Сервисы сетевой близости на базе беспроводных сетей

Горлов А.О.

***Аннотация*—В статье рассматриваются вопросы разработки сервисов, использующих сетевую близость в средах беспроводной передачи данных для решения актуальных задач. В первой части статьи дается толкование понятию сетевой близости, описываются ее свойства. Во второй части рассматриваются существующие подходы к моделированию систем, использующих свойства сетевой близости, даются понятия, используемые в этой области, обозначаются роли элементов существующих моделей. В остальной части статьи описывается предлагаемая в статье архитектура сервисов, использующих свойства сетевой близости на базе беспроводных сетей, на примере технологии Bluetooth.**

***Ключевые слова*—Network proximity, Bluetooth, iBeacon, Push-notifications.**

I. Введение

В последнее время набирают популярность сетевые технологии, которые используют свойство локальности (в пространственном значении), или так называемой сетевой близости, в беспроводных сетях. К таким технологиям относится, прежде всего, технология iBeacon, выпущенная не так давно компанией Apple, а также Google Nearby[11] и Samsung Placedge[10]. Все они используют в своей основе технологию передачи данных Bluetooth Low Energy, наиболее существенным достоинством которой отмечаются ее сверхнизкие требования к энергопотреблению. Многие устройства, выполненные по спецификации Bluetooth Low Energy (Bluetooth LE), могут работать на одной батарейке в течение года без необходимости подзарядки [9]. Низкое энергопотребление не единственное достоинство, которое обеспечивает растущую популярность устройствам, созданным с использованием Bluetooth LE. Так, со схожими требованиями к питанию существовали технологии, используемые в построении сенсорных сетей, и до появления Bluetooth LE, например, ZigBee [7]. Важным преимуществом, служащим сегодня распространению технологий, основанных на Bluetooth LE, является повсеместное использование сетевого стека Bluetooth: модулями Bluetooth оснащены сегодня мобильные устройства, компьютеры, различные устройства ввода, манипуляторы, автомобили и т.д. Вместе с тем стоит отметить, что традиционный (классический) Bluetooth всё еще является более популярным в сравнении с Bluetooth LE. И как будет описано далее, предлагаемая в статье архитектура сервисов, использующих свойства сетевой близости, также в качестве демонстрационного примера применяет привычный стек протоколов Bluetooth, что, впрочем, никоим образом ее не ограничивает.

С помощью технологий, использующих свойство сетевой близости, решают сегодня различные классы задач:

1. Опреление местоположения внутри помещений;
2. Предоставление дополнительной, зависящей от местоположения информации клиентам магазинов, посетителям музеев, выставок и т.д.;
3. Использование в розничной торговле для размещения рекламы, акций и купонов[5];
4. Информирование в системах общественного транспорта[8];
5. Аутентификация клиентов, предоставление доступа к объектам, либо к управлению ими[4];
6. Различные задачи для решений "умного города";
7. Системы платежей в местах оплаты, не требующих платежных картах и наличных средств, и др. [4]

II. Сетевая близость

Сетевой близостью можно назвать свойство беспроводных сетей, позволяющее некоторым образом определить отношение расстояния между узлами сети. За такое отношение часто принимают бинарную функцию, которая позволяет определить "видимость" одного устройства другим, т.е. устройство беспроводной сети способно определить, находится ли другое устройство в некоторой его окрестности или нет. В данном случае под окрестностью, или диапазоном видимости, строго говоря, понимают не геометрическую окрестность, а окрестность устройства, в которой устройство способно демодулировать и декодировать пакет, посланный некоторым трансмиттером (передатчиком). Таким образом, предикат, определяющий отношение сетевой близости, возвращает 1, если устройство может получить пакет, отправленный другим устройством, и 0 в ином случае[3].

Важно отметить, что свойство сетевой близости доступно независимо от конкретной технологии построения беспроводных сетей. Это справедливо и для Bluetooth. Также при определении "видимости" одного устройства другим часто интересуются другим свойством, уровнем сигнала (RSSI)[3]. Это свойство позволяет помимо определения близости устройств узнать степень этой близости. На точность определения степени близости устройств влияет множество факторов, в т.ч. и случайных, которые стараются учитывать в системах позиционирования внутри помещений (IPS)[3], [2].

Степень близости, использующую RSSI источника

Статья получена 27 апреля 2015.

А.О. Горлов — студент магистратуры факультета ВМК МГУ им. М.В. Ломоносова (email: srvisor@gmail.com)

сообщения, можно рассматривать как другой способ определения отношения расстояния между узлами беспроводной сети. Функция степени близости уже не является бинарной, и в качестве значения может возвращать, например, величины в децибелах, относительные величины и т.д. Чаще всего используются децибелы, т.к. именно в этих величинах значение уровня сигнала чаще всего доступно принимающему устройству.

В IPS такое определение близости устройств применяют для позиционирования устройств, используя методы триангуляции, трилатерации и т.п.[2] При решении многих задач толкование свойства сетевой близости, как расстояния между устройствами, не требуется, например, такими задачами являются все из ранее перечисленных кроме первой.

Далее рассматриваются оба варианта определения близости, и как бинарной функции, и как функции расстояния, однако стоит уточнить, что не всегда возможно определить уровень сигнала, или это возможно только при установлении соединения между устройствами[6]. Не теряя общности можно использовать понятие сетевой близости, уточняя по необходимости, когда речь идет о функции расстояния.

Сетевая близость характеризуется рядом свойств:

1. Работоспособность в помещениях;
2. Прецизионность;
3. Мобильность.

1. Работоспособность в помещениях.

Данное свойство хорошо применимо в IPS. В то время, как системы глобального позиционирования, такие как GPS, в зданиях не работают или работают очень плохо, сетевая близость удачно используется внутри помещений для позиционирования, при этом она не требует длительного поиска спутников связи или стационарных вышек, сложного оборудования и т.д. Для работы IPS, используя только сетевую близость, необходимо иметь достаточно плотную сеть устройств, поддерживающих выбранные технологии беспроводных сетей. Плотность сети, т.е. насколько часто встречаются устройства по пути движения, или насколько много их находится в окрестности устройства, зависит от характеристик выбранной технологии беспроводных сетей и объясняется алгоритмами, вычисляющими положение устройства относительно устройств в его окрестности. Такие алгоритмы требуют, по меньшей мере, трех устройств в границах видимости. Чем больше видимых устройств, тем выше, обычно, точность позиционирования. Для подобных сервисов требуются также карты помещений, которые необходимо предварительно загрузить на устройство, а также метаданные о местоположении на карте известных устройств.

2. Прецизионность.

Сетевая близость позволяет локализовать объекты, которые находятся в непосредственной близости, при этом можно быть уверенным в точности данных при использовании функции расстояния, т.к. локальность сетевой близости обеспечивает известные возможности, зависящие от характеристик выбранной беспроводной технологии передачи данных. Важным качеством является и то, что сетевая близость позволяет локализовать объекты логически, в данном случае точность информации о видимости устройств, скорее, несет семантическую нагрузку, а не географическую, пространственную. С помощью сетевой близости можно говорить об указании местоположения с точностью до объекта локализации, это может быть этаж, офис, комната и т.д. В частности, сетевая близость может указывать и на видимость некоторого заранее известного объекта, оснащенного передатчиком, даже если он находится не внутри какого-либо помещения, например, автомобиля (современные автомобили оснащаются модуля Bluetooth).

3. Мобильность.

Данное свойство позволяет с помощью сетевой близости устанавливать отношения между нестационарными устройствами. В отличие от систем глобального позиционирования, в котором для удовлетворения указанного требования необходимо обладать знаниями о географических координатах всех устройств, отношение видимости между которыми потребовалось бы установить, сетевая близость позволяет указать, какие устройства находятся поблизости друг от друга, не зная их абсолютного размещения, не имея сведений об их географических координатах. Таким образом, даже, если оба устройства фактически не обладают знаниями о своих географических координатах, они могут получить сведения о ближайших устройствах даже в тех случаях, когда они двигаются, но расстояние между ними не превышает возможностей выбранной технологии беспроводной связи. Примером, когда это свойство актуально, может являться подвижной состав электропоезда, оснащенный Bluetooth модулем, в котором находится пассажир с Bluetooth-устройством. В данном случае можно также отметить и возможность указать расположение устройства с точностью до вагона. Подобные возможности хорошо пригодны в информировании в системах общественного транспорта.

Таким образом, можно отметить, что сетевая близость позволяет локализовывать объекты с точностью выбранной беспроводной технологии передачи данных, т.е. выбирая конкретную технологию можно контролировать варьировать свойство локальности сетевой близости.

III. Существующие подходы

A. Термины.

В существующих на данный момент системах, использующих свойство сетевой близости, устоялись некоторые решения, за элементами которых закреплены определенные роли. Так устройства, которые опрашивают другие устройства в беспроводной среде, осуществляя поиск, называют считывающими устройствами, или сканерами, те устройства, которые вещают, - транспондерами, передатчиками. Передатчики, которые выступают в роли меток в

беспроводной среде, например, которые передают свой MAC-адрес и два числа Major и Minor, в технологии iBeacon, также часто именуются тегами, или метками. Процесс поиска устройств сканером называют обнаружением устройств.

B. Считывающие устройства.

Такими устройствами могут быть любые устройства, оснащенные модулями беспроводной связи, например, модулями Bluetooth. Ими могут быть компьютеры, планшеты, но чаще всего это коммуникаторы. На коммуникатор предварительно требуется установить специальную программу, осуществляющую обнаружение меток.

C. Метки.

Например, в технологии iBeacon, имеющая сегодня наибольшую популярность из тех, которые используют Bluetooth LE, метки вещают с определенной частотой пакет, содержащий:

1. iBeacon префикс, содержащий информацию, что beacon принадлежит протоколу iBeacon;
2. UUID идентификатор - идентификатор, который позволяет отделить beacon-ы одной компании от beacon-ов другой;
3. Major - идентификатор, который может идентифицировать группу beacon-ов, например, принадлежащих некоторому магазину компании, другой магазин тогда будет иметь другой идентификатор Major.
4. Minor - идентификатор, который может идентифицировать beacon в одной группе;
5. Measured Power - уровень сигнала метки, измеренный на расстоянии одного метра от нее;

При этом частота широковещательных сообщений может быть изменена, и часто составляет от 50 мс до 2 с.

D. Процесс обнаружения.

Этот процесс запускается сканирующим устройством, которое собирает обнаруженные метки.

Сделаем ряд важных замечаний о технологии iBeacon. Подход технологии Apple практически целиком перенимается реализациями других производителей, так что отдельных пояснений они не требуют.

1. Стоит отметить, что метки iBeacon требуют установленного регламента назначения уникальных идентификаторов компаний, но что особенно важно, так это то, что метки нуждаются в поддержке специального протокола;
2. Метки реализуются в качестве специальных устройств, которым требуется монтаж;
3. Наиболее важным, пожалуй, является необходимость устанавливать специальное программное обеспечение на считывающее устройство, причем специальное в данном случае означает, что это программное обеспечение должно быть своё для каждой компании, которая использует технологию iBeacon.

Все эти пукнты можно считать недостатками технологии iBeacon и аналогичных ей, поэтому далее будет рассматриваться подход, предложенный в проекте Bluetooth Data Points (BDP), изложенный в [1].

Отметим также важные отличия идей, изложенных в BDP, от технологии iBeacon:

1. Отсутствие необходимости в установке специального приложения для каждого вендора: для работы требуется только одно приложение, которое умеет читать все метки и получать по ним необходимую информацию;
2. В BDP используются метки, не требующие специального протокола: в качестве меток используются обычные пакеты идентикации устройства в стандартном протоколе Bluetooth;
3. В качестве транспондеров в BDP можно использовать любое устройство, оснащенное модулем Bluetooth, причем создать метку и запустить ее в работу можно программным путем, что позволяют сделать современные мобильные операционные системы, достаточно включить Bluetooth устройство в режиме discoverable.
4. Сама метка в BDP данных не передает, данные о распространяемом с помощью меток содержимом хранятся на удаленном сервере, к которому сканирующее устройство самостоятельно получает доступ.

Далее предлагаемая в статье архитектура сервиса использует в качестве своей основы идеи, заложенные в BDP, так что дальнейшее описание модели сервисов, основанных на сетевой близости, можно считать развитием модели BDP.

IV. Архитектура сервисов, основанных на сетевой близости.

В архитектуре сервиса, использующего сетевую близость на базе беспроводных сетей, выделим несколько крупных элементов:

1. Сканирующее устройство;
2. Метки;
3. Удаленный сервис;
4. Хранилище.

Проведем описание элементов архитектуры и их возможной реализации на основе стандартного Bluetooth. Общности архитектура не потеряет, и при небольших изменениях, всё то же справедливо и для реализаций, использующих Bluetooth LE или WiFi.

1. Сканирующее устройство.

На сканирующее устройство устанавливается приложение, основными задачами которого являются:

1. Сканирование беспроводной среды на предмет обнаружения меток;
2. Формирование из меток и контекстной информации слепков;
2. Передача слепков удаленному сервису;
3. Подписка на push-уведомления;
4. Отображение уведомлений;
5. Совершение некоторых действий в ответ на уведомления;
6. Активация/деактивация обнаружения и обработки

уведомлений;

7. Добавление правил отображения информации на удаленный сервер, если само сканирующее устройство выступает в роли метки.

Приложение сканирования состоит из двух частей. Одна часть - визуальная, отображение настроек приложения (настроек контроля сканирования), отображение объявлений, просмотр обнаруженных слепков, выполнения действий, переданных в данных push-уведомления и т.п. Другая часть - собственно сканирующий сервис, запущенный в фоновом режиме, не имеет визуальной составляющей, однако предоставляет доступ к своему состоянию и настройкам через область уведомлений. Сканирующий фоновый сервис запускается при старте приложения, и работает, даже если приложение завершить, при этом может запустить визуальную часть приложения для контроля состояния сканирования. При старте сервиса проверяется, включен ли Bluetooth-адаптер, и если не включен, включается, далее запускается процесс обнаружения меток. По завершении процесса обнаружения меток, они собираются вместе с некоторой контекстной информацией в слепок и отправляются удаленному сервису. Режим работы сервиса настраивается, но общая логика его заключается в том, что он запускает сканирование раз в какой-то установленный промежуток времени, по завершении которого уходит в ожидание на определенное время. Процесс сканирования в стандартном протоколе Bluetooth может занимать около 12 секунд, поэтому можно произвести настройку сервиса для принудительного завершения сканирования до истечения 12 секунд. Это может осуществляться, например, с целью экономии заряда батареи, т.к. после завершения сканирования сервис выключает Bluetooth-устройство. Еще одной причиной принудительного завершения сканирования можно отметить желание сократить формирования слепка для более быстрой отработки общего цикла доставки объявлений.

2. Метки.

Метками, как и в BDP, в предлагаемой архитектуре являются обычные Bluetooth-устройства, на которых может запускаться приложение, переводящее устройство в режим discoverable, т.е. в режим доступного для обнаружения. Устройство может быть переведено в указанный режим и иным путем, если есть такая возможность, без установки на него соответствующего приложения. Больше от устройства ничего не требуется, никаких данных специального назначения оно не передает. Основная его задача - быть доступным.

3. Удаленный сервис.

Сервис реализуется на основе архитектуры REST, используя в качестве тела HTTP-сообщений JSON-объекты, и выполняет несколько задач, среди которых (в скобках указаны endpoint-ы сервиса, соответствующие рассматриваемым запросам):

1. Регистрация в сервисе доставки push-уведомлений (выполняется при запуске, без необходимости совершения запроса извне);
2. Регистрация слепков (запрос registerFingerprint);
3. Поиск правил по слепку (внутренняя работа сервиса, выполняемая на запрос регистрации слепка);
4. Формирование содержимого, используя найденные правила (поиск данных в хранилище по правилам);
5. Отправка push-уведомлений с содержимым (запрос на сторонний сервер доставки push-уведомлений);
6. Ведение статистики (сохранение в хранилище данных по запросам);
7. Добавление правил (запрос addRule).
8. Регистрация точек (запрос registerPoint).

В качестве платформы для разворачивания сервиса удобно использовать PaaS.

4. Хранилище.

Хранилище использует реляционную систему управления базами данных и хранит:

1. Присланные слепки со временем их прихода;
2. Правила;
3. Метки;
4. Данные запросов;
5. Объявления.

Опишем, как выглядит полный жизненный цикл предлагаемого сервиса.

Сначала заводятся метки, они регистрируются в сервисе с помощью отправки по endpoint-у *registerPoint* с передачей в качестве параметра данных метки. Регистрацию метки может производить клиентское приложение, которое выступает в роли самой метки, другое клиентское приложение, если пользователем предоставлены данные о метке, а также с помощью веб-интерфейса доступа к сервису. На сканирующее устройство устанавливается приложение для сканирования меток. После установки запускается процесс сканирования описанным выше образом, по завершении сканирования, проверяется, получены ли необходимые данные для формирования слепка. Если данные не получены, сканирование откладывается на установленный таймаут и повторяется снова по его прошествию. Если данные получены, формируется слепок, далее он отправляется сервису по endpoint-у *registerFingerprint* с передачей данных слепка. Сервис обрабатывает слепок и производит поиск правил, которым он удовлетворяет. Если такие правила найдены не были, на этом работа сервиса по обработке текущего запроса завершается сохранением данных по запросу. Если соответствующие правила обнаружились, по ним выполняется формирование содержимого ответа, поиск объявления и отправка данных по нему push-уведомлением на сервис доставки с последующим сохранением данных по запросу. Далее сервис доставки push-уведомлений доставляет уведомления на сканирующее устройство, обработчик, подписанный на уведомления, вызывается мобильной операционной системой, и управление передается сканирующей

программе, которая добавляет локальные уведомления в область уведомлений с содержимым, пришедшим в push-уведомлении. Если по правилам сервис сформировал несколько уведомлений, то несколько уведомлений будет доставлено и сканирующему устройству. И задачей сканирующей программы уже является возможная группировка полученных уведомлений.

Более подробно параметры запросов, отправляемые объекты и логика описаны далее.

V. Объявления

Объявления - это некоторый текст, который сервис должен отправить сканирующему устройству в ответ на присланный слепок в случае, если нашлись правила, ему удовлетворяющие. В качестве объявления может быть и обычный текст, текст с форматированием, HTML-страница, URL и т.п. В простейшем случае, как это рассмотрено в BDP, сканирующее приложение является обычным веб-браузером, т.е. WebView встроенным в приложение, в которое загружается URL, присланный сервисом. Но наиболее универсальным объявление может быть, если в нем будет указан тип данных, содержащихся в его теле. В таком случае не нужно выбирать некоторый общий тип данных, а можно добавлять поддержку новых типов данных в сканирующее приложение при условии, что формат данных, присланных в объявлении, не меняется. Таким образом, можно задавать не только текст, графику, ресурс на удаленном сервере и т.д., но и более сложные данные, а также действия, которые может поддерживать сканирующее приложение. Одним из примеров таких типов могут быть географические координаты с текстом, которые необходимо отобразить на карте. Современные мобильные операционные системы позволяют одному приложению запускать другие приложения, причем часто не только стандартные, но сторонние, установленные и не являющиеся часть системы стандартной поставки. URL в таком случае разумно было бы открывать в стандартном браузере, точно также, как и графические файлы, или звуковые файлы.

Для определения такого поведения, необходим объект объявления, который сможет в себе инкапсулировать такую логику.

Таким образом, объект объявления должен в себе содержать:

1. Тип данных.
2. Сами данные.
3. Действие, которое необходимо над ними совершить.

В JSON, отправляемом с сервиса, пример такого объявления может выглядеть следующим образом:

```
{
  "type": "geographic-coordinates",
  "data": { "latitude": 55.4506, "longitude": 37.3704 },
  "action": "show-on-map"
}
```

При приеме нескольких уведомлений сканирующее приложение должно быть способно группировать их в области уведомлений. Для группировки можно использовать категорию объявления, таким образом, к представленному выше формату можно добавить категорию. В таком случае представленное выше объявление может выглядеть так:

```
{
  "type": "geographic-coordinates",
  "data": { "latitude": 55.4506, "longitude": 37.3704 },
  "action": "show-on-map",
  "category": "trade"
}
```

Семантика за категориями может закрепляться любая, но разумно предположить, что это будут категории доступных услуг.

В хранилище такое же объявление может выглядеть так:

Announces (primary key id, creation_time, activation_start, activation_period, data, type, action, category), где:

1. id - идентификатор объявления;
2. creation_time - время создания объявления;
3. activation_start - время, с которого объявление считается активным;
4. activation_period - период, в течение которого с начала активации объявление считается активным;
5. data - данные;
6. type - тип данных;
7. action - действие над данными;
8. category - категория объявления.

Следует отметить, что активация объявлений может контролироваться не только сервисом, но и сканирующим приложением, когда сканирующее приложение (посредством настройки) может выбирать, например, насколько старые объявления оно будет отображать, независимо от того, активно оно до сих пор или нет. В таком случае, по образцу с категорией в отправляемом JSON можно указывать и другие данные, представленные в хранилище, например, дату создания объявления или дату, с которой объявление считается активированным и период его активности.

VI. Метки и слепки.

В отличие от BDP в предлагаемой архитектуре также вместо просто меток используются слепки. Слепки - это объекты, которые собираются по завершении процесса сканирования меток, состоящие из следующего набора данных:

1. Список меток;
2. Время создания слепка;
3. Тип сети (для описания мы приняли, что тип сети Bluetooth);
4. MAC-адрес Bluetooth-адаптера сканирующего устройства;

5. RSSI (опционально, в случае, если используется Bluetooth LE);
6. Комментарий (опционально, простой текст, описывающий место, в котором слепок был сформирован, задается пользователем по требованию сканирующего приложения);
7. Географические координаты (опционально, в случае, если сервис гео-позиционирования включен, работает и может предоставить координаты);
8. Идентификатор сканирующего устройства для отправки push-уведомления.

Таким, например, может быть JSON, который отправляется удаленному сервису в качестве тела HTTP-пакета на endpoint *registerFingerprint*:

```
{
  "macAddress": "34:FC:2F:42:59:4F",
  "networkType": "Bluetooth",
  "points": [ { "deviceAddress": "7C:31:93:C9:37:CA", "deviceClass": "5a020c", "deviceName": "HTC Device" } ],
  "time": 1430017732769,
  "registrationId": "APA91bHun4MxP5e..."
}
```

Здесь points - это список меток. Метки содержат следующие данные:

1. MAC-адрес;
2. Класс устройства;
3. Имя устройства.

Точки могут быть соответственно зарегистрированы с помощью отправки на endpoint сервиса registerPoint JSON-сообщения следующего содержания:

```
{
  "deviceAddress": "7C:31:93:C9:37:CA",
  "deviceClass": "5a020c",
  "deviceName": "HTC Device"
}
```

Преимуществом использования слепков вместо отдельных меток можно считать возможность установить, что они обнаружены совместно. Кроме того, сам слепок может нести дополнительную информацию, которая не описывается собственно метками, например, контекстную информацию. В данном случае, опциональными указаны комментарий, координаты и т.д. Потенциально же, как развитие идеи, можно ожидать добавление в слепок информации, например, об интенсивности света, окружающих звуках и т.д.

В хранилище данные сущности могут выглядеть, например, так:

Fingerprints(primary key id, mac_address, network_type, time), где

1. id - идентификатор слепка;
2. mac_address - MAC-адрес сканирующего устройства;
3. network_type - тип сети;
4. time - время создания слепка.

Как видно, не хранится *registrationId*, идентификатор для отправки push-уведомлений, хранить его не имеет смысла, т.к. это транзиентные данные, нужны только для отправки уведомлений, да и устройство всё равно должно его отправлять каждый раз при отправке слепка. Связь слепка и меток также не задается в отношении слепка, т.к. для этого создается специальное отношение:

Fingerprints_Points(fingerprint_id, point_id), где

1. fingerprint_id - идентификатор слепка;
2. point_id - идентификатор метки.

Для меток создается следующее отношение:

Points(primary key id, mac_address, device_class, device_name), где

1. id - идентификатор метки;
2. mac_address - MAC-адрес метки;
3. device_class - класс устройства;
4. device_name - имя устройства.

Атрибуты *mac_address, device_class u device_name* должны быть ограничены на уникальность, чтобы обеспечить свойство уникальности добавляемых меток.

VII. Правила

В простейшем случае правило - это связка метки и объявления, и условие его выполнения - наличие метки в слепке. Если метка в слепке присутствует, соответствующее объявление отправляется сканирующему устройству, идентифицированному registrationId. Сколько метод в слепке обнаружено, для которых заданы правила, столько уведомлений сервис отправляет.

Отношение, представляющее правило может выглядеть, как:

Rules(primary key id, point_id, announce_id) где

1. id - идентификатор правила;
2. point_id - идентификатор метки;
3. announce_id - идентификатор объявления.

Пара point_id и announce_id должна быть с ограничением на уникальность для того, чтобы правила не дублировались по существу.

Надо заметить, что важным свойством является возможность привязать к одной точке несколько объявлений.

Для такой простой модели правил лучше подходят другие типы баз данных, например, Key-Value базы данных или RDF[12]. Однако, если предположить, что правила могут задавать несколько более сложным способом, реляционная модель может оказаться более эффективной. Так мы можем добавить некоторые операции над правилами, такие как AND и OR, а сами правила формулировать в виде простых предикатов, например POINT_IS_VISIBLE, тогда объединяя несколько правил с помощью указанных операций и сформулированных предикатов, мы можем создавать

более сложные правила из простых. В данном случае подходящая модель базы данных является предметом исследования, т.к. такие правила фактически образуют выражение, которое хорошо представимо в виде дерева, внутренними вершинами которого являются операции, а листьями - данные из полученного слепка.

VIII. Заключение

В первой части статьи было рассмотрено понятие сетевой близости, ее свойства, описаны преимущества, доступные сервисам, использующим их. Далее были описаны технологии, использующие Bluetooth LE, рассмаотрены их ограничения, указаны недостатки. В остальной части работы предложена архитектура и подход к построению сервисов, основанных на сетевой близости, в основу которого в качестве более гибкой альтернативы популярному решению выбрана модель BDP. В работе были предложены также решение, которые можно считать развитием идей, изложенных в BDP, описаны возможные подходы к реализации сервисов, использующих свойства сетевой близости. Для такой реализации достаточно обычных мобильных телефонов или устройств с модулями беспроводной связи небольшого радиуса действия, а также применения распространенных технологий. При этом описанная архитектура не ограничена и имеет несколько точек роста, дающих возможность продолжать исследования в этой области и развивать имеющуюся модель.

IX. Примечание редактора

В магистерской диссертации автора, а также в данной статье, написанной на ее основе, рассматривается развитие модели Bluetooth Data Points [3, 13]. Основная идея, как указывается автором, состоит в том, что при использовании модели “мобильный телефон – как сенсор” (что есть вполне естественный подход, учитывая распространение мобильных устройств) датчики беспроводных сетей (Bluetooth, Wi-Fi) оказываются самыми распространенными “сенсорами”, присутствуя в каждом мобильном телефоне. К тому же, их измерения полностью стандартизованы. Соответственно, информация о сетевом окружении становится одной из наболее легко достижимых при описании контекста. Этим и объясняется факт использования информации о сетевом окружении (сетевой близости) для замены географических координат в предлагаемой модели контекстно-ориентированных сервисов. Данные (информация) привязывается не к координатам, а к доступным (в данный момент времени) сетевым узлам. Разница с классическим подходом (данные привязаны к географическим координатам) наиболее полно проявляется в том случае, когда сетевые узлы, определяющие доступность данных, сами перемещаются.

Достоинством данной работы является рассмотрение работы такой модели при автоматическом (фоновом) отслеживании изменений в сетевом окружении. Другим важным моментом является открытый код созданных компонент, что позволит расширять и модифицировать авторскую модель в рамках работ по данной тематике, проводимых в лаборатории Открытых Информационных Технологий ВМК МГУ имени М.В. Ломоносова [14].

Библиография

[1] Намиот Д. Е. Мобильные Bluetooth теги //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 5. – С. 17-23.

[2] Namiot D. On Indoor Positioning //International Journal of Open Information Technologies. – 2015. – Т. 3. – №. 3. – С. 23-26.

[3] Neal Patwari and Alfred O. Hero, Using Proximity and Quantized RSS for Sensor Localization in Wireless Networks, Univercity of Michigan, Dept. of Electrical Engineering and Computer Science, CA, USA, Sep. 2003.

[4] Agusti Corbaco Salas, Indoor Positioning System based on Bluetooth Low Energy, Universitat Politechnica De Catalunya, Barcelonatech, Barcelona, Jun. 2014.

[5] Bianca Deordica and Marian Alexandru, Advertisement using Bluetooth Low Energy, Transilvania Univercity, Brasov, Romania, Review of the Air Force Academy, No 2 (26) 2014.

[6] Android Bluetooth Gatt https://developer.android.com/reference/android/bluetooth/BluetoothGatt.html Retrieved: Apr, 2015

[7] ZigBee https://ru.wikipedia.org/wiki/ZigBee Retrieved: Apr, 2015

[8] SITA Common Use Beacon Registry https://www.developer.aero/Beacon-Registry-API/API-Overview Retrieved: Apr, 2015

[9] Bluetooth Low Energy Retrieved: https://en.wikipedia.org/wiki/Bluetooth_low_energy Apr, 2015

[10] Samsung Placedge Retrieved: https://placedge.samsung.com/ Apr, 2015

[11] Google Nearby https://developers.google.com/games/services/android/nearby Retrieved: Apr, 2015

[12] Resource Description Framework https://ru.wikipedia.org/wiki/Resource_Description_Framework Retrieved: Apr, 2015

[13] Namiot D., Sneps-Sneppe M. On Mobile Bluetooth Tags //arXiv preprint arXiv:1502.05321. – 2015.

[14] Гурьев Д. Е., Намиот Д. Е., Шнепс М. А. О телекоммуникационных сервисах //International Journal of Open Information Technologies. – 2014. – Т. 2. – №. 4. – С. 13-17.

Network proximity services on wireless networks

A.O. Gorlov

Abstact — the article deals with the development of services that use network proximity in wireless networks to solve actual problems. The first part of the article reveals the concept of network proximity and describes its properties. The second part of the article discusses existing approaches to modeling systems, using the properties of the network proximity. It gives the concepts commonly used in this area, and describes the role of the elements of the existing models.The rest of the article offers architecture of the services using properties of the network proximity based on wireless networks. This architecture is described by the example in which the Bluetooth technology is chosen as wireless protocol.

Keywords — network proximity, Bluetooth, iBeacon, push-notifications.