УДК 004.7

В. О. Демиш, **Б. Н. Пищик**

Новосибирский государственный университет
ул. Пирогова, *2*, *Новосибирск*, *630090*, *Россия*

Конструкторско-технологический институт вычислительной техники СОРАН
ул. Акад. Ржанова, *6*, *Новосибирск*, *630090*, *Россия*

v.demish@gmail.com, *Borix.Pishchik@gmail.com*

АВТОНОМНАЯ РАБОТА ANDROID-ПРИЛОЖЕНИЙ И АЛГОРИТМЫ СИНХРОНИЗАЦИИ ДАННЫХ

Рассматриваются проблемы разработки мобильных приложений, поддерживающих автономный режим работы (без подключения к сети) с последующим проведением сеансов синхронизации связи. На примере Android приложений рассмотрены алгоритмы синхронизации данных, используемые в Google Analytics и Evernote. Обозначена актуальность задачи оптимизации хранения данных, необходимых для сеансов синхронизации, на мобильном устройстве.

Ключевые слова: мобильные платформы, синхронизация данных, разработка мобильных приложений, базы данных.

Введение

Согласно данным CiscoSystems объем мобильного трафика за 2013 г. увеличился на 83 % – до 1,5 экзабайтов к концу 2013 г. с 820 петабайтов в конце 2012 г. [1] Темпы роста рынка мобильных устройств говорят о том, что тенденция роста объемов мобильного трафика будет только укрепляться.

С развитием мобильных устройств стремительно развивается и сфера мобильных приложений. Растет количество мобильных приложений, стремительно расширяются возможности инструментов для их разработки. К примеру, если сейчас начать обучаться программированию под мобильную операционную систему Android по литературе 2–3-годичной давности, то вполне можно встретить множество устаревших (deprecated) методов. Это объяснимо: с мая 2010 по октябрь 2013 г. Google продвинула операционную систему Android с версии 2.2 до 4.4.

Параллельное распространение технологии облачных вычислений (Cloud Computing) [1] явным образом приводит к необходимости взаимодействия мобильных устройств и облачных сервисов – ведь это взаимодействие двух стремительно развивающихся в настоящее время технологий.

С учетом того, что облачные сервисы требуют постоянного подключения к сети Интернет, а мобильные устройства зачастую работают в режиме оффлайн, перед разработчиками мобильных приложений встает задача синхронизации (или интеграции, в зависимости от за-

[1] Cisco Systems. Cisco Visual Networking Index: Global Mobile. 2014. URL: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.pdf

Демиш В. О., *Пищик Б. Н.* Автономная работа android-приложений и алгоритмы синхронизация данных // Вестн. Новосиб. гос. ун-та. Серия: Информационные технологии. 2014. Т. 12, вып. 3. С. 25–33.

дачи) данных мобильного и облачного приложения в моменты выхода мобильных устройств в сеть.

Рассмотрим несколько примеров приложений, в которых решена эта задача.

- **Dropbox** (http://www.dropbox.com) – облачное хранилище данных. Dropbox позволяет пользователю размещать файлы на удаленных серверах при помощи клиента или с использованием веб-интерфейса через браузер. При установке клиентского программного обеспечения Dropbox на компьютере создается синхронизируемая папка. Существуют мобильные клиенты для мобильных устройств.
- **Evernote** (http://evernote.com) – веб-сервис и набор программного обеспечения для создания и хранения заметок. В качестве заметки может выступать фрагмент форматированного текста, веб-страница целиком, фотография, аудиофайл или рукописная запись. Заметки могут также содержать вложения с файлами другого типа. Их можно сортировать по блокнотам, присваивать им метки, редактировать и экспортировать. Заметки со всех устройств пользователя синхронизуются с сервисом. Существуют мобильные клиенты для мобильных устройств.
- **AmazonKindle** (http://kindle.amazon.com) – вместе с серией электронных книг компания Amazon.com выпускает мобильные приложения, позволяющие читать книги на мобильных устройствах. При этом происходит синхронизация книг и закладок в книгах, читаемых с различных устройств.

Во всех примерах вы можете полноценно работать и без подключения к сети – записывать файлы в папку Dropbox, редактировать записки Evernote и читать электронные книги с Kindle. Но с каждым следующим выходом в сеть перечисленные приложения будут проводить сеансы синхронизации с центральным сервером.

Таким образом, взаимодействие мобильных приложений с облачными сервисами зачастую предполагает решение задач синхронизации данных. Причем в каждом отдельном случае детали этой задачи могут значительно отличаться в зависимости от таких факторов, как направленность синхронизации данных, объемы передаваемых данных, различные требования к устойчивости и т. д.

В настоящей статье приведен анализ решения задачи отложенной синхронизации данных от Google, реализованного в сервисе сбора аналитики Google Analytics [2], а также решения от Evernote для синхронизации электронных заметок на различных устройствах.

Платформа GoogleAnalytics

Изначально GoogleAnalytics – это платформа для сбора и обработки статистики посещения веб-сайтов, разработанная корпорацией Google. Платформа позволяет владельцам веб-ресурсов пользоваться сервисом сбора и обработки аналитики посредством размещения специального кода на языке JavaScript на страницах сайта, подлежащих сбору статистики по их использованию.

Размещаемый таким образом JavaScript-код содержит идентификатор веб-сайта, по которому идет сбор аналитики. Таким образом, вся статистика по работе посетителей с веб-сайтом отправляется на сервер Google и становится доступной для анализа.

С развитием мобильных технологий Google расширила сервис аналитики до работы с мобильными устройствами. Принцип работы остался прежним, но вместо JavaScript-кода происходит подключение соответствующей мобильной библиотеки и заполнение специального конфигурационного файла. И если веб-аналитика построена на анализе посещения *страниц* веб-ресурсов, то мобильная аналитика анализирует работу с так называемыми Activity. Activity представляет собой визуальный интерфейс (отдельный экран) для одного действия, которое пользователь может совершить в рамках мобильного приложения.

Сервис продолжает собирать аналитику использования мобильного приложения даже тогда, когда мобильное устройство отключено от сети. Разберем на конкретном примере принцип работы Google Analytics в этом контексте. С этой целью создадим тестовое приложение под Android и подключим к нему библиотеку Google Analytics Android SDK (при написании статьи использовалась библиотека версии 3.01).

Анализ решения проблемы отложенной отправки данных

Проведем анализ работы инструментов GoogleAnalytics в режиме оффлайн на примере мобильной платформы Android. Для этого потребуется AndroidSDK, а также IDE (в примере использовалась Android Studio).

Создадим новый проект (с пустым Activity), затем создадим еще одно Activity и снабдим приложение кнопкой для его открытия из главного окна приложения. Оба Activity снабдим обработчиками событий onStart и onStop (реализующими сбор данных для Google Analytics):

```
@Override
protected void onStart() {
super.onStart();
EasyTracker.getInstance(this).activityStart(this); // Add this method.
}

@Override
protected void onStop() {
super.onStop();
EasyTracker.getInstance(this).activityStop(this);
}
```

Подключим, как описывалось ранее, GoogleAnalyticsSDK и запустим приложение (рис. 1). Поскольку в дальнейшем потребуется посмотреть на базу SQLite созданного приложения, необходимо запускать его не на реальном устройстве, а посредством эмулятора. Нажимая на кнопку открытия другого Activity (на рис. 1 это кнопка «AnotherActivity») и кнопку «Назад», предопределенную в операционной системе Android, сгенерируем тем самым исходные данные для GoogleAnalytics. В панели GoogleAnalytics можно удостовериться, что переходы между Activity успешно зафиксированы.

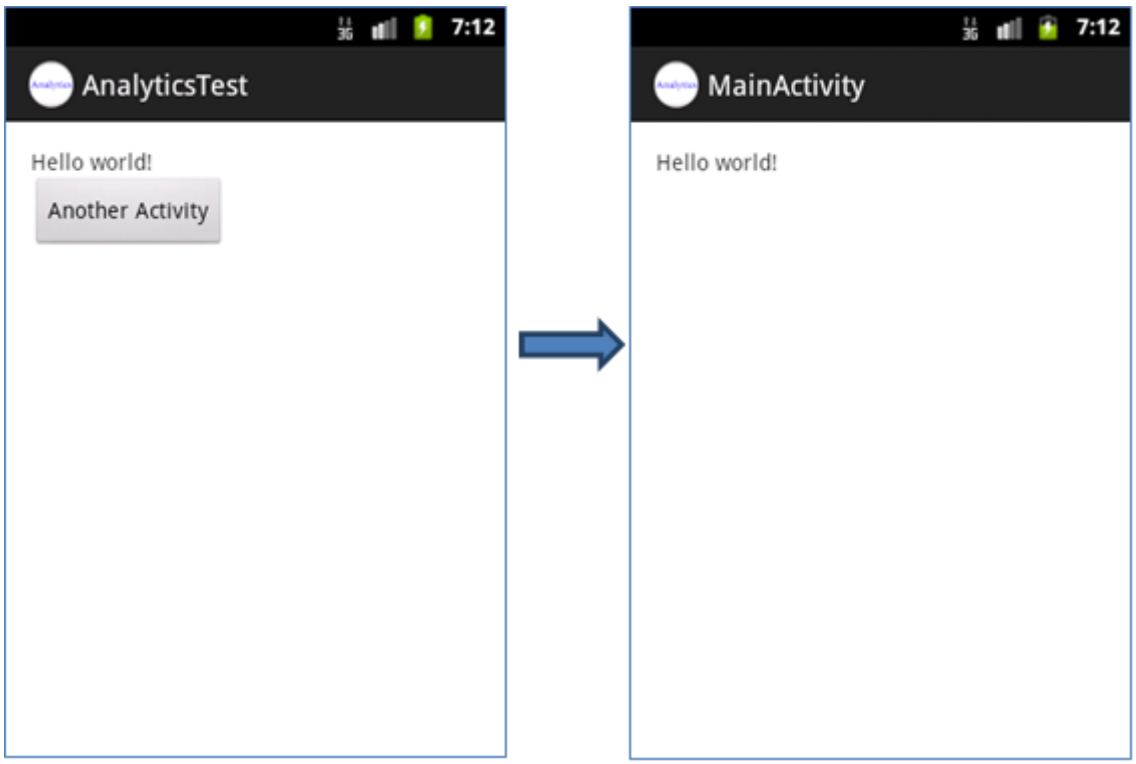

Рис. 1. Тестовое приложение

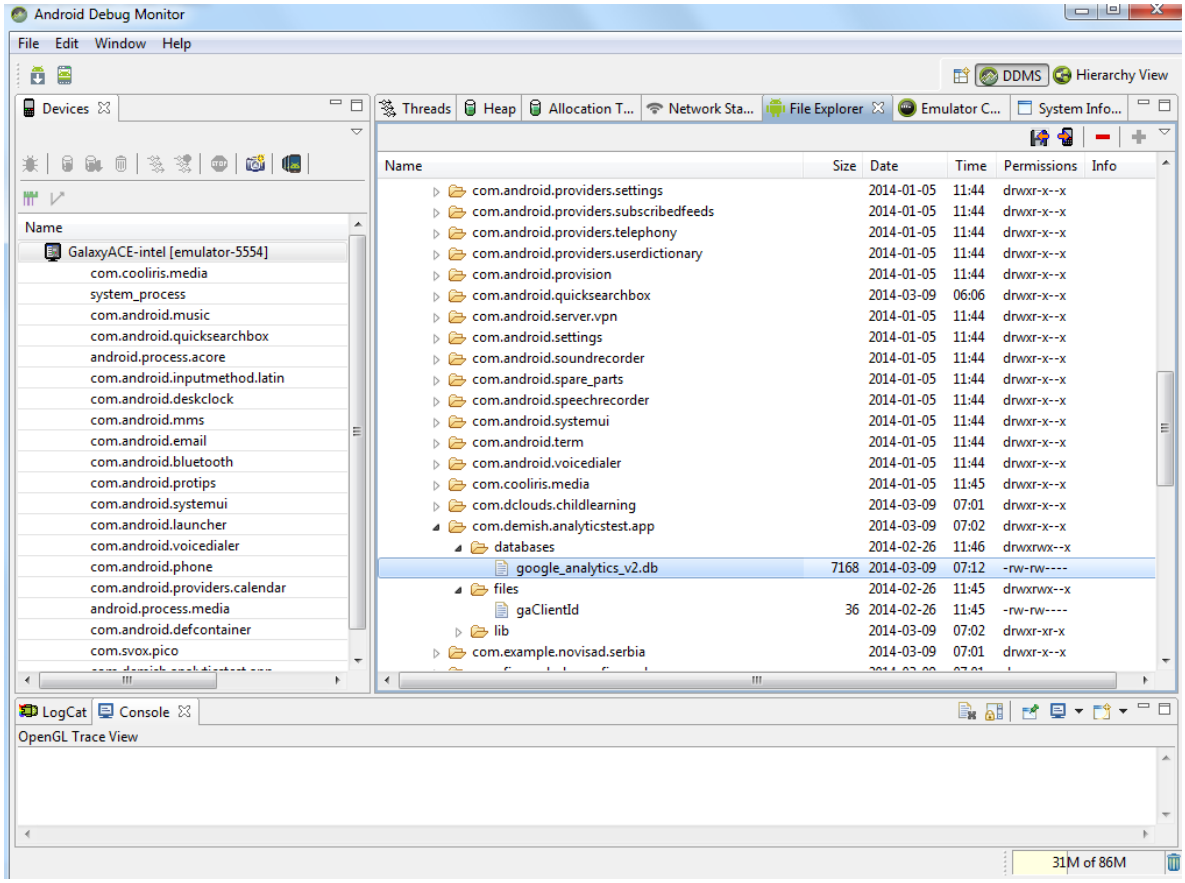

Рис. 1. File Explorer

hit_id	hit_time	hit_url	hit_string	hit_app_id
31	1394344964517	https:	v=1&ul=en-us&t=appview&ht=1394344964517&sr=320x480&an=AnalyticsTest&tid=...	0
32	1394344966484	https:	v=1&ul=en-us&t=appview&ht=1394344966484&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
33	1394344967003	https:	v=1&ul=en-us&t=appview&ht=1394344967003&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
34	1394344967423	https:	v=1&ul=en-us&t=appview&ht=1394344967423&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
35	1394344967915	https:	v=1&ul=en-us&t=appview&ht=1394344967915&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
36	1394344968369	https:	v=1&ul=en-us&t=appview&ht=1394344968369&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
37	1394344968857	https:	v=1&ul=en-us&t=appview&ht=1394344968857&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
38	1394345106743	https:	ul=en-us&ht=1394345106743&sr=320x480&sc=start&aid=com.demish.analyticstest.app&cid...	0
39	1394345108103	https:	v=1&ul=en-us&t=appview&ht=1394345108103&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
40	1394345108682	https:	v=1&ul=en-us&t=appview&ht=1394345108682&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
41	1394345109090	https:	v=1&ul=en-us&t=appview&ht=1394345109090&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
42	1394345139683	https:	v=1&ul=en-us&t=appview&ht=1394345139683&sr=320x480&an=AnalyticsTest&tid=UA-484...	0
43	1394345140236	https:	v=1&ul=en-us&t=appview&ht=1394345140236&sr=320x480&an=AnalyticsTest&tid=UA-484...	0

Рис. 2. Данные таблицы hits2 Google Analytics

Теперь, отключив Интернет от эмулятора Android-устройства (к примеру, включив на нем режим полета), повторное переключение между экранами тестового приложения уже не будет фиксироваться в GoogleAnalytics. Но, тем не менее, в дальнейшем, при подключении сети, эта аналитика все же будет передана на серверы Google. Как это достигается?

Запустив из AndroidStudio инструмент AndroidDebugMonitor, выберем в нем панель FileExplorer. Скопируем базу данных SQLite из эмулятора мобильного устройства на компь-

ютер. Она расположена по адресу /data/data/<Имя приложения>/databases/google_analytics_v2.db.

Когда база будет на компьютере, можно открыть ее любым инструментом для работы с SQLite, например, SQLiteManager для Firefox.

В базе данных три таблицы:

1) android_metadata;
2) hits2;
3) sequence.

Первая таблица имеет один атрибут *locale*, содержащий данные о локализации приложения (в случае рассматриваемого тестового приложения в таблице одна строка со значением «en_US»). Последняя таблица хранит данные об SQL-последовательностях, используемых для значений первичных ключей таблиц базы данных.

Наиболее интересна таблица hits2:

```
CREATETABLEhits2 (
'hit_id' INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,
'hit_time' INTEGER NOT NULL,
'hit_url' TEXT NOT NULL,
'hit_string' TEXT NOT NULL,
'hit_app_id' INTEGER)
```

Каждая строка в этой таблице – это исходные данные для GoogleAnalytics (рис. 3). В поле hit_time хранится дата зафиксированного события, а его описание располагается в поле hit_string. Описание события представляет собой набор параметров (в том же поле размещается также информация о возникающих в процессе работы приложения исключениях), содержащих такие данные, как:

- версия приложения;
- региональные параметры;
- время события;
- тип события (для исключений тут будет «exception);
- разрешение экрана мобильного приложения;
- идентификационный код для GoogleAnalytics;
- другие данные, актуальные для конкретного типа событий.

Пример такого описания:

v=1&ul=en-us&t=appview&ht=1394345108682&sr=320x480&an=AnalyticsTest&tid=<Код GoogleAnalytics>&aid=com.demish.analyticstest.app&cid=fb24b1b1-1a6a-4175-9c94-8c3a4e8d13d1&av=1.0&_u=.98&_v=ma3.0.1&cd=com.demish.analyticstest.app.MainActivity2

Таким образом, GoogleAnalytics сохраняет в базу данных информацию о работе с приложением в том случае, если нет возможности сразу отправить эту информацию на сервер Google. Когда же эта возможность появляется, вся сохраненная информация отправляется в Google. Кроме того, дата фиксации события на мобильном устройстве не зависит от того, когда информация о событии будет отправлена в GoogleAnalytics. Это позволяет видеть реальную картину использования приложений.

Режимы отправки данных аналитики с мобильных устройств также различны: есть автоматический (по умолчанию попытка отправки данных предпринимается каждые 30 минут) и ручной режимы.

Для настройки сбора информации в режиме оффлайн есть параметр QueueTime (qt, в миллисекундах), регулирующий разницу между временем события и временем отправки информации о нем. Информация о событиях, для которых эта разница будет больше параметра QueueTime, не будет обработана. Однако фактически этот параметр всегда ограничен временем 04.00 следующего после времени события дня.

Синхронизация данных на примере Evernote

Рассмотрим теперь другой, более сложный вариант синхронизации данных на примере популярного мобильного приложения для работы с электронными заметками, которое уже упоминалось ранее, – Evernote.

Работу приложения можно проанализировать также при помощи инструментов разработки под Android, но Evernote представляет собой более сложное приложение по отношению к библиотеке GoogleAnalytics. Поэтому такой подход несколько не оправдан, вместо этого обратимся к документации для разработчиков приложений, которая предоставлена авторами Evernote.

С помощью документации от Evernote последовательно разберем принципы работы приложения, а также реализованный в нем алгоритм синхронизации данных.

Описание интерфейса

Все сущности сервиса описаны при помощи языка описания интерфейсов Apache Thrift [2] [3; 4]. И уже на его основе создан собственный протокол Evernote Data Accessand Management (EDAM) [3].

Thrift позволяет использовать файлы-описания интерфейса для дальнейшей генерации исходных кодов на различных языках. Сгенерированные классы применяются для работы с данными и вызова удаленных процедур, они представляют собой готовую основу для API, в рамках которого требуется реализовать необходимую бизнес-логику. Все описанные сущности, необходимые классы для работы серверной и клиентской части, генерируются автоматически.

Чтобы построить API при помощи Thrift, необходимо описать используемые сущности при помощи доступных в Thrift типов данных, а также описать в специальной нотации используемые сервисы. Детали проектирования можно изучить в соответствующей литературе [4]. Остановимся подробнее на Evernote.

В Evernote определены следующие сущности:

- User;
- UserAttribute;
- Notebook;
- Tag;
- Note;
- NoteAttribute;
- Resource;
- ResourceAttribute;
- SavedSearch;
- LinkedNotebook.

А также два сервиса с набором методов (перечислены в скобках):

- UserStore (authenticate, refreshAuthentication);
- NoteStore (createNote, createNotebook, createSearch, createTag, deleteNote, deleteNotebook, deleteSearch, deleteTag, getNote, getNoteContent, getNotebook, getResource, getResourceData, getSearch, getSyncChunk, getSyncState, getTag, listNotebooks, listNotes, listSearches, listTags, updateNote, updateNotebook, updateSearch, updateTag).

Предназначение сущностей и суть методов в общих чертах понятны из названий. Более полную информацию можно получить из документации Evenote.

Получившиеся вследствие кодогенерации Thrif и реализации бизнес-логики API позволяет осуществить взаимодействие с множеством различных приложений. Грамотно спроектированные сущности и сервисы позволяют воспользоваться тем же самым API для реализации алгоритма синхронизации данных на центральном сервере Evernote и клиентскими устройствами.

Алгоритм синхронизации. В документации от Evernote выделены следующие требования к синхронизации.

- Синхронизация осуществляется в клиент-серверном режиме, центральный сервер является основным.
- Клиенты обладают локальной базой данных для хранения информации, так что синхронизация ограничена логическим представлением данных; нет необходимости рассматривать данные на более низком, физическом уровне.

[2] Apache. Язык описания интерфейсов Apache Thrift. http://thrift.apache.org/.
[3] Corporation, Evernote. Evernote Synchronization via EDAM. 2011.

- Необходимо поддержка полной синхронизации (fullsync) и обмена изменениями (incremental sync) в базе данных; ситуация, в которой каждый сеанс синхронизации представляет собой передачу всей базы данных, недопустима.
- Синхронизация должна быть устойчивой к разрывам сети, в случае которых сеанс синхронизации должен в дальнейшем продолжиться без значительных повторов в передаче данных; для первоначальной синхронизации клиенты должны иметь возможность продолжить передачу данных после перебоев в работе сети.
- Сеанс синхронизации не должен блокировать работу сервиса; допустимы изменения от других клиентов во время сеанса синхронизации.

В Evernote используется так называемый Update Sequence Number (USN). С помощью USN можно упорядочить данные в порядке их изменения. При создании нового аккаунта USN для него инициализируется единицей (точнее, единица будет соответствовать первому созданному объекту). Каждое добавление / изменение / удаление данных в рамках аккаунта характеризуется привязкой к операции USN и увеличением его на единицу.

Для функционирования алгоритма синхронизации на сервере и клиенте используется ряд переменных.

На сервере:

- *updateCount* – наибольший USN для каждого аккаунта;
- *fullSyncBefore* – специальная дата, которая позволяет указать на необходимость проведения клиентом обязательной полной синхронизации (full sync); если последняя успешная синхронизация на клиенте была ранее даты fullSyncBefore, то потребуется проведение полной синхронизации.

На клиенте:

- *lastUpdateCount* – значение серверной переменной updateCount при последней синхронизации;
- *lastSyncTime* – дата последней синхронизации данных (получается с сервера).

Таким образом, для синхронизации клиенты должны получить данные с сервера, изменившиеся с момента последней синхронизации, т. е. имеющие USN выше *lastUpdateCount*.

Пошагово синхронизация выглядит следующим образом.

1. Клиент получает новые и изменившиеся объекты с сервера (с момента последней синхронизации).
2. Клиент согласовывает полученные объекты с локальной базой данных.
3. Клиент отправляет изменения на сервер.
4. Клиент запоминает состояние сервера для последующей синхронизации.

Разрешение конфликтов происходит на клиенте. Для реализации третьего шага измененные на клиенте данные должны помечаться соответствующим флагом «изменен». Более подробная информация об алгоритме синхронизации (оформленная в виде псевдокода) есть в документации для разработчиков.

Выводы

В статье рассмотрены два совершенно различных подхода к взаимодействию клиентской части (мобильного приложения) и сервера в отношении синхронизации данных. Для анализа алгоритма работы GoogleAnalytics продемонстрированы соответствующие инструменты для разработчиков.

Ввиду различной специфики решаемых задач рассмотренные подходы значительно отличаются друг от друга:

- в GoogleAnalitycs осуществляется однонаправленная передача информации с клиента на сервер; каждое сообщение никак не связано с остальными (появление нового сообщения не отражается на сформированных ранее); синхронизация данных сводится к передаче всех сообщений на сервер без какой-либо дополнительной обработки;
- в Evernote реализован алгоритм синхронизации с центральным сервером, позволяющий производить как полную синхронизацию данных, так и обмен изменениями; необходимость в такой реализации диктуется двунаправленным обменом информации – данные модифицируются как на сервере, так и на клиенте.

Для осуществления однонаправленной синхронизации можно использовать реализацию очереди на клиенте, в которую складываются сообщения для последующей отправки, а для двунаправленной синхронизации требуется реализация более сложного алгоритма.

Несмотря на относительную простоту, реализация очереди на клиента может сопровождаться значительными проблемами, вызванными неограниченным ростом базы данных: если мобильное устройство будет постоянно находиться в режиме оффлайн, то база данных для хранения информации к отправке будет разрастаться. Как именно с этим бороться? Из очевидных решений - ограничение объема базы данных, периода актуальности данных (к примеру, сохранять информацию не более чем недельной давности).

В статье рассмотрены два подхода из множества [5–7] других, но и из сравнения лишь двух решений можно сделать вывод о чрезвычайно сильной зависимости в выборе алгоритма синхронизации от стоящих перед синхронизацией требований. Какой (или какие) алгоритмы следует применять в той или иной задаче, можно сказать лишь после детального ее изучения.

Обозначенная проблема переполнения базы данных с информацией, необходимой для проведения синхронизации, актуальна для всех алгоритмов синхронизации. Это обусловлено необходимостью накопления всех происходящих изменений. Таким образом, решение задачи оптимизации хранения данных, требуемых для синхронизации, напрямую влияет на эффективность алгоритма синхронизации данных в целом.

Список литературы

1. *Риз Дж.* Облачные вычисления. СПб.: БХВ-Петербург, 2011.

2. *Брайан К.* Google Analytics для профессионалов (Advanced Web Metrics with Google analytics). 3-е изд. М.: Диалектика, 2013.

3. *Agarwal A.*, *Slee M.*, *Kwiatkowski M.* Thrift: Scalable Cross-Language Services Implementation. URL: http://thrift.apache.org/static/files/thrift-20070401.pdf/.

4. *Randy A.* The Programmer's Guide to Apache Thrift. M.: Manning Publication Co, 2013.

5. *Fang Hengming*, *Chen Jia*, *Xu Bin.* The Interaction Mechanism based on JSON for Android Database Application // Information Technology Journal. 2013. Vol. 12. P. 224–228.

6. *Ranjeet Singh*, *Chiranjit Dutta.* A Synchronization Algorithm of Mobile // International Journal of Application or Innovation in Engineering & Management (IJAIEM). 2013. Vol. 3, iss. 2. P. 491–497.

7. *B Sri Ramya*, *Shirin Bhanu Koduri*, *M. Seetha.* A Stateful Database Synchronization Approach for Mobile Devices // International Journal of Soft Computing and Engineering (IJSCE). 2012. Vol. 2, iss. 3.

Материал поступил в редколлегию 25.09.2014

V. O. Demish, **B. N. Pischik**

Novosibirsk State University
2 Pirogov Str., *Novosibirsk*, *630090*, *Russian Federation*

Design Technological Institute of Digital Techniques SB RAS
6 Rzhanov Str., *Novosibirsk*, *630090*, *Russian Federation*

v.demish@gmail.com, *Borix.Pishchik@gmail.com*

OFFLINE ANDROID APPLICATION AND ALGORITHMS OF DATA SYNCHRONIZATION

This article deals with problem of development mobile applications allowing offline mode (without network connection) with following synchronization sessions. By example of Android ap-

plications described algorithm of data sync used in Google Analytics and Evernote. Denoted actuality of optimization data required for synchronization sessions on mobile devices.

Keywords: mobile platforms, data synchronization, mobile application development, databases.

References

1. Reese J. *Cloud computing*. St.-Petersburg, 2011. (in Russ.)

2. Brian K. *Google Analytics for professionals* (*Advanced Web Metrics with Google analytics*). Moscow, 2013. (in Russ.)

3. Agarwal A., Slee M., Kwiatkowski M. *Thrift: Scalable Cross-Language Services Implementation*. URL: http://thrift.apache.org/static/files/thrift-20070401.pdf/.

4. Randy A. *The Programmer's Guide to Apache Thrift*. Moscow, Manning Publication Co, 2013.

5. Fang Hengming, Chen Jia, Xu Bin. The Interaction Mechanism based on JSON for Android Database Application. *Information Technology Journal*, 2013, vol. 12, p. 224–228.

6. Ranjeet Singh, Chiranjit Dutta. A Synchronization Algorithm of Mobile. *International Journal of Application or Innovation in Engineering & Management* (*IJAIEM*), 2013, vol. 3, iss. 2, p. 491–497.

7. B Sri Ramya, Shirin Bhanu Koduri, M. Seetha. A Stateful Database Synchronization Approach for Mobile Devices. *International Journal of Soft Computing and Engineering* (*IJSCE*), 2012, vol. 2, iss. 3.