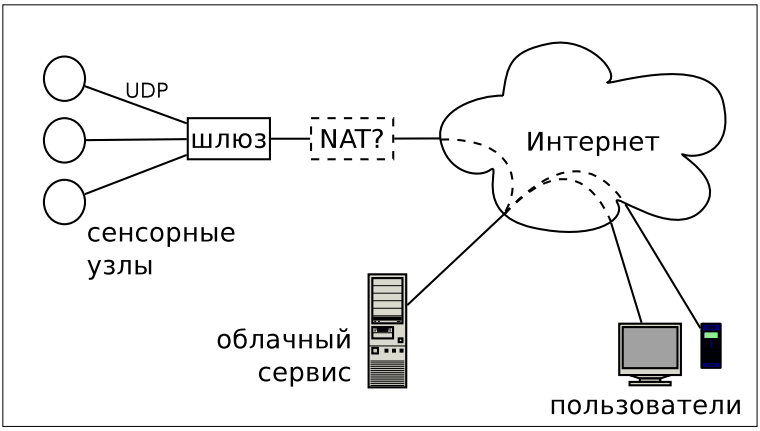


Рис. 1. Сенсорная сеть с доступом пользователей через Интернет

- сенсорные узлы, работающие в импульсном режиме («активность—сон»), не способны выступать в роли серверов; им удобно передавать данные в роли клиента по своему расписанию на какой-то сервер. Шлюз прикладного уровня может выступать в роли сервера как для сенсорных узлов, так и для Интернет-клиентов;
- низкая пропускная способность сенсорной сети требует специализированных протоколов с акцентом на минимизацию объёма передаваемых данных;
- низкая пропускная способность сенсорной сети делает её уязвимой не только для сетевых атак, но даже для постоянно присутствующего в Интернет сетевого шума;
- несовпадение версий протоколов межсетевого взаимодействия: в сенсорной сети используется IPv6, в то время как современные Интернет-провайдеры часто предоставляют только IPv4, возможно с NAT[**16**];
- удобные для сенсорных узлов протоколы на основе UDP плохо поддерживаются NAT (короткое время жизни информации о UDP-потоках в NAT с большим числом клиентов);
- удобные для Интернет-пользователей протоколы на основе TCP имеют недостатки с точки зрения применения в сенсорных сетях (см. раздел 2, стр. 267);
- для использования сенсорных сетей с широко распространёнными системами мониторинга, использующими протокол SNMP[**19**], необходимо разделение протоколов, поскольку протокол SNMP

плохо применим в сенсорных сетях. Существуют реализации шлюзов между протоколами сенсорных сетей и SNMP[22].

Вместе с тем, использование сенсорными узлами протоколов на основе IP всё же оправдано, поскольку даёт возможность строить сенсорную сеть с использованием нескольких различных сетевых технологий.

Протокол Etherbox относится к протоколам, используемым между шлюзом и узлами сенсорной сети. Мы рассматриваем протокол Etherbox в сравнении с двумя близкими по назначению протоколами: MQTT-SN[1] и CoAP[3]. Затем мы рассматриваем организацию программного обеспечения устройств сенсорной сети: сенсорных узлов и шлюза. В случае протокола Etherbox мы называем шлюз *управляющей станцией сенсорной сети*, поскольку этот компьютер не только играет роль шлюза между сенсорной сетью и Интернет, но и *организует* [8] работу сенсорной сети[1].

1. Модульная организация сенсорных узлов

Протокол Etherbox разработан для сенсорных сетей с модульной организацией узлов [9]. Модульная организация позволяет собрать из готовых модулей сенсорный узел с нужным количеством интерфейсов нужных типов. Модули объединяются с помощью шин расширения I^2C [7] и RS-485 [12][2]. Модули делятся на периферийные и базовые.

Периферийные модули имеют один или более интерфейсов для подключения датчиков и исполнительных механизмов (приводов). В сенсорном узле может быть установлено от одного до нескольких десятков[3] периферийных модулей. Программное обеспечение микроконтроллера периферийного модуля обслуживает интерфейсы датчиков, в некоторых случаях вносит калибровочные поправки в проходящие через модуль данные и реализует функции ведомого на шине расширения. Для реализации этих функций обычно достаточно небольшого микроконтроллера: процессор 8 бит, ОЗУ 128 байт, память программ 8 Кбайт.

[1]Возможен также вариант размещения управляющей станции в облаке. В этом случае функции шлюза редуцируются до туннелирования пакетов между сенсорной сетью и управляющей станцией.

[2]Не исключается использование и других шин расширения, например CANBus или Bluetooth LE.

[3]Максимальные количества интерфейсов определяются для каждой шины индивидуально исходя из разрядности адреса и характеристик физического интерфейса. Например, для современного RS-485 максимальное число модулей равно 256.

Базовый модуль имеет один или несколько интерфейсов для подключения к коммуникационной сети (IEEE 802.15.4, Bluetooth LE, WiFi, Ethernet, ...). В каждом сенсорном узле установлен один[4] базовый модуль. Программное обеспечение базового модуля реализует сетевой стек коммуникационного интерфейса, исполняет роль ведущего на шинах расширения узла и отвечает за передачу данных между периферийными модулями и коммуникационной сетью. В базовых модулях используются микроконтроллеры с 32-битным процессором, ОЗУ 32 Кбайт, памятью программ 128 Кбайт или более мощные.

Выбранный способ соединения модулей — с помощью шины, и реализация динамического назначения адресов на шине позволяют избежать конфликтов модулей между собой[5]. Благодаря этому, с аппаратной точки зрения сборка узла и пополнение его новыми периферийными модулями для подключения новых датчиков не представляет проблем. Но далее нужно поддержать все установленные модули в программном обеспечении базового модуля, с учётом типов подключенных к ним датчиков и приводов. В частности, для каждого датчика нужно задать сценарий опроса, выбор которого определяется не только типом датчика, но и его назначением в конкретной системе.

Как показывает опыт, настройку программного обеспечения узла на конкретное применение нежелательно делать до установки. Лучше, если монтажник устанавливает модули с универсальным программным обеспечением и подключает датчики к модулям в том порядке, в каком ему оказывается удобнее на месте. Монтажник должен документировать результаты работы: серийные номера установленных в каждом узле модулей и таблицу подключения датчиков к интерфейсам. Роль документации может играть фотография или видеосъёмка на смартфон. Далее по документации возможна удалённая конфигурация программного обеспечения сенсорного узла под конкретное применение.

В [**10**] для удалённой настройки узлов на конкретное применение предлагается использовать программы в форме байт-кода специализированной виртуальной машины — проглеты. Протокол Etherbox представляет собой средство доставки проглетов от управляющей станции к сенсорным узлам и результатов их выполнения от сенсорных узлов к управляющей станции.

---

[4]Для повышения надёжности возможно использование нескольких базовых модулей: один управляет шиной, остальные находятся в горячем резерве.

[5]Для сравнения, в модульных системах Arduino при комплектовании системы требуется согласование используемых каждым модулем сигналов ввода-вывода[**13**].

2. Протокол Etherbox

Взаимодействие с сенсорным узлом по протоколу Etherbox напоминает взаимодействие с удалённым устройством в режиме «командная строка/удалённый шелл» по протоколу rsh[11]. При работе в режиме «удалённый шелл» каждая передаваемая порция данных содержит одну или несколько команд, которые исполняются удалённым устройством в режиме интерпретации. Результат выполнения команд возвращается пользователю. Среди команд могут присутствовать операторы цикла, что позволяет задавать постоянно работающие сценарии функционирования устройства. Протокол Etherbox реализует ту же идею с учётом специфики сенсорных сетей и модульной организации сенсорных узлов.

Как и другие протоколы непосредственной связи с сенсорными узлами (MQTT-SN, CoAP[6]), протокол Etherbox реализован на базе протокола UDP [5]. Мотивация выбора протокола UDP следующая:

- лёгкость реализации протокола UDP в устройствах с ограниченными ресурсами по сравнению с TCP;
- протокол UDP даёт возможность использования для уменьшения трафика в сенсорной сети групповой (multicast) адресации;
- протокол TCP добивается доставки всех отправленных данных при помощи повторной отправки (*перепосылки*) потерянных пакетов, а для сенсорных сетей иногда важнее быстрая доставка свежих данных, чем полное отсутствие потерь;
- значительная (до нескольких секунд) нестабильность задержки передачи пакетов в сенсорной сети при работе сенсорных узлов в импульсном режиме (активность — сон) при использовании протокола TCP может вызывать ненужные перепосылки пакетов [24].

Поскольку сенсорные узлы представляют собой устройства с ограниченными ресурсами, команды должны быть представлены в виде, удобном для интерпретации сенсорным узлом. В случае «удалённого шелла» команды представляются как текстовые строки, которые требуют грамматического разбора перед интерпретацией. В протоколе Etherbox программы (*проглеты*) представляются в виде байт-кода специализированной виртуальной машины Etherbox32vm [10], который интерпретируется без предварительной подготовки.

[6]CoAP использует UDP, но в настоящее время также разрабатывается версия CoAP на основе TCP [23] для случаев, когда прохождение пакетов UDP затруднено брандмауэрами или NAT.

Результаты исполнения проглетов возвращаются на управляющую станцию в форме образа памяти проглета с изменениями, произошедшими в памяти в ходе исполнения проглета. Формирование результата в таком виде требует минимальных усилий от сенсорного узла. Для уменьшения объёма трафика есть возможность отправить в качестве ответа не весь образ проглета, а фрагмент образа, содержащий только нужную информацию.

2.1. Типы взаимодействия

Базовым типом взаимодействия для протокола Etherbox является тип «запрос—ответ». Сенсорный узел выступает как сервер, управляющая станция как клиент. Пакет-запрос содержит проглет, который исполняется в виртуальной машине Etherbox32vm сенсорного узла. Когда выполнение проглета завершается, нормально или аварийно, образ памяти проглета возвращается в пакете-ответе на управляющую станцию. В частности, в рамках этого типа взаимодействия можно выполнить считывание показаний одного или нескольких датчиков.

В системе команд виртуальной машины Etherbox32vm есть команды, позволяющие содержащемуся в запросе проглету задавать другие типы взаимодействия.

Тип «запрос без ответа» реализуется командой `bif`, подавляющей отправку ответа при завершении проглета. Этот тип взаимодействия полезен при использовании групповой (multicast) адресации, когда запрос обрабатывается одновременно многими узлами и хочется избежать перегрузки сети хоровым ответом от многих узлов. Хотя, если ответ на групповой запрос всё же нужен, добавление в проглет команды случайной задержки `mdelay` позволяет распределить ответы во времени и таким образом избежать перегрузки[7].

Тип «запрос—множественный ответ» реализуется командой `send`, вызывающей отправку ответа без завершения проглета. Если проглет большой, для уменьшения объёма трафика можно использовать команду `send3`, которая отправляет в качестве ответа не весь образ памяти проглета, а выбранный фрагмент. Тип «запрос—множественный ответ» подобен публикации данных в протоколе MQTT и имеет прямой аналог в протоколе CoAP: опцию observe[4], запрашивающую у узла повторение ответа на запрос при каждом изменении состояния запрашиваемого параметра. Возможности протокола Etherbox в этой части шире: в проглете можно алгоритмически определить понятие «изменение состояния», чтобы избежать неоправданно частых ответов.

[7] В протоколе CoAP аналогичный механизм случайной задержки реализуется на уровне протокола (Leisure period [3]).

Таблица 1. Структура пакета—запроса

Размер (байты)	Обозначение	Описание
20/40	iphdr	заголовок IPv4/IPv6
8	udphdr	заголовок UDP
16	hmac	код контроля целостности
4	sec	отметка времени (секунды)
3	usec	отметка времени (микросекунды)
1	handle	индекс обработчика ответа
0 . . . 1428	proglet	байт-код проглета

Таблица 2. Структура пакета—ответа

Размер (байты)	Обозначение	Описание
20/40	iphdr	заголовок IPv4/IPv6
8	udphdr	заголовок UDP
8	eui	уникальный идентификатор сенсорного узла (EUI-64)
16	hmac	код контроля целостности
4	sec	отметка времени (секунды)
3	usec	отметка времени (микросекунды)
1	handle	индекс обработчика ответа
1	exitcode	код завершения выполнения проглета
2	lastaddr	адрес завершения выполнения проглета
1	pad	зарезервированное поле
0 . . . 1416	image	дамп образа памяти проглета (фрагмент)

Например, считать изменением состояния изменение температуры не менее чем на 10 единиц АЦП и не ранее, чем через 10 секунд с момента предыдущего изменения состояния.

2.2. Структура пакетов

Структура пакетов протокола Etherbox показана в таблицах 1,2.

Поле `hmac` предназначено для контроля целостности и аутентичности пакета[14].

Поля `sec` и `usec` представляют собой отметку времени отправки пакета с управляющей станции, которая имеют двойное назначение. Во-первых, отметка времени используется для подавления атак методом повторения (replay attack): сенсорный узел отвергает пакет, если

не выполняется условие

$$(sec_i, usec_i) > (sec_{i-1}, usec_{i-1}), \tag{1}$$

где i — порядковый номер пакетов, поступающих от одного и того же источника. В случае изменения порядка пакетов в процессе передачи, что не исключается при использовании протокола UDP, это условие приводит к потере пакета. В таких случаях пакет будет послан повторно механизмом перепосылки на прикладном уровне.

Во-вторых, отметка времени используется для синхронизации внутренних часов сенсорного узла с астрономическим временем. Алгоритм синхронизации примитивный и в отличие от используемого в протоколе NTP [15] не учитывает задержку распространения пакетов и её вариацию, но гарантирует монотонность возрастания показаний внутренних часов. Монотонность важна, поскольку показания внутренних часов используются при заполнении полей `sec, usec` в ответных пакетах, для которых в момент получения на управляющей станции тоже будет проверяться условие 1 (стр. 270).

Поле «индекс обработчика ответа» (`handle`) позволяет управляющей станции выбирать нужный обработчик для пакета в случае, когда на сенсорном узле одновременно работает несколько проглетов. В режиме «удалённый шелл» для запуска на узле нескольких процессов одновременно пришлось бы использовать отдельное соединение для каждого процесса или добавлять информацию для демультиплексирования сообщений на прикладном уровне. Для модульных сенсорных узлов одновременная работа нескольких проглетов это штатный режим (раздел 3.4), поэтому информация для демультиплексирования (индекс обработчика ответа) добавлена в заголовок пакета.

Поле `proglet` содержит готовый к исполнению загружаемый образ проглета: команды и инициализированные данные.

Поле `eui` содержит уникальный идентификатор узла EUI-64 [20], который позволяет управляющей станции идентифицировать узел-отправитель пакета даже если он использует локальный IP-адрес [17] или адрес, полученный по протоколу DHCP [18] или находится за NAT [16].

Поле `exitcode` содержит код завершения проглета. Нулевой код указывает, что ответ отправлен в результате нормального завершения работы проглета, или отправлен без завершения проглета командой `send`. Отличный от нуля код указывает на аварийное завершение проглета; в этом случае в поле `lastaddr` содержится адрес команды виртуальной машины, при выполнении которой возникла ошибка.

Поле `image` содержит образ памяти всего проглета или его фрагмент, содержащий нужную управляющей станции информацию. Схема передаваемых данных известна обработчику ответа на управляющей станции, который выбирается по полю `handle`.

2.3. Начальная конфигурация сенсорных узлов

Начальная конфигурация — процедура, которая выполняется после подачи питания на сенсорный узел, и включает в себя присоединение к сети и вход в режим регулярного приёма-передачи данных. Описание процедуры начальной конфигурации часто остаётся за рамками спецификаций протоколов. Мы рассмотрим эту процедуру для протокола Etherbox, а также для протоколов MQTT-SN[1] и CoAP[3], насколько о ней можно судить по опубликованным спецификациям.

В случае протокола MQTT-SN узел логически является клиентом брокера MQTT, публикует и принимает сообщения, ассоциирование с определёнными темами (topics). Работа узла начинается с определения адреса шлюза. Адрес может быть заранее сконфигурирован (храниться в энергонезависимой памяти узла) или определяться динамически при помощи сообщений `ADVERTISE`, `SEARCHGW`, `GWINFO`. Далее, узел регистрирует публикуемые им имена тем при помощи сообщений `REGISTER` и подписывается на темы, которые он желает получать от брокера при помощи сообщений `SUBSCRIBE`. Наконец, узел переходит в рабочий режим, в котором регулярно публикует данные с помощью сообщений `PUBLISH`; имена публикуемых тем, соответствие тем датчикам и расписание публикации должны быть заранее сохранены в энергонезависимой памяти узла.

В случае протокола CoAP узел может быть как сервером — пассивно ожидать обращений от шлюза или других узлов, так и клиентом — активно обращаться к шлюзу или другим узлам. В протоколе предусмотрен механизм обнаружения (discovery), использующий групповую передачу (multicast) и позволяющий получить список ресурсов, предоставляемых каждым из узлов, по предопределённому URI `coap://host/.well-known/core`. CoAP узел—сервер после включения готов отвечать на запросы всем, кто знает URI его ресурсов, в том числе отвечать на запросы обнаружения. CoAP узел—клиент может обращаться к шлюзу и другим CoAP-узлам локального сегмента по заранее сконфигурированным адресам и сценариям, или использовать механизм обнаружения и затем обращаться к ресурсам обнаруженных узлов по заранее сконфигурированному сценарию.

В случае протокола Etherbox сенсорный узел логически является сервером, ожидающим обращений со стороны управляющей станции (шлюза). После включения питания узел готов принимать пакеты с проглетами и исполнять их. Узел не знает, какие датчики к нему подключены, хотя при помощи процедуры сканирования шин может получить список идентификаторов EUI-64 присутствующих на шинах расширения периферийных модулей. Узел может получить IP-адрес по DHCP или иметь только локальный адрес (link-local [**17**] [**21**]), но в любом случае сразу присоединяется к multicast-группе `etherbox-all`.

В цикл работы управляющей станции входит регулярная (сравнительно редкая, например раз в минуту) отправка на групповой адрес `etherbox-all` проглета с единственной командой `poll`. По этой команде сконфигурированные узлы отрабатывают цепочку команд, сохранённую командой `savepoll` — каждый свою, зависящую от роли узла в системе. На свежевключенном узле сохранённых команд нет, и узел отвечает управляющей станции пакетом с кодом ошибки `NEEDCONF`. Таким образом управляющая станция узнаёт о наличии узла, нуждающегося в конфигурации.

Конфигурация состоит в отправке узлу одного или нескольких проглетов. В простейшем случае это единственный конфигурационный проглет, содержащий команду `setkey` для установки ключа доступа, общего для всех узлов в данной подсети, и команду `savepoll` с пустой цепочкой команд. Сконфигурированный таким образом узел будет работать как сервер: ожидать запросов (например, «считать показания всех датчиков») и отвечать на них. Узел не будет отправлять никаких пакетов по собственной инициативе.

Более сложная конфигурация включает в себя отправку узлу проглетов с типом взаимодействия «запрос-множественный ответ». Такие проглеты остаются работать на узле постоянно, отправляя управляющей станции результаты опроса датчиков: периодически или при наступлении определённых условий. Это позволяет реализовать часть «издатель» в модели «издатель-подписчик». Что касается части «подписчик», она реализуется управляющей станцией: она отправляет удалённому брокеру запрос на подписку, а по приходе данных от брокера пересылает их узлу по протоколу Etherbox: управляющая станция выступает как клиент, узел как сервер. Подробнее этот процесс рассмотрен в разделе 3.1.

Таким образом, отличительной чертой сетей на основе протокола Etherbox является хранение конфигурации вне сенсорных узлов — на управляющей станции. Управляющая станция должна иметь информацию о всех узлах, которые могут появиться в сети (идентификатор EUI-64 и индивидуальный ключ доступа базового модуля),

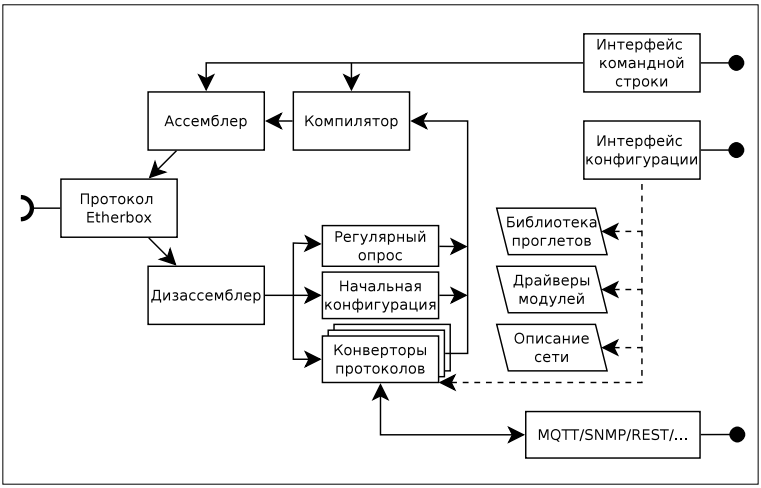


Рис. 2. Структура программного обеспечения управляющей станции

информацию об установленных в узле периферийных модулях (идентификаторы EUI-64 и типы модулей), о типах датчиков и приводов, подключенных к каждому периферийному модулю, о месте их установки и роли в системе. Эта информация (*описание сети*) вводится в систему администратором на основании документации, полученной от монтажников.

3. Архитектура программного обеспечения

3.1. Программное обеспечение управляющей станции

Управляющая станция поддерживает взаимодействие с сенсорными узлами по протоколу Etherbox, предоставляя пользователям доступ к сенсорной сети по одному или нескольким стандартным протоколам.

Структура программного обеспечения управляющей станции представлена на рис. 2.

Цепочка *Компилятор — Ассемблер — Протокол Etherbox* обеспечивает компиляцию проглета с заданными аргументами и отправку его в пакете протокола Etherbox сенсорному узлу или группе узлов. Поступающие из сенсорной сети пакеты обрабатываются модулем *протокол Etherbox*, который проверяет аутентичность ответа, извлекает содержащийся в ответе образ памяти проглета и передаёт его

дизассемблеру. Дизассемблер представляет образ в текстовом виде, удобном для дальнейшей обработки.

После дизассемблирования по полю `handle` заголовка выбирается обработчик ответа. Как правило, обработчик находится в том же программном модуле, который формировал проглет, приславший ответ. Два таких программных модуля — *Регулярный опрос* и *Начальная конфигурация* — являются постоянной частью программного обеспечения управляющей станции, а остальные — *Конверторы протоколов* — добавляются через *Интерфейс конфигурации*.

Благодаря тому, что ответ обрабатывается тем же программным модулем, который формировал запрос, структура данных в ответе известна обработчику. Не требуется наличия каких-либо тегов в массиве данных или определения схемы данных в явном виде. Автор модуля одновременно разрабатывает проглет, формирующий ответ, и обработчик ответа. Схема данных неявно присутствует в алгоритмах проглета и обработчика ответа.

Модуль *Регулярный опрос* с заданным в конфигурации интервалом отправляет проглеты с единственной командой `poll` по групповому адресу `etherbox-all`. Дополнительно возможна отправка запросов `poll` по индивидуальным адресам узлов, которые недоступны для multicast-трафика. По команде `poll` каждый узел отрабатывает цепочку команд, заданную во время начальной конфигурации.

Модуль *Начальная конфигурация* взаимодействует с узлами сразу после включения. Модуль использует информацию об аппаратной конфигурации сенсорного узла (перечень установленных в узле периферийных модулей, подключённых к узлу датчиков и приводов) из *Описания сети*, и проводит процедуру конфигурации как описано в разделе 2.3, посылая узлу один или более проглетов. При этом используются проглеты из *Библиотеки проглетов*. В процедуре начальной конфигурации участвуют также *Конверторы протоколов*, которые добавляют в список проглетов начальной конфигурации необходимые им проглеты.

Конверторы протоколов обеспечивают передачу данных из сенсорной сети пользователю и обратно — от пользователя в сенсорную сеть по стандартным протоколам. В качестве примера рассмотрим популярный протокол издатель-подписчик MQTT[**2**].

Конверторы Etherbox-MQTT работают как обработчики ответов, получая поступившие из сенсорной сети пакеты после дизассемблирования. Конвертор отображает координаты датчика (идентификатор периферийного модуля, идентификатор интерфейса периферийного модуля) в логическое пространство имён тем MQTT и передаёт данные *протоколу MQTT* для публикации.

Конверторы MQTT-Etherbox при старте управляющей станции отправляют по *протоколу MQTT* запросы на подписку. Список тем для подписки задаётся в *описании сети*. При поступлении данных по подписке данные демультиплексируются по имени темы и поступают в тот из конверторов, который отвечает за преобразование данной темы.

Поступление данных по подписке интерпретируется как команда одному или нескольким сенсорным узлам — например, включить реле. По данным из *описания сети* конвертор преобразует имя темы в координаты привода (идентификатор узла, идентификатор периферийного модуля, идентификатор интерфейса периферийного модуля), выбирает соответствующий команде проглет из *библиотеки проглетов*, и передаёт его *компилятору*. Поступившие по подписке данные фигурируют при компиляции как константы. Альтернативно, проглет может быть скомпилирован и ассемблирован заранее, а данные копироваться в область данных в бинарном образе проглета. Наконец, бинарный образ проглета передаётся сенсорным узлам по протоколу Etherbox, по индивидуальному или групповому IP-адресу.

Драйверы периферийных модулей представляют собой программные компоненты, предоставляющие методы для кодирования команд и декодирования ответов устройства. В образе памяти проглета данные обычно представлены в «сыром» виде — в виде блоков состояния периферийных модулей. Одной из функций драйверя периферийного модуля является извлечение нужных данных из блоков состояния и преобразование их из единиц АЦП в физические величины.

Традиционно в операционных системах драйвером называют программный компонент, непосредственно взаимодействующий с устройством и предоставляющий системе унифицированный интерфейс к устройству. Но в нашем случае с устройством через шины расширения взаимодействует проглет, выполняющийся на базовом модуле сенсорного узла, а драйвер находится на управляющей станции; непосредственное взаимодействие драйвера с устройством невозможно. Применяются два варианта использования драйверов периферийных модулей:

(*1*) проглет играет роль посредника, не понимающего, какую операцию он исполняет и не умеющего проанализировать результат операции. Проглет однократно исполняет подготовленную управляющей станцией команду `i2c`, и отправляет результат для анализа на управляющую станицию. Этот вариант подходит только для типа взаимодействия «запрос-ответ»;

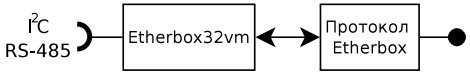


Рис. 3. Структура встроенного программного обеспечения базового модуля сенсорного узла

(*2*) проглет содержит подготовленные управляющей станцией команды `i2c` и содержит код, позволяющий извлекать для анализа отдельные поля из считанных с устройства статусных блоков и модифицировать команды перед отправкой устройству. Для генерации кода необходимо знание структуры данных, передаваемых на устройство и обратно; компилятор получает описание структуры от драйвера периферийного модуля. Этот вариант применяется в проглетах с типом взаимодействия «запрос-множественный ответ».

Интерфейс конфигурации предназначен для передачи на управляющую станцию *описания сети* и программного обеспечения: *библиотеки проглетов*, библиотеки *драйверов периферийных модулей* и *конверторов протоколов*. Интерфейс конфигурации реализуется как сервер протокола remote shell — удалённый интерпретатор командной строки UN*X, что позволяет как менять конфигурацию управляющей станции удалённо вручную, так и автоматизировать этот процесс.

Интерфейс командной строки играет вспомогательную роль: он позволяет оператору обратиться с запросом к любому узлу в терминах ассемблера виртуальной машины Etherbox32vm или на языке Etherbox2, получать ответы от узлов и наблюдать журнал взаимодействия с узлами в реальном времени.

3.2. Встроенное программное обеспечение периферийных модулей

Структура программного обеспечения периферийных модулей варьируется в зависимости от специфики модуля и не представляет интереса для настоящей статьи. Только один компонент присутствует в любом модуле: реализация функции ведомого на шине расширения I²Сили RS-485. Периферийные модули получают команды и передают данные только по обращениям от базового модуля.

3.3. Встроенное программное обеспечение базового модуля

Структура встроенного программного обеспечения базового модуля сенсорного узла показана на рис. 3. Она очень проста: в сетях на основе протокола Etherbox баланс сложности сознательно смещён от сенсорных узлов к управляющей станции.

Встроенное программное обеспечение универсально и реализует только одну функцию: исполнение проглетов, принимаемых по протоколу Etherbox. Виртуальная машина Etherbox32vm поддерживает одновременное исполнение нескольких проглетов в режиме мультипрограммирования без принудительного перепланирования: передача управления от проглета к проглету происходит только при выполнении проглетом команд, приостанавливающих выполнение (например, `mdelay`). Количество одновременно исполняемых проглетов ограничено только объёмом памяти микроконтроллера базового модуля и для микроконтроллеров с объёмом памяти 64КБ составляет около 20.

Поступающие из сенсорной сети пакеты обрабатываются модулем *протокол Etherbox*, который проверяет аутентичность запроса, извлекает содержащийся в пакете образ памяти проглета и передаёт его интерпретатору виртуальной машины *Etherbox32vm*. В ходе исполнения проглет может обращаться к периферийным модулям на чтение и на запись в роли ведущего на шинах расширения. Проглет также может отправить свой образ памяти, целиком или фрагмент, в качестве ответа на управляющую станцию — однократно или многократно.

3.4. Загружаемое программное обеспечение базового модуля

Загружаемое программное обеспечение обеспечивает настройку сенсорного узла на конкретное применение и состоит из одного или более проглетов, доставляемых на узел по протоколу Etherbox.

Количество резидентных проглетов на каждом сенсорном узле определяется управляющей станцией. Теоретически возможно реализовать опрос всех периферийных модулей в одном резидентном проглете. Но это решение сложно в реализации: относительно большой проглет придётся создавать отдельно для каждого сенсорного узла, и разбор ответа в конверторе протокола на управляющей станции тоже будет сложным. Кроме того, в этом варианте трудно реализовать разный период опроса для разных датчиков.

Более удобным оказывается вариант «по одному резидентному проглету для каждого периферийного модуля». В библиотеке проглетов на управляющей станции должен быть минимум один проглет для каждого типа периферийного модуля, используемого в сети. При компиляции проглеты параметризуются на основании описания сети — в частности, в качестве параметра задаётся идентификатор EUI-64 модуля, который должен обслуживать данный проглет. Если в узле установлено несколько однотипных периферийных модулей, один и тот же проглет компилируется несколько раз с разными параметрами.

Таким образом, архитектура загружаемого программного обеспечения в основном повторяет модульную конструкцию узлов сенсорной сети.

Отдельного обсуждения заслуживают проглеты, несущие команды управления приводами. Большинство периферийных модулей имеют несколько входных и выходных каналов, к которым подключены датчики и приводы, поэтому в момент прибытия проглета, несущего команду, на узле уже находится резидентный проглет, обслуживающий датчики того же периферийного модуля. В большинстве случаев операции периферийных модулей атомарны, так что проглет, несущий команду, может, никак не координируясь с другими проглетами, обратиться к модулю командой `i2c` и возвратить результат на управляющую станцию (тип взаимодействия «запрос—ответ»).

Но для некоторых типов периферийных модулей сеансы взаимодействия не атомарны. Рассмотрим в качестве примера модуль BB-RS232, реализующий интерфейс RS-232. Сеанс взаимодействия на интерфейсе RS-232 требует нескольких операций с модулем по шине I^2C:

(*1*) запись блока данных для передачи в интерфейс RS-232;

(*2*) чтение статуса модуля — ожидание ответа от устройства;

(*3*) чтение ответа устройства из FIFO-буфера модуля.

При отсутствии координации между проглетами, пытающимися одновременно провести сеанс связи на интерфейсе RS-232, возможно пересечение операций, влекущее нарушение дисциплины обмена (обращение к устройству, занятому исполнением другой команды).

Проблема координации решается следующим образом. Во время начальной конфигурации на узел загружается резидентный проглет, предоставляющий другим проглетам интерфейс (API) для доступа к интерфейсу RS-232 в форме публичной функции [**10**]. На входе в интерфейс реализуется блокировка одновременного доступа: проглеты, вызвавшие публичную функцию в момент, когда на интерфейсе RS-232 проходит другой сеанс взаимодействия, обнаруживают установленный флажок занятости интерфейса и выполняют команду задержки `mdelay` в цикле до тех пор, пока интерфейс не освободится.

3.5. Надежность доставки сообщений

При использовании протокола Etherbox в сочетании с протоколами на основе TCP, в частности MQTT, возникает проблема несоответствия уровней надёжности доставки сообщений. В протоколе MQTT даже на уровне QoS=0 (best effort – без гарантии доставки)

надёжность доставки обеспечивается нижележащим протоколом TCP. Протокол Etherbox реализован на базе протокола UDP, и потери пакетов в сенсорной сети никак не обрабатываются на уровне протокола.

Возможны следующие подходы к обеспечению надёжности доставки команд сенсорным узлам:

(*1*) не контролировать потери пакетов между управляющей станцией и сенсорным узлом; для обеспечения надёжной доставки использовать возможности перепосылки пакетов протоколом MQTT на уровнях QoS=1, QoS=2;

(*2*) реализовать контроль потерь и повторную отправку пакетов на уровне конвертора протокола — хранить полученное по протоколу MQTT сообщение `PUBLISH` до получения ответа от сенсорного узла;

В части передачи данных от сенсорных узлов есть следующие варианты:

(*1*) не контролировать потери пакетов между сенсорным узлом и управляющей станцией, смириться с потерями и надеяться, что следующая порция данных потеряна не будет;

(*2*) реализовать контроль потерь на уровне конвертора протокола — поддержать отправку подтверждений получения данных конвертором и повторную отправку данных проглетом в случае отсутствия подтверждения.

Выбор вариантов определяется программами конверторов протоколов, так что все варианты могут использоваться одновременно, для каждого датчика может быть выбран наиболее подходящий для него вариант.

Заключение

Протокол прикладного уровня Etherbox реализует взаимодействие между узлами сенсорной сети и управляющей станцией (сервером) в форме небольших программ специализированной виртуальной машины Etherbox32vm (проглетов). Такой способ взаимодействия даёт большую гибкость в управлении сенсорными узлами при сравнительно простом и унифицированном встроенном программном обеспечении сенсорных узлов. Это свойство особенно ценно для сенсорной сети с модульной конструкцией узлов: узлы можно комплектовать из модулей непосредственно на месте установки без предварительной настройки программного обеспечения узлов.

Встроенное программное обеспечение обеспечивает подключение узла к сети, а настройка узла на конкретное применение с поддержкой каждого из установленных в узле модулей выполняется удалённо при помощи протокола Etherbox. Настроенный узел может работать как в пассивном режиме, передавая данные по запросу управляющей станции, так и в активном, отправляя данные по расписанию или при наступлении определённых условий. Сочетание обоих режимов может использоваться для реализации модели взаимодействия «издатель-подписчик».

Универсальность и упрощение программного обеспечения сенсорных узлов достигается за счёт усложнения программного обеспечения управляющей станции сети: компьютера, имеющего полную информацию об узлах сети, координирующего её работу и играющего роль шлюза между сенсорной сетью и Интернет. Это усложнение представляется оправданным, поскольку управляющая станция это полноценный компьютер с виртуальной памятью, развитыми средствами программирования и сетевым питанием, а то время как сенсорные узлы — резко ограниченные по памяти, процессору и электропитанию устройства со спартанской средой программирования.

Список литературы

[1] MQTT For Sensor Networks (MQTT-SN) Protocol Specification Version 1.2., 2013, URL: http://mqtt.org/MQTT-S_spec_v1.2.pdf ↑ 265,271

[2] *MQTT 3.1.1 specification*, OASIS, 2015, URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html ↑ 274

[3] Z. Shelby, K. Hartke, C. Bormann. *The Constrained Application Protocol (CoAP)*, RFC7252, RFC Editor, 2014, URL: http://www.rfc-editor.org/rfc/rfc7252.txt ↑ 265,268,271

[4] K. Hartke. *Observing Resources in the Constrained Application Protocol (CoAP)*, RFC7641, RFC Editor, 2015, URL: http://www.rfc-editor.org/rfc/rfc7641.txt ↑ 268

[5] J. Postel. *User Datagram Protocol*, RFC768, RFC Editor, 1980, URL: http://www.rfc-editor.org/rfc/rfc768.txt ↑ 267

[6] J. Postel. *Transmission Control Protocol*, RFC793, RFC Editor, 1981, URL: http://www.rfc-editor.org/rfc/rfc793.txt ↑

[7] *UM10204. I^2C-bus specification and user manual*, NXP Semicondictor, 2014, 64 p., URL: http://www.nxp.com/documents/user_manual/UM10204.pdf ↑ 265

[8] М. Д. Недев, Ю. В. Шевчук. «Сенсорная сеть с организацией извне», *Труды Третьей российской конференции с международным участием «Технические и программные средства систем управления, контроля и измерения»*, УКИ'12 (Москва, 16–19 апреля 2012 г.), ИПУ РАН, М., 2012, ISBN: 978-5-91450-100-3. ↑ 265

[9] С. М. Абрамов, Ю. В. Шевчук, А. Ю. Пономарев, С. М. Пономарева, Е. В. Шевчук. «Сенсорная сеть с модульной архитектурой», *Программные системы: теория и приложения*, **6**:4(27) (2015), с. 197–208, URL: http://psta.psiras.ru/read/psta2015_4_197-208.pdf ↑ 265

[10] Ю. В. Шевчук, А. Ю. Шевчук. «Виртуальная машина "Etherbox32vm"», *Программные системы: теория и приложения*, **7**:4(31) (2016), с. 119–143, URL: http://psta.psiras.ru/read/psta2016_4_119-143.pdf ↑ 266, 267, 278

[11] rsh - remote shell. Linux man page, URL: https://linux.die.net/man/1/rsh ↑ 267

[12] *Electrical Characteristics of Generators and Receivers for Use in Balanced Digital Multipoint Systems*, TIA/EIA Interim Standard, TIA/EIA-485-A, Telecommunications Industry Association, Arlington, VA, USA, 1998. ↑ 265

[13] *Arduino: Shield Pin Usage*, URL: http://playground.arduino.cc/Main/ShieldPinUsage ↑ 266

[14] JH. Song, R. Poovendran, J. Lee, T. Iwata. *The AES-CMAC Algorithm*, RFC4493, RFC Editor, 2006, URL: http://www.rfc-editor.org/rfc/rfc4493.txt ↑ 269

[15] D. Mills, J. Martin, J. Burbank, W. Kasch. *Network Time Protocol Version 4: Protocol and Algorithms Specification*, RFC5905, RFC Editor, 2010, URL: http://www.rfc-editor.org/rfc/rfc5905.txt ↑ 270

[16] P. Srisuresh, M. Holdrege. *IP Network Address Translator (NAT) Terminology and Considerations*, RFC2663, RFC Editor, 1999, URL: http://www.rfc-editor.org/rfc/rfc2663.txt ↑ 264, 270

[17] S. Cheshire, B. Aboba, E. Guttman. *Dynamic Configuration of IPv4 Link-Local Addresses*, RFC3927, RFC Editor, 2005, URL: http://www.rfc-editor.org/rfc/rfc3927.txt ↑ 270, 272

[18] R. Droms. *Dynamic Host Configuration Protocol*, RFC2131, RFC Editor, 1997, URL: http://www.rfc-editor.org/rfc/rfc2131.txt ↑ 270

[19] *SNMP RFCs*, URL: http://www.snmp.com/protocol/snmp_rfcs.shtml ↑ 264

[20] *Guidelines for Use of Extended Unique Identifier (EUI), Organizationally Unique Identifier (OUI), and Company ID (CID)*, IEEE, 2017, URL: http://standards.ieee.org/develop/regauth/tut/eui.pdf ↑ 270

[21] S. Thomson, T. Narten, T. Jinmei. *IPv6 Stateless Address Autoconfiguration*, RFC4862, RFC Editor, 2007, URL: http://www.rfc-editor.org/rfc/rfc4862.txt ↑ 272

[22] H. Lindholm-Ventola, B. Silverajan. *CoAP-SNMP Interworking in IoT Scenarios*, Tampere University of Technology. Department of Pervasive Computing, 2014, ISBN: 978-952-15-3219-1. ↑ 265

[23] C. Bormann, S. Lemay, H. Tschofenig, K. Hartke, B. Silverajan, B. Raymor, Ed.. *CoAP (Constrained Application Protocol) over TCP, TLS, and WebSockets*, IETF, 2017, URL: https://tools.ietf.org/html/draft-tschofenig-core-coap-tcp-tls ↑ 267

[24] M. Scharf, M. Necker, B. Gloss. "The Sensitivity of TCP to Sudden Delay Variations in Mobile Networks", Networking 2004, Lecture Notes in Computer Science, vol. **3042**, eds. N. Mitrou, K. Kontovasilis, G.N. Rouskas, I. Iliadis, L. Merakos, Springer, Berlin–Heidelberg, 2004, pp. 76–87. ↑ 267

Рекомендовал к публикации *д.ф.-м.н. С. В. Знаменский*

Пример ссылки на эту публикацию:

Ю. В. Шевчук, Е. В. Шевчук, А. Ю. Пономарёв и др. «Etherbox: протокол для управления модульной сенсорной сетью», *Программные системы: теория и приложения*, 2017, **8**:4(35), с. 263–283.

URL: http://psta.psiras.ru/read/psta2017_4_263-283.pdf

Эта же статья по-английски: DOI 10.25209/2079-3316-2017-8-4-285-303

Об авторах:

Юрий Владимирович Шевчук

Заведующий лабораторией телекоммуникаций ИЦМС ИПС им. А.К. Айламазяна РАН, к.т.н. Область интересов: системное программирование, цифровая электроника, сети компьютеров, сенсорные сети, мониторинг и управление территориально распределёнными объектами, распределённое программирование

e-mail: sizif@botik.ru

Елена Васильевна Шевчук

Старший научный сотрудник ИЦМС ИПС им.А.К.Айламазяна РАН. Область интересов: распределенные вычисления, сенсорные сети, мониторинг и управление территориально распределёнными объектами

e-mail: shev@shev.botik.ru

Александр Юрьевич Пономарёв

Ведущий инженер ИЦМС ИПС им.А.К.Айламазяна РАН. Область интересов: цифровая и аналоговая схемотехника, сенсорные сети, импульсные преобразователи напряжения

e-mail: harry@opus.botik.ru

Игорь Анатольевич Фохт

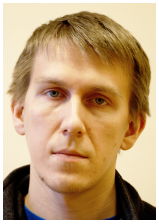
Старший научный сотрудник ИЦМС ИПС им.А.К.Айламазяна РАН. Область интересов: Сенсорные сети, окружающий интеллект, распределённые вычисления, системы автоматического управления, метрология

e-mail: vogt@vgt.botik.ru

Алексей Викторович Елистратов

Инженер-исследователь ИЦМС ИПС им.А.К.Айламазяна РАН. Область интересов: цифровая и аналоговая схемотехника, сенсорные сети, системы автоматизации проектирования

e-mail: concept@pereslavl.ru



Андрей Юрьевич Вахрин

Инженер-исследователь ИЦМС ИПС им.А.К.Айламазяна РАН. Область интересов: цифровая и аналоговая схемотехника, сенсорные сети, мониторинг электросетей

e-mail: dispells@pereslavl.ru

Роман Евгеньевич Яровицын

Инженер-исследователь ИЦМС ИПС им.А.К.Айламазяна РАН. Область интересов: цифровая и аналоговая схемотехника, сенсорные сети, мониторинг электросетей

e-mail: develop@pereslavl.ru