электронный научно-технический журнал
ИНЖЕНЕРНЫЙ ВЕСТНИК
Издатель Общероссийская общественная организация "Академия инженерных наук им. А.М. Прохорова"
ISSN 2307-0595

Реализация технологии обмена сообщениями между двумя смарт-устройствами посредством Bluetooth-канала

 # 12, декабрь 2017

Хохлов С. А., Глебов А. С., Прохорова З. Р.

УДК: 004.773

МГТУ им. Н.Э. Баумана, Москва, Россия
s.a.khokhlov@gmail.com
aleksey.glebov2010@yandex.ru
zlata.oz@yandex.ru

Введение

Неотъемлемой частью IT-индустрии являются вычислительные сети, одной из важнейших функций которых является пересылка данных. С учетом широкой распространенности портативных устройств (как правило, имеющих аккумуляторы малых объемов), низкий уровень энергопотребления Bluetooth-адаптера и высокая скорость передачи данных делают Bluetooth-канал важной технологией для обмена сообщениями. В связи с этим, реализация подключения двух портативных устройств с использованием Bluetooth-протокола является актуальной темой на современном уровне развития IT-индустрии.

В рамках производственной практики на предприятии Samsung RnD Institute RUS был разработан алгоритм обмена данными по Bluetooth-каналу между двумя устройствами Samsung Gear S2 (смарт-часы с операционной системой Tizen). Данный алгоритм был впоследствии использован при реализации тестового игрового приложения, поддерживающего режим для двух игроков.

Исследование и анализ интернет-ресурсов показали, что аналогов подобной технологии для данных устройств не существует. Более того, согласно информации производителя, организация обмена данными по Bluetooth-каналу между несколькими экземплярами этих часов не была предусмотрена при их разработке.

В данной статье рассмотрена технология передачи данных с помощью Bluetooth, реализация обмена сообщениями между двумя носимыми устройствами Samsung Gear S2, а также проанализированы проблемы, возникающие при передаче данных посредством данной технологии и предложены их решения.

1. Обзор технологии передачи данных в сети Bluetooth. Обмен сообщениями.

Bluetooth - технология беспроводной связи, обеспечивающая экономный (с точки зрения потребляемой энергии) и дешевый способ радиосвязи между различными типами электронных устройств. Интерфейс Bluetooth позволяет передавать данные различного типа. Существует множество “слоев” спецификации Bluetooth, которые позволяют работать с данными различного типа [1], некоторые из них представлены на рис. 1.

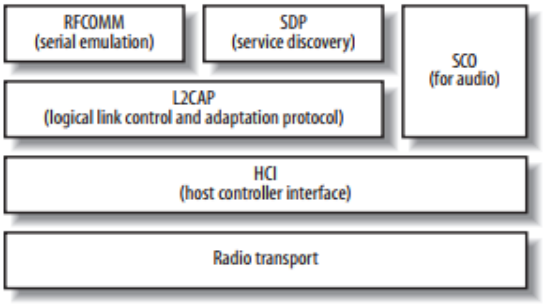

Рис. 1. Слои спецификации Bluetooth

Для передачи данных могут быть использованы асимметричный (721 Кбит/сек в одном направлении и 57,6 Кбит/сек в другом) и симметричный методы (432,6 Кбит/сек в обоих направлениях). Работающий на частоте 2.4 ГГц приемопередатчик, позволяет в зависимости от степени мощности устанавливать связь в пределах 10 или 100 метров. Соединение в пределах 10 м позволяет сохранить низкое энергопотребление, компактный размер и достаточно невысокую стоимость компонентов.

Каждое Bluetooth-устройство имеет свой Bluetooth адрес (называемый BDADDR); этот адрес похож на MAC-адрес Ethernet-устройств и следует тем же соглашением об адресах, указанных в стандарте ANSI/IEEE 802, установленном IEEE. Первые три октета этого адреса обозначают организовано унифицированный идентификатор (OUI), уникальный для каждого производителя.

Связывание (сопряжение) - это процесс установления “доверия” между двумя устройствами Bluetooth. Пользователь должен ввести соответствующие коды, называемые персональными идентификационными номерами (ПИН) на два устройства. В некоторых ситуациях одно из устройств может иметь предварительно настроенный PIN-код. PIN-коды обычно представляют собой последовательность цифр; они обеспечивают небольшую безопасность и предназначены только для сопряжения. После успешного соответствия PIN-кодов, устройства согласовывают ключ ссылки, более криптографически защищенный код, который затем используется в качестве механизма управления доступом между двумя устройствами.

Bluetooth-устройства считаются обнаруживаемыми, если могут быть найдены другими устройствами. В процессе поиска опрашивающее устройство устраивает широковещательную рассылку особо закодированного сообщения. Если устройство принимает это сообщение, оно посылает ответное сообщение, обозначая свое присутствие. Как правило, устройство должно быть обнаруживаемым для установления соединения.

Bluetooth работает по принципу FHSS (Frequency-Hopping Spread Spectrum), т. е. передатчик разбивает данные на пакеты и передает их по псевдослучайному алгоритму скачкообразной перестройки частоты (1600 раз в секунду), или шаблону (pattern), составленному из 79 подчастот. Таким образом, обмен сообщениями осуществляется только между двумя устройствами, которые каждые 625 мкс (один временной слот) синхронно перестраиваются с одной несущей частоты на другую по псевдослучайному алгоритму, определяющемуся отдельно для каждого соединения.

2. Bluetooth-сокет. Алгоритм соединения Bluetooth-устройств

Под сокетом обычно понимают программный интерфейс для обмена данными между процессами, которые могут исполняться как на одном устройстве, так и на различных, связанных между собой сетью. Сокет - абстрактный объект, представляющий конечную точку соединения. Следует иметь в виду, что существуют два типа сокетов: сокет сервера и сокет клиента, и зачастую они работают в паре.

По принципу работы сокеты делятся на два вида: потоковые сокеты и датаграммные сокеты (рис. 2).

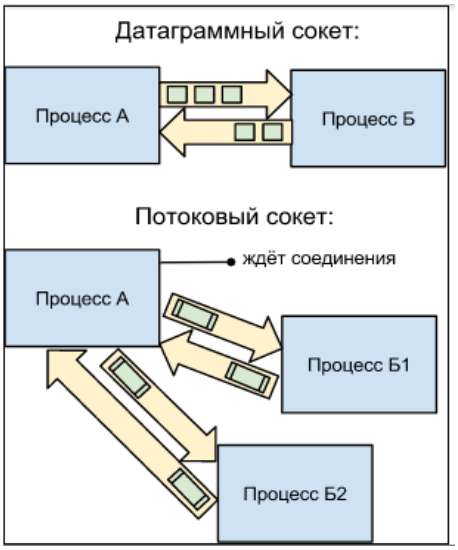

Рис. 2. Виды сокетов

Сокет датаграмм также называют сокетом без соединения. Этот сокет не устанавливает соединения между клиентом и сервером: передаваемые дейтаграммы могут быть не доставлены, доставлены с задержкой или доставлены без задержки (в таком случае данные в дейтаграмме гарантированно корректные). Также не гарантируется порядок доставки пакетов, а поток данных не контролируется.

Сокет потоков называют сокетом с установкой соединения. Этот вид соединения является более надежным с точки зрения доставки пакетов и ее гарантированности. В отличие от сокета датаграмм, в результате использования протокола TCP этот вид сокетов контролирует поток данных, гарантирует доставку пакетов, а также их порядок. Минусом данного сокета является более низкая скорость передачи, что обусловлено повторной передачей потерянных пакетов и большим количеством выполняемых операций над пакетами.

Транспортный протокол RFCOMM, представленный на рис. 1, реализует создание Bluetooth-сокетов, которые работают аналогично сокетам потоков в интернет-протоколах [2-4]. То есть, производится установка соединения, поток данных контролируется, доставка пакетов гарантирована, а также соблюдается порядок передачи пакетов.

Bluetooth-сокет в общем случае представляет собой набор функций, используемых для обмена данными между двумя bluetooth-устройствами, причем каждое из устройств может выполнять роль как сервера (поставщик сервиса), так и клиента (пользователь сервиса). В зависимости от роли устройства, процесс создания соединения для устройств различен. Процессы обмена данными и прекращения соединения, в свою очередь, для обоих устройств будут одинаковы.

При попытке открытия соединения с другим устройством, выполняется поиск протокола обнаружения служб (SDP), и определяются протокол и канал, использующиеся для соединения. Если соединение установлено и сокет успешно открыт, возвращается экземпляр BluetoothSocket с открытым состоянием. Затем этот сокет используется для обмена данными между подключенными устройствами.

Для установки соединения устройство-сервер регистрирует сервис и создает сокет для отслеживания и разрешения входящих запросов на соединения. В свою очередь, устройство-клиент подключается к сервису и ожидает соединения. После установки соединения оба устройства могут обмениваться данными. Когда обмен данными окончен, соединение между устройствами должно быть разорвано, а сервис должен быть закрыт [5].

3. Реализация алгоритма соединения Bluetooth-устройств Gear S2

Как уже было отмечено в разделе 2, для соединения двух устройств по Bluetooth-каналу были реализованы серверная и клиентская части. Одно из устройств открывает серверный сокет, а второе - инициализирует соединение, используя MAC-адрес сервера. Сервер и клиент считаются соединенными, когда они оба имеют активный BluetoothSocket на одном и том же RFCOMM-канале [6]. После этого они могут получать и отправлять потоки данных. Сервер и клиент по-разному получают требуемый BluetoothSocket. Сервер получает его, когда входящее соединение принято. Клиент - когда открывает RFCOMM для сервера.

При соединении устройств одно из них ведет себя как сервер, то есть удерживает открытый BluetoothServerSocket. Цель сервера - ждать запроса на входящее соединение, и когда оно подтверждено, создать BluetoothSocket. С точки зрения сервера процедура подключения выглядит следующим образом:

- С помощью вызова метода registerRFCOMMServiceByUUID() происходит регистрация сервиса и сервер получает BluetoothServerSocket. Одним из параметров вызываемого метода является идентификационное имя сервиса. Система автоматически добавит его в базу Service Discovery Protocol (SDP). В качестве параметров метода также указывается UUID сервера, который используется клиентом для установления соединения.

- Начинается прослушивание запроса на соединение через метод onconnect(). Это блокирующий метод, который возвращает результат, когда соединение подтверждено, либо, когда сгенерировано исключение. Соединение считается подтвержденным, если удаленное устройство пошлет запрос на соединение с UUID, указанным при регистрации серверного сокета. В случае успеха метод onconnect() возвращает настроенный на соединение сокет.
- Если возникает необходимость принять дополнительное соединение, вызывается метод close(). Это приведет к освобождению сокета и всех его ресурсов, но не закроет соединенный сокет. В отличие от TCP/IP, протокол RFCOMM позволяет работать только с одним клиентом на канале, поэтому в большинстве случаев имеет смысл вызывать метод close() сразу же после принятия сокета.

Для инициализации соединения с удаленным серверным устройством, устройство-клиент получает объект BluetoothDevice, содержащий информацию о сервере. Этот объект используется для инициализации сокета и Bluetooth-соединения. Со стороны клиента процедура подключения к серверу выглядит следующим образом:

- С помощью метода connectToServiceByUUID() инициализируется подключение к серверу и происходит запрос на создание сокета на стороне клиента. Значение параметра UUID должно совпадать со значением, указанным сервером при регистрации сервиса.
- Вызов метода onconnect() происходит при установлении соединения. После вызова этого метода система будет выполнять SDP поиск на удаленном устройстве, чтобы сопоставить UUID. В случае успеха, при условии подтверждения запроса со стороны сервера, будет открыт RFCOMM канал. Это блокирующий вызов. Если по каким-то причинам соединение сорвется или выйдет timeout, будет сгенерировано исключение.

После успешного соединения, каждое из соединенных устройств имеет объект BluetoothSocket с помощью которого можно реализовать передачу/прием данных. Чтение и запись данных осуществляется с помощью методов readData() и writeData() соответственно [6-7].

Прежде чем открывать канал передачи данных и создавать сокеты, устройства связываются стандартными средствами Bluetooth. Процесс связывания представлен в листинге 1 (*здесь и далее программный код представлен на языке Javascript*):

Листинг 1:

```
function multiOnload(){
        if (adapter.address == "AC:EE:9E:BE:C3:3C"){
                adapter.createBonding('00:02:A6:9C:5B:38', BluetoothSuccessServer, function(er) {});
        }else{
                adapter.getDevice('AC:EE:9E:BE:C3:3C',
                BluetoothSuccessClient, function(er) {});
        }}
```

BDADDR-адреса сопрягаемых устройств зашиты в программный код. Вызвано это тем, что поиск других bluetooth-устройств для Gear S2 был предусмотрен производителем и не представляет собой особого интереса для разрабатываемой технологии.

Представленная в листинге 1 функция multiOnload() проверяет является ли устройство сервером или клиентом. В первом случае для объекта BluetoothAdapter вызывается метод createBonding(), принимающая на вход bluetooth-адрес клиента и в случае успеха вызывающая функцию BluetoothSuccessServer(), которая регистрирует сервис, к которому будет подключаться клиент. Если же устройство является клиентом, то для объекта BluetoothAdapter вызывается метод getDevice(), принимающая на вход bluetooth-адрес сервера и в случае успеха вызывающая функцию BluetoothSuccessClient(), совершающее подключение к указанному устройству.

В листингах 2 и 3 представлены функция регистрации сервиса сервером и функция подключения к нему клиента соответственно.

Листинг 2:

```
function BluetoothSuccessServer(device){
        adapter.registerRFCOMMServiceByUUID("39E9AE15-62E4-
                4529-9019-8A2C07A27051", "TTService",onUUIDsuccess,onUUIDerror);
}
```

Листинг 3:

```
function BluetoothSuccessClient(device){
        device.connectToServiceByUUID("39E9AE15-62E4-
                4529-9019-8A2C07A27051",onSocketConnected, function(er) { });
}
```

Следует уточнить, что функция регистрации получает на вход UUID сервера и его имя, в случае успеха вызывает функцию onUUIDsuccess() (см. листинг 4), которая создает сокет и выполняет передачу сообщений. Функция подключения, в свою очередь, также получает на вход UUID сервиса, к которому должна подключиться, и, в случае успеха, вызывает клиентскую функцию onSocketConnected() (см. листинг 5) для создания сокета и обмена сообщениями.

Листинг 4:

```
function onUUIDsuccess(handler) {
  alert("Server created");
  serviceHandler = handler;
  handler.onconnect=function(socket){
        alert("Connection success");
        $("#multi_text").remove();
        startGameMulti();
        sock = socket;
        socket.onmessage = function(){
                var data = socket.readData();
                serverMsg= "";
                for (var i = 0; i < data.length; i++){
                        serverMsg += String.fromCharCode(data[i]);
                }
                var platform2 = document.querySelector("#platform2");
```

```
                platform2.style.transform = 'rotate(' + serverMsg + 'rad)';
                rad2 = Number(serverMsg);
        };
        socket.onclose = function(){
                sock = null;
        };
  };
}
```

Данная функция отслеживает подключение клиента к серверу и с помощью метода socket.onmessage() перехватывает записываемые в сокет сообщения, а затем обрабатывает их на стороне сервера.

Листинг 5:

```
function onSocketConnected(socket){
  clientSocket = socket;
  alert("Connection success");
  $("#multi_text").remove();
  startGameMulti();
  socket.onmessage = function() {
        var data = socket.readData();
        datamsg= "";
        for (var i = 0; i < data.length; i++){
                datamsg += String.fromCharCode(data[i]);
        }
        if (datamsg.charAt(0)!=="<"){
                var platform2 = document.querySelector("#platform2");
                platform2.style.transform = 'rotate(' + datamsg + 'rad)';
                rad2 = Number(datamsg);
                datamsg= "";
        } else {
                var strong="";
                var mas = new Array();
                mas = datamsg.split("<");
                strong = strong.concat( "<", mas[1], "<", mas[2], "<", mas[3], "<",
                                                                        mas[4],"<", mas[5]);
                $("#game_place").append(strong);
                datamsg= "";
        }};
  socket.onclose = function(){};
}
```

Данная функция работает аналогичным образом, но на стороне клиента. Здесь имеет место более сложная обработка приходящих сообщений, так как нужно выполнить их перевод из чар-кода, в котором передаёт Bluetooth-канал, в текстовый формат, что обусловлено спецификой разрабатываемого приложения.

4. Проблемы и решения. Анализ задержек при передаче сообщений

Одной из проблем, встающих при работе с Bluetooth могут быть взаимные помехи с мобильной сетью LTE-стандарта. Однако, вследствие использования производителем Bluutooth-спецификации версии 4.1, этого не происходит, так как пакеты данных автоматически координируются [8].

Так как передача по протоколу RFCOMM реализована аналогично передаче по протоколу TCP, то проблема потери сообщений не возникает, но увеличивается время доставки сообщений. Таким образом, основной проблемой, возникающей при реализации описываемой технологии, является задержка при передаче сообщений. Для решения данной проблемы было принято решение уменьшить размер передаваемого сообщения в тестовом приложении.

В исходном варианте реализации алгоритма, ориентированного на передачу сообщений тестовым приложением, сервер в каждом сообщении передавал клиенту положение всех точек на экране (см. рис. 3.), однако, при увеличении числа точек, эти сообщения становились очень большими, и происходили потери, так как следующий вызов сокета происходил раньше, чем предыдущее сообщение успевало полностью записаться. В связи с этим было принято решение передавать только начальные данные для каждой точки, а затем осуществлять одинаковую обработку на каждом из устройств. Такой подход позволил добиться минимальной задержки передачи сообщений. В самом худшем случае происходила рассинхронизация изображения на один кадр, что можно увидеть на рис. 3.: на устройстве-клиенте еще осталось несколько точек на самой внешней орбите (они пропадут в следующий момент времени), а на устройстве-сервере они уже пропали.

Рис. 3. Пример рассинхронизации устройств

Анализ ситуации, изображенной на рис.3., позволяет оценить задержку между устройствами. При наложении изображений друг на друга получается картина, схематично представленная на рис.4.

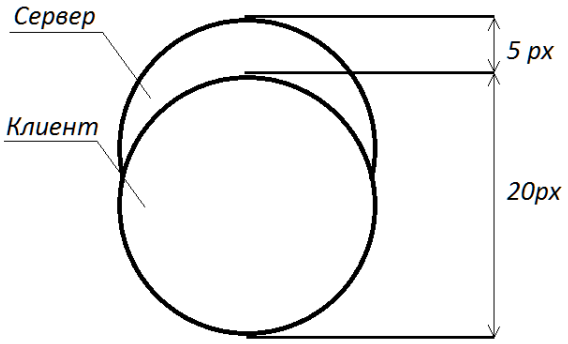

Рис. 4. Положение спрайтов на сервере и клиенте

Каждый спрайт имеет диаметр 20 пикселей. При этом разница в положении спрайтов на устройстве-сервере и устройстве-клиенте равна примерно четверти от размера самого спрайта, то есть 5 пикселей. В листинге 6 представлен фрагмент кода программы, отвечающий за перемещение спрайтов по экрану.

Листинг 6:

```
time1 =  setInterval(function(){
            $(".sprite").each(function(){
                    var x = parseInt($(this).css("left").split("px"));
                    var y = parseInt($(this).css("top").split("px"));
                    $(this).css({"top":(y + Math.round(Math.sin((parseInt( $(this).attr("alt"))) * 2 *
Math.PI / 360) * 5)), "left":(x + Math.round(Math.cos((parseInt( $(this).attr("alt"))) * 2 * Math.PI / 360) * 5))});
                    … }, 50);
```

Из листинга 6 видно, что изменение положения спрайта по осям X и Y происходит так, чтобы суммарный путь, пройденный спрайтом, был равен 5 пикселям (гипотенуза). Именно столько составляет разница между положениями спрайтов на двух устройствах. Исходя из того, что данная функция вызывается раз в 50 мс, можно сделать оценку задержки между двумя устройствами, которая также будет составлять порядка 50 мс.

Другой способ оценки задержки основывается на пропускной способности канала связи. В данном случае используется симметричный метод передачи, который, как было сказано в разделе 1, позволяет передавать данные в обоих направлениях со скоростью до 432,6 Кбит/сек.

$$432{,}6\ \text{Кбит/с} = 54{,}075\ \text{Кб/с} = 55372{,}8\ \text{байт/с} \quad (1)$$

Из выражения (1) видно, что по такому каналу можно передать примерно 55,4 байта информации за одну миллисекунду. Было установлено, что средний размер передаваемого через сокет сообщения составляет 2,5 Кб, исходя из чего можно оценить время, необходимое для передачи, которое и является задержкой между двумя устройствами:

$$\frac{2500\ \text{б}}{55{,}4\ \text{б/мс}} = 45{,}1\ \text{мс} \quad (2)$$

Из выражения (2) видно, что полученная оценка задержки практически совпадает с предыдущей и составляет около 50 мс.

Заключение

Был разработан работоспособный алгоритм передачи сообщений между двумя носимыми устройствами Samsung Gear S2 по Bluetooth-каналу по протоколу RFCOMM с использованием сокетов. Данный алгоритм был сразу же использован при создании тестового приложения, представляющего собой игру с режимом для двух игроков.

Проведенные испытания показали практическую применимость разработанного алгоритма для решения задач, подобных рассмотренной, однако остается проблема задержки пересылаемых сообщений. Ограниченность API и базовой функциональности данной модели смарт-часов не позволяют найти более оптимальное решение этой проблемы. Следует заметить, что задержка составляет порядка 50 мс, что не оказывает существенного влияния на функционирование приложения, в котором использовался разработанный алгоритм.

Список литературы

[1]. Weeks R., Dumbill E., Jerson B. Linux Unwired (Chapter 7 - “Bluetooth”): O’Reilly. 2004. 312 с.

[2]. Таненбаум Э., Уэзеролл Д. Компьютерные сети. Учебное пособие. 5-е изд. СПб.: Питер. 2012. 960 с.

[3]. Бройдо В.Л., Ильина О.П. Вычислительные системы, сети и телекоммуникации. 4-е изд. СПб.: Питер. 2011. 560 с.

[4]. Орлов С.А., Цилькер Б.Я. Организация ЭВМ и систем. Учебник для вузов. 2-е изд. СПб.: Питер. 2011. 688 с.

[5]. Официальный сайт Bluetooth / Электронный ресурс. Режим доступа: https://www.bluetooth.com/ (Дата обращения: 09.10.2017).

[6]. API операционной системы tizen / Электронный ресурс. Режим доступа: https://developer.tizen.org/node/18032 (Дата обращения: 09.10.2017).

[7]. Татарников О. Bluetooth и безопасность / Электронный ресурс. Режим доступа: http://compress.ru/article.aspx?id=10807 (Дата обращения: 09.10.2017).

[8]. Bluetooth / Электронный ресурс. Режим доступа: http://www.tadviser.ru/index.php/Статья:Bluetooth (Дата обращения: 08.10.2017).