# О модели данных для контекстно-зависимых сервисов

© Д.Е. Намиот © М.А. Шнепс-Шнеппе
Московский Государственный Университет имени М.В. Ломоносова, ЦНИИС
Москва
dnamiot@gmail.com sneps@mail.ru

## Аннотация

В данной работе рассматриваются вопросы представления данных для одной модели контекстно-зависимых сервисов. В этой модели в качестве основной информации о контексте используются данные о беспроводных сетях, существующих в конкретный момент времени в точке присутствия мобильного терминала (мобильного телефона). Сети могут быть представлены точками доступа Wi-Fi, узлами Bluetooth и Bluetooth тегами. Сервисы для такого рода модели состоят в предоставлении мобильным абонентам доступа к определенной информации, в зависимости от доступности (наличия) беспроводных сетей. Другой вариант – публикация мобильными пользователями контекстно-зависимой информации, связанной с беспроводными сетями. Вопросы представления данных и управления ими для такого рода сервисов и служат предметом рассмотрения данной работы.

## 1 Контекстно-зависимые сервисы

Согласно классическому определению, контекст представляет собой произвольные характеристики, которые могут быть добавлены к местоположению.

В работе, которая впервые ввела термин ‘контекстно-зависимый’ (‘context-aware’) применительно к программному обеспечению и сервисам, авторы [1] определили контекст как местоположение, идентификацию окружающих людей и предметов, а также изменения в указанном окружении. Другое определение [2], описывает контекст как произвольную информацию, которая может характеризовать ситуацию или сущность. Сущность здесь есть персона, место или объект, существенный для взаимодействия между пользователем и приложением, включая само приложение и его пользователя. Последнее определение более “алгоритмическое”, поскольку допускает прямое перечисление сущностей для конкретного сценария в приложении. Это определение наиболее соответствует тому, которому мы следуем в данной работе.

**Труды XVII Международной конференции DAMDID/RCDL’2015 «Аналитика и управление данными в областях с интенсивным использованием данных», Обнинск, 13-16 октября 2015**

Согласно [3], контекстно-зависимая информация в мобильных приложениях содержит пользовательские профили и предпочтения, местоположения пользователей, характеристики соединения с мобильной сетью, характеристики используемых мобильных устройств, объекты, находящиеся поблизости, а также информацию о поведении (историю).

Традиционно, рассматриваются следующие модели применения (использования) контекстной информации в приложениях:

a) обмен информацией между пользователями (пользователями и приложениями) с учетом контекста,

b) ситуационное реагирование,

c) определение (поиск) подходящей информации в зависимости от контекста [4]

В данной работе мы больше касаемся последнего аспекта – поиска и представления контекстно-зависимой информации.

С практической точки зрения, мобильные приложения (сервисы) всегда контекстно-ориентированы (по крайней мере, должны быть таковыми). Предоставление информации, то есть ее доступность и формат представления, естественным образом зависят от текущего положения мобильного пользователя и условий в которых находится мобильный терминал (телефон).

В практических реализациях, как было указано выше, контекст (то есть, используемые характеристики) должен как-то специфицироваться. Одним из таких решений по конкретизации контекста является использование беспроводных сетей как дополнение (или даже замена) к текущему местоположению мобильного пользователя (пользователей). Причина такого решения вполне очевидна – датчики (контроллеры) беспроводных сетей (Wi-Fi, Bluetooth) присутствуют во всех современных мобильных телефонах и, соответственно, с точки зрения концепции телефон как сенсор [5], именно датчики беспроводных сетей являются самым распространенным сенсором.

Привязка информационного наполнения к текущим измерениям состояния беспроводных сетей означает, что мобильному пользователю становятся видны (доступны) какие-либо информационные объекты в зависимости от текущего состояния беспроводных сетей по месту

его нахождения. В простейшем примере, доступность (видимость) конкретной точки доступа Wi-Fi определяет доступность соответствующего сегмента данных. Смысл состоит в том, что область распространения сигнала Wi-Fi (Bluetooth) ограничена и видимость (доступность) беспроводного узла, таким образом, становится некоторой характеристикой местоположения (“поблизости” от сетевого узла). Особенно такой подход будет актуален для помещений, где доступ к GPS (Global Positioning System - система глобального позиционирования) затруднен.

При этом важно отметить два момента. Во-первых, информация о беспроводных сетях уже используется (может использоваться) производителями мобильных операционных систем (производителями мобильных сервисов) для уточнения местоположения. Так-называемый подход A-GPS (Assisted GPS - технология, ускоряющая «холодный старт» GPS-приёмника за счет предоставления необходимой информации через альтернативные каналы связи) и Wi-Fi позиционирование являются примерами такого использования [6]. Но информация о беспроводных сетях при этом используется как промежуточный этап для получения гео-координат. В этом и состоит отличие от контекстно-зависимых сервисов, которые не связывают информацию о беспроводных сетях с конкретным местоположением. Наиболее ярко это различие демонстрирует мобильная точка доступа (непосредственно) на телефоне. В контекстно-зависимом сервисе информация привязана к видимости (доступности) точки доступа, а не к ее координатам, которые, естественно, изменяются при перемещении телефона. Второе замечание связано с тем, что рассматриваемый подход никак не связан с безопасностью и сетевым обменом (протоколами). Речь идет только о чтении представления сети (сетей), а не о присоединении к ним.

Дальнейшая часть работы структурирована следующим образом. В разделе 2 мы рассматриваем базовые элементы представления сетевой близости. В разделе 3 мы обсуждаем информационные модели (проекты), которые используют (могут использовать) информацию о сетевой близости. В разделе 4 представлена модель ключ-значение для хранения информации для радиокарт.

## 2 Сетевая близость

В настоящем разделе мы остановимся на том, каким образом измеряется сетевая близость, то есть, оценка приближенности мобильного устройства мобильного устройства к объектам беспроводной сети.

Базовое понятие – отпечаток (fingerprint) [7]. Представляет собой вектор, содержащий информацию о видимых узлах беспроводной сети. Для каждого узла (например, точки доступа Wi-Fi) присутствует его идентификация и измеренная сила сигнала. В качестве идентификатора узла выступает так называемый MAC-адрес (от англ. Media Access Control — управление доступом к среде, уникальный идентификатор, присваиваемый каждой единице активного оборудования компьютерных сетей).

Для оценки близости измеренный отпечаток сравнивается с некоторым эталонным значением, или же с другим подобным измерением. Сравнение двух отпечатков $f_1$ and $f_2$, выполняется следующим образом. Обозначим M как объединение MAC-адресов $f_1$ и $f_2$. Для MAC-адреса *m* из множества M $f_1(m)$ есть количество раз, которые адрес m был зарегистрирован в $f_1$, $f_2(m)$ – количество упоминаний в отпечатке $f_2$. Тогда схожесть S для $f_1$ and $f_2$ вычисляется следующим образом:

$$\mathrm{MinMax}(m) = \min(f_1(m),f_2(m))/\max(f_1(m), f_2(m))$$

$$S = \sum_{m \in M} (f_1(m)+f_2(m)) * \mathrm{MinMax}(m)$$

S принимает большие значения, если MAC-адрес встречается часто в обоих отпечатках.

Данный подход не использует силу сигнала. При подключении RSSI (силы сигнала) используется вычисление ближайшей точки (K-среднее) для текущего измерения на мобильном устройстве и предварительно подготовленной сетки “стандартных” измерений [8]. Здесь необходимо отметить, что такой подход, широко используемый в навигации в помещениях, предполагает наличие этих самых “стандартных” измерений. Это является наиболее проблематичным моментом – подготовка такой радио-сетки достаточно трудоемкое дело.

Вместо абсолютных значений силы сигнала беспроводных узлов часто оперируют их относительными позициями. Это позволяет нивелировать вариации в измерении абсолютных значений (значения могут зависеть от положения мобильного телефона, заряда его собственной батареи и т.д.). Например, если в конкретной позиции мобильное устройство обнаружило три видимых точки доступа с соответствующими сигналами (в dB)

$(SS_A; SS_B; SS_C) = (-20; -90; -40)$

мы можем заменить абсолютные значения их относительными рангами

$(R_A; R_B; R_C) = (1; 3; 2)$ [21]

и использовать далее ранговый коэффициент корреляции. [9]

Альтернативный подход используется компанией Apple в iBeacons [10] (рисунок. 1). iBeacons – это Bluetooth тег на базе стандарта Bluetooth Low Energy.

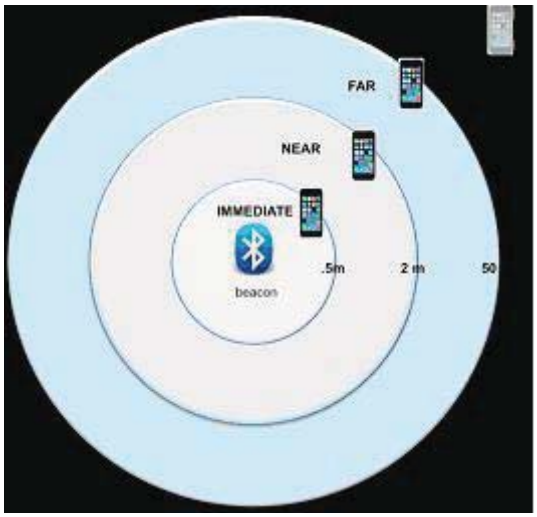


Рис.1 Определение расстояния для iBeacons

Расстояние (близость) мобильного устройства к тегу оценивается по одиночному элементу и эмпирическим правилам для изменения силы сигнала в зависимости от расстояния. При этом система ориентируется на качественную оценку приближения: рядом с тегом, близко к тегу, далеко от тега (удаление больше 2 метров).

## 3 Существующие проекты

В качестве пионерской работы в этой области стоит, видимо, назвать работы Alex Pentland [11] по Reality Mining. В базовых работах использовалась информация о перемещениях мобильного телефона для анализа (раскрытия) процессов в каком-либо ограниченном пространстве. Дальнейшее развитие этого подхода привело к заключению о том, что можно собирать использовать мобильный телефон для сбора информации об окружении.


Рис. 2 Журнал сбора данных в Funf

Пакет Funf [12] позволяет мобильному устройству собирать информацию о доступных сенсорах. Процесс сбора информации (что искать, с какой частотой), естественно, конфигурируется - рисунок 2).

Централизованного хранения собранной информации в данной системе нет. Каждое мобильное устройство собирает данные индивидуально. При необходимости поддержки некоторого общего набора данных необходимо самостоятельно организовывать их интеграцию.

Реальный совместный сбор сетевой информации присутствует в системе OpenSignal [13]. Это проект (мобильное приложение), который собирает единую базу данных о сетевых объектах. Он начался с идеи оценки качества мобильной связи, изначально собирал информацию об имеющихся базовых станциях мобильной сети (характеристика покрытия зависит от наличия базовых станций), в настоящее же время может собирать информацию и о других сетевых узлах (например, точках доступа Wi-Fi ) – рисунок 3. Поддерживается только достаточно ограниченный программный интерфейс (API).

Рис. 3 Карта пакета OpenSignal

Программный интерфейс позволяет запрашивать информацию о мобильных сетях. Явная возможность получать информацию о беспроводных сетях (по крайней мере, в текущей реализации) – отсутствует.

К классической форме представления информации мобильным пользователям в зависимости близости к сетевым устройствам относится упомянутая выше система Bluetooth тегов iBeacon. Мобильное приложение получает идентификатор тега (тег не передает ничего, кроме своего идентификатора) и, в зависимости от этого идентификатора, доставляет определенную информацию пользователю или разрешает выполнение каких-либо действий (рисунок 4).

Однако особенностью этой технологии (по крайней мере, в реализации от Apple), является то, что получателем сообщений от тегов является конкретное приложение. Для конкретного приложения должно быть статически предписано, сигналы от каких тегов данному приложению разрешено получать. Иными словами, статически (при создании приложения), указано, близость к каким сетевым устройствам конкретному приложению разрешено анализировать. В результате, каждое конкретное приложение работает

с конкретным (как правило, достаточно небольшим) набором тегов. Сама операционная система (iOS), работает, естественно, со всеми тегами, но ее интерфейс в части общей базы данных производителем (Apple) не раскрывается.

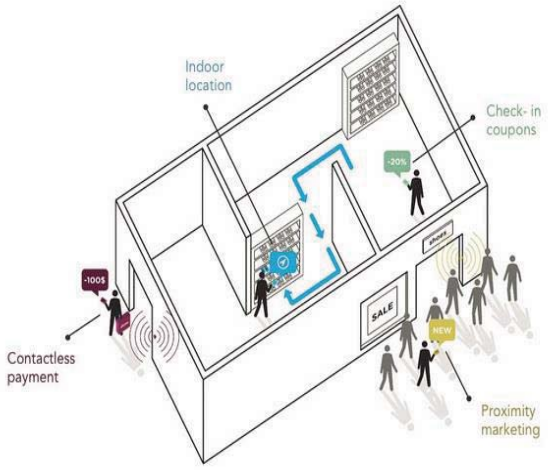


Рис. 4 Модель iBeacons

Вместе с тем, есть примеры “централизованной” расстановки тегов и поддержки единой базы данных для них. Речь идет, например, об общей базе беспроводных устройств на основе Bluetooth Low Energy, установленных в аэропортах [14]. Технические детали собственного построения базы данных не раскрываются, для разработчиков предоставляется REST API, который позволяет получать списки беспроводных тегов в привязке к аэропорту. Сторонние разработчики могут предлагать информационные сервисы для пассажиров в охваченных аэропортах. Мобильное приложение определяет идентификатор тега, поблизости от которого находится пользователь, а запрос к базе данных с идентификатором тега позволяет определить, где (в какой зоне/области конкретного аэропорта) этот тег находится. И, соответственно, в зависимости от этого выдавать какую-либо информацию мобильному пользователю, разрешать ему определенные действия и т.д.

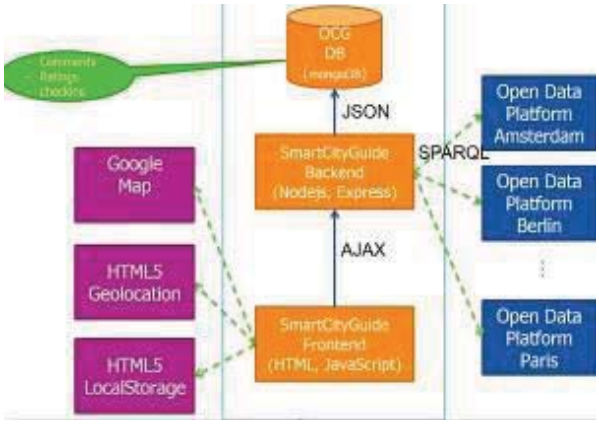


Рис. 5 The Open City Database

Еще одним примером является TheOpenCityDatabases проект (Fraunhofer, FOCUS [15] – рисунок 5). Здесь идея состоит в ведении единой базы данных о городских событиях и объектах. В число объектов могут входить и элементы сетевой инфраструктуры. Аналогичную роль может играть база городских сенсоров в проекте SmartSantander [16].

## 4 Модель данных для контекстно-зависимых сервисов на уровне города (CityProximus)

В данном разделе мы представляем модель и ее тестовую реализацию по представления контекстно-зависимых сервисов на уровне города – CityProximus.

Наш проект CityProximus представляет собой систему для поддержки контекстно-зависимых сервисов на основе сетевой близости. Выше мы указывали, что именно сервисы на основе сетевой близости являются самым простым способом реализации контекстно-зависимых вычислений. Проект CityProximus является дальнейшим развитием идей, реализованных в проектах Spot Expert [17] и Bluetooth Data Points [10]. В указанных проектах предлагались модели для поддержки сетевой близости в мобильных приложениях. Такого рода приложения использовались, например, в создании сервисов для организаций розничной торговли. Критика (замечания) относительно этих проектов касались именно вопросов масштабирования – а как это все будет функционировать при большом количестве сетевых узлов? CityProximus и представляет собой масштабируемый проект для контекстно-зависимых сервисов. Это масштабирование связано с изменением модели представления данных и, соответственно, с другой реализацией системы управления данными.

Общая идея системы для поддержки контекстно-зависимых сервисов состоит в создании (и, соответственно, поддержке) коллекции связок:

*Слепок сети – > Контент для Представления*

А также мобильного приложения для интерпретации текущего контекста (видимого представления беспроводной сети). Каждая связка представляет собой продукцию (логический оператор):

*если присутствует такой отпечаток сети, то мобильному пользователю доступен заданный контент*

Соответственно, система должна включать в свой состав следующие компоненты:

- Базу данных для хранения контента и правил его представления

- Редактор правил
- Сервер приложений
- Мобильное приложение для показа информации мобильным пользователям.

Мобильное приложение представляет из себя контекстно-зависимый браузер. Браузер определяет текущий сетевой отпечаток (список видимых сетевых узлов), отправляет этот список на сервер приложений (используется REST API) и получает в ответ контент для отображения (интерпретации). Контент представляет собой текст (JSON), интерпретацией которого занимался непосредственно контекстно-зависимый браузер. Для определения доступного контента для поступившего запроса сервер приложений выявлял истинные продукции (сработавшие правила). В начальных реализациях, условия в правилах представляли собой логические выражения, где предикаты описывали присутствие (отсутствие) беспроводной сети (сетей) и диапазон возможных значений для силы сигнала. Для быстрого определения сработавших правил использовался известный алгоритм RETE (традиционная модель для продукционных экспертных систем). В качестве базы данных использовалась система MySQL.

Развитие этого подхода шло по двум направлениям. Во-первых, описание логических выражений (условий в правилах) было затруднительно для пользователей и должно было быть как-то упрощено. Во-вторых, как указано выше, требовалось обеспечить больше возможностей для масштабирования системы работы с данными.

В части упрощения представления данных мы следовали модели iBeacons. Условия в правилах для доступности контента представляют собой просто логические предикаты, связанные с одни единственным узлом беспроводной сети, которые играют роль тегов. Соответственно, каждое правило выглядит подобным образом:

ЕСЛИ *УЗЕЛ_СЕТИ* (N) ДОСТУПЕН ТО *ДОСТУПЕН_КОНТЕНТ* (S)

С точки зрения хранения данных это приводит нас к классической модели key – value. Ключ здесь – это адрес узла беспроводной сети (MAC-адрес), значение – вектор элементов, составляющих контент, привязанный к данному адресу (текст, графическое изображение и т.д.) Модель Key-Value является одной из наиболее часто используемой в NoSQL. Одно из возможных решений по ее реализации - Apache Accumulo [18]. Это распределенное key-value хранилище. База данных для CityProximus представляет собой распределенную именно хэш-таблицу. Основные запросы выполняются по ключу, но возможно построение дополнительных индексов.

Данные представляются в виде пар ключ-значение, где ключ формируется из следующих элементов: RowID, Column (Family, Qualifier, Visibility) и отметка о времени (Timestamp). Ключ и значение представляются байтовым массивом, временная метка имеет тип *Long*. Таблица состоит из отсортированных пар ключ-значение. Accumulo сортирует ключи лексикографически по возрастанию. Отметки времени сортируются по убыванию, так что при последовательном просмотре последние версии ключа читаются первыми [19].

С точки зрения модели данных, CityProximus должен хранить следующую информацию:

(recordID, MAC_address, data_array)

Каждая запись сохраняет фрагмент данных (*data_array*) для заданного беспроводного узла (*MAC_address*). Для одного узла можно определять несколько информационных элементов.

Если говорить о статистике работы и оценке производительности, что необходимо для промышленного применения, то следует добавить еще, как минимум, два поля: дату создания правила и дату его последней модификации. У каждого правила есть автор (владелец) и он может временно деактивировать (активировать) созданные им правила. В итоге мы приходим к следующей структуре:

(userID, recordID, MAC_address, timestamp_created, timestamp_modified, status, data_array)

здесь поле *status* есть булево значение, которое и описывает текущий статус правила (активно оно или нет).

Массив данных содержит JSON структуру. Здесь описывается контент, который задается (определяется) правилом. Идея выбора JSON базируется на том, что такая структура позволяет клиенту (клиентскому приложению) самостоятельно решать, как должен отображаться соответствующий элемент. JSON массив, относящийся к правилу, содержит набор индивидуальных элементов:

```
[
{ “type”:”some_type”,”data”:”some_data”},
{“type”: ...},...
]
```

Программный API для базы данных правил возвращает тот же самый JSON массив. Поля здесь описывают стандартные типы, поддерживаемые системой. В текущей версии – это следующий набор: *text, url, image, email, phone, fbprofile, twprofile*. Text – это просто последовательность символов, *url* и *image* описывают веб-ресурсы, *fbprofile* и *twprofile* также представляют веб-ресурсы, но имеют специальную интерпретацию.

Например, *fbprofile* представляет некоторый URL для профиля из Facebook, *twprofile* есть то же самое, но для Twitter. Это позволяет различным программным клиентам реализовывать собственные алгоритмы отображения. Например, можно отображать каждый из указанных веб-адресов просто как ссылку, а можно использовать публичные интерфейсы социальной сети и запросить изображение (аватар) из пользовательского профайла.

Типичный запрос к базе данных правил является поиском по MAC-адресу. Это прямой запрос по ключу (первичный индекс) и он выполняется максимально быстро. Показанная производительность Apache Accumulo на такого рода запросах – 100 миллионов транзакций в секунду [20].

С точки зрения сбора статистики, система "эмулирует" события в веб-браузере. Событие (запись в веб-логе) здесь – это обращение устройства с MAC-адресом $MAC_1$ в момент времени *t* к фрагменту данных, ассоциированному с устройством $MAC_2$.

Базовый алгоритм функционирования достаточно прозрачен. Мобильное приложение (контекстно-зависимый браузер) получает (обновляет) список доступных (видимых) беспроводных узлов. Далее, для каждого узла (MAC-адреса) браузер выполняет поиск в базе данных правил. Собранные таким образом JSON-компоненты объединяются в единый массив, который и отображается (интерпретируется) в браузере. Отметим, что при сборке элементы массива могут быть отсортированы в соответствии с измеренной силой сигнала для конкретного беспроводного узла. Это позволит, например, показывать первыми в списке элементы, которые относятся к более близким узлам.

Естественно, среди всего набора беспроводных узлов, доступных в произвольный момент времени для мобильного устройства будут и такие, которые не относятся к проекту CityProximus. Обращение к базе правил с их идентификаторами заведомо вернет пустой результат. Для сокращения числа запросов мы использовали две стратегии. Во-первых, программным образом создаваемые узлы всегда имеют стандартное наименование. Это уже позволяет отфильтровать часть кандидатов. Во-вторых, можно задействовать кэш.

В качестве механизма реализации кэширования данных в предложенном проекте можно использовать Bloom-фильтр. Bloom-фильтр представляет собой метод (подход) для представления конечного множества элементов (ключей), который обеспечивает эффективную поддержку запросов о принадлежности произвольного элемента к данному множеству.

Согласно этому алгоритму мы начинаем искомое представление с вектора *V* из *m* бит. Изначально все биты установлены в 0. Далее выбираются k независимых хэш-функций $K = \{h_1, h_2, \ldots, h_k\}$, каждая из них в интервале $\{1,\ldots,m\}$. Для каждого элемента (ключа) *a* из множества A биты в позициях $h_1(a)$, $h_2(a)$, ..., $h_k(a)$ в V устанавливаются в 1. Возможны повторения – конкретный бит будет устанавливаться в 1 несколько раз. Если поступает запрос с ключом *b*, то мы проверяем биты в позициях $h_1(b)$, $h_2(b)$, ..., $h_k(b)$. Если один из них есть 0, то данный ключ точно не находится в множестве A. В противном случае мы заключаем, что ключ находится в кэше. Кэш здесь есть набор проверяемых MAC-адресов.

Это вероятностный подход, возможны ложные срабатывания (ложные заключения о принадлежности). Частота таких срабатываний управляется выбором пары значений *m* и *k*. В данном случае ложное срабатывание будет означать просто лишний запрос к базе данных.

## 5 Заключение

В данной работе рассмотрена модель представления данных для контекстно-зависимых сервисов, основанных на понятии сетевой близости. В таких сервисах информация о местоположении заменяется информацией о близости объекта к элементам сетевой инфраструктуры. В качестве таких элементов выступают узлы беспроводной сети (Wi-Fi, Bluetooth).

В качестве хранилища данных рассмотрено использование модели ключ-значение и ее конкретной реализации в проекте Apache Accumulo. Рассмотрены вопросы организации кэширования данных.

В качестве областей применения такого рода сервисов можно назвать приложения, ассоциируемые с направлением Smart Cities. В частности, системы информирования мобильных пользователей в помещениях (например, посетителей торговых заведений, кампусов и т.д.).

## Литература


[1] G. Schilit and B. Theimer Disseminating Active Map Information to Mobile Hosts. IEEE Network, 8(5) (1994) pp. 22-32

[2] A. Day, "Understanding and Using Context" (2001). Human-Computer Interaction Institute. Paper 34. http://repository.cmu.edu/hcii/34

[3] N. Hristova and G. M. P. O`Hare, "Ad-me: Wireless Advertising Adapted to the User Location, Device and Emotions," in Thirty-Seventh Hawaii International Conference on System Sciences (HICSS-37), 2004.

[4] Namiot, D., & Sneps-Sneppe, M. (2013, March). Wireless networks sensors and social streams. In Advanced Information Networking and Applications Workshops (WAINA), 2013 27th International Conference on (pp. 413-418). IEEE.

[5] Namiot, Dmitry, and Manfred Sneps-Sneppe. "On Open Source Mobile Sensing." Internet of Things,

Smart Spaces, and Next Generation Networks and Systems. Springer International Publishing, 2014. 82-94.

[6] Yassin, M., Rachid, E., & Nasrallah, R. (2014, November). Performance comparison of positioning techniques in Wi-Fi networks. In Innovations in Information Technology (INNOVATIONS), 2014 10th International Conference on (pp. 75-79). IEEE.

[7] Namiot D. Network Proximity on Practice: Context-aware Applications and Wi-Fi Proximity //International Journal of Open Information Technologies. – 2013. – Т. 1. – №. 3. – С. 1-4.

[8] Y. Chen, Y. Chawathe, A. LaMarca, and J. Krumm. "Accuracy characterization for metropolitan-scale Wi-Fi localization", In ACM MobiSys, 2005.

[9] M. Kjaergaard, M. Wirz, D. Roggen, and G.Troster Mobile sensing of pedestrian flocks in indoor environments using WiFi signals Pervasive Computing and Communications (PerCom), 2012 IEEE International Conference on pp. 95 – 102.

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

[11] Eagle, N., & Pentland, A. (2006). Reality mining: sensing complex social systems. Personal and ubiquitous computing, 10(4), 255-268.

[12] Funf package http://www.funf.org

[13] Farshad, A., Marina, M. K., & Garcia, F. (2014, May). Urban WiFi characterization via mobile crowdsensing. In Network Operations and Management Symposium (NOMS), 2014 IEEE (pp. 1-9). IEEE.

[14] SITA Common Use Beacon Registry https://www.developer.aero/Beacon-Registry-API/API-Overview

[15] Tcholtchev N. et al. Fraunhofer Inst. for Open Commun. Syst.(FOKUS), Berlin, Germany //Local Computer Networks Workshops (LCN Workshops), 2012 IEEE 37th Conference on. – IEEE, 2012. – С. 860-867.

[16] Sanchez, L., Galache, J. A., Gutierrez, V., Hernández, J. M., Bernat, J., Gluhak, A., & García, T. (2011, June). Smartsantander: The meeting point between future internet research and experimentation and the smart cities. In Future Network & Mobile Summit (FutureNetw), 2011 (pp. 1-8). IEEE.

[17] Sneps-Sneppe M., Namiot D. Smart cities software: customized messages for mobile subscribers //Wireless Access Flexibility. – Springer Berlin Heidelberg, 2013. – С. 25-36.

[18] Halldórsson, Guðmundur Jón. Apache Accumulo for Developers. Packt Publishing Ltd, 2013.

[19] Accumulo User Manual https://accumulo.apache.org/1.6/accumulo_user_manual.html#_introduction Retrieved: Apr, 2015

[20] Sen, R., Farris, A., & Guerra, P. (2013, June). Benchmarking Apache Accumulo BigData Distributed Table Store Using Its Continuous Test Suite. In Big Data (BigData Congress), 2013 IEEE International Congress on (pp. 334-341).

## On Data Model for Context–Aware Services

Dmitry Namiot, Manfred Sneps-Sneppe

This paper discusses the issues of data representation for one model of context-aware services. This model is based on network proximity ideas. For network proximity, the location information (latitude, longitude) is replaced with information about proximity to network nodes. Network nodes play a role of tags and let position the users within some restricted area. In this paper, we target wireless networks nodes (Wi-Fi access points, Bluetooth nodes). Services for such models provide to mobile subscribers access to certain information, depending on availability (presence) of wireless networks. And another option is a publication by mobile users context-sensitive information associated with wireless networks. In our paper, we discuss key-value data store deployment as a basic persistence mechanism for network proximity. Also, we provide a description for system-wide cache on the base of Bloom filter.