Санкт-Петербургский Государственный Университет

Математико-механический факультет

Кафедра системного программирования

Реализация мобильных сервисов для доступа к удаленным устройствам на базе платформы Ubiq Mobile.

Курсовая работа студента 445 группы
Тумановой Кристина Сергеевны

Научный руководитель В. В. Оносовский

Санкт-Петербург

2010

Оглавление

1. Введение и постановка задачи

Задачей данной курсовой работы является разработка универсального структурного шаблона для реализации на базе платформы Ubiq Mobile мобильных сервисов, предоставляющих доступ к удаленным устройствам. Реализация сервиса, позволяющего пользователю получать на свой мобильный телефон изображения с веб-камеры, установленной на удаленном компьютере, была выбрана как более частный случай общей задачи.

Платформа Ubiq Mobile инкапсулирует в себе большинство технических деталей, специфических для мобильных устройств, тем самым предоставляя разработчикам простой и наглядный API для разработки современных распределенных мобильных онлайн-приложений и сервисов. Разработанные для платформы Ubiq Mobile мобильные сервисы характеризуются высокой эффективностью, богатыми функциональными возможностями и низкими требованиями к ресурсам, что позволяет им эффективно работать на различных мобильных устройствах и в различных типах мобильных соединений, в том числе и в условиях относительно медленных каналов связи (GPRS, EDGE).

Одной из ключевых идей платформы Ubiq Mobile является использование терминальной архитектуры, при которой мобильные устройства выступают в роли графических терминалов, а вся бизнес-логика приложений реализуется на сервере. Это, с одной стороны, существенно облегчает инкапсуляцию «мобильно-зависимых» технических деталей и, в целом, упрощает структуру разрабатываемых распределённых приложений и снижает уровень требований к ресурсам мобильных устройств.

С другой стороны, терминальная архитектура накладывает на разрабатываемые приложения ряд ограничений, основными из которых являются относительная статичность пользовательского интерфейса (например, по сравнению с «локальными» анимированными мобильными приложениями) и невозможность offline работы. Оба этих ограничения являются фундаментальными свойствами, присущими любым терминальным системам. С другой стороны, для большинства информационных и деловых сервисов наличие анимации в пользовательском интерфейсе не является чем-то принципиально важным, а невозможность offline работы отчасти компенсируется реализованным в платформе механизмом сохранения состояния пользовательских сессий.

Преимущества и ограничения платформы Ubiq Mobile определяют «целевые» классы мобильных онлайн-приложений, для которых использование платформы наиболее оправдано и эффективно. Одним из важных классов «целевых» приложений являются приложения для дистанционного управления удаленными устройствами и системами, имеющими интерфейс с ПК, с мобильных телефонов. В качестве удаленных устройств могут выступать

- Веб-камеры, подключённые к стационарным компьютерам пользователей, с которых пользователь может получать изображения на свой мобильный телефон;
- Системы видеонаблюдения, транслирующие в непрерывном режиме «картинку» с нескольких камер наблюдения на мобильное устройство;
- Системы управления загородным домом (офисом, магазином, гаражами, складскими помещениями), имеющие компьютерный интерфейс. В данном случае пользователь получает возможности удаленного контроля за климатом (свет, температура воздуха, влажность), контроля охранных систем (пожарная охрана, охрана от вторжения) и контроля сбоев в инженерных системах (протечка воды, утечка газа).
- Системы безопасности и сигнализации, например, для автомобиля. При проникновении в автомобиль или при попытке его транспортировки, система посылает сигнал на телефон владельца; владелец же, в свою очередь, может послать системе команду блокировать двигатель автомобиля, включить сирену или другие специальные средства.
- И т.д.

Все эти системы обладают рядом общих черт –

- Происходит двусторонний обмен информацией (от системы к пользователю – информация о состоянии системы, от пользователя к системе – управляющие команды);
- Удалённое устройство или система работает в непрерывном режиме;
- Пользователь должен иметь возможность в любой момент «подключиться» к устройству со своего мобильного телефона;

- Также желательно иметь возможность посылки системой «сигналов тревоги» при наступлении каких-то важных событий, требующих вмешательства пользователя.

В рамках данной курсовой работы в качестве примера разрабатывается сервис передачи изображений с удаленных веб-камер на мобильные телефоны пользователей. Рассматриваемая модель представлена на рис. 1.

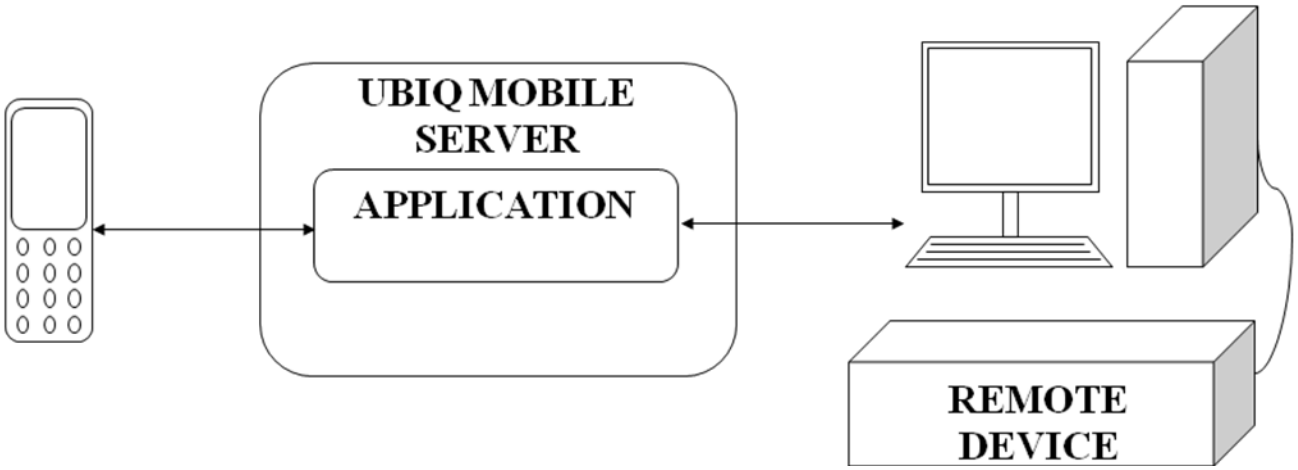

Рисунок 1. Общая модель сервиса.

Пусть имеется удаленное устройство, подключённое к компьютеру, имеющему выход в Интернет, которое может выступать, с одной стороны, в роли источника некоторой информации (например, набора значений параметров системы управления загородным домом или изображений, передаваемых с веб-камеры), а, с другой стороны, способно воспринимать управляющие команды от компьютера, к которому оно подключено. Пользователь хочет удаленно, посредством мобильного устройства (телефона или коммуникатора), управлять им: получать информацию о текущих значениях измеряемых параметров или о выполненных действиях и, возможно, на основе этой информации формировать и отправлять устройству команды для изменения состояния контролируемых им объектов (например, команду повысить температуру в загородном доме). Информация от устройства должна передаваться пользователю не только по его запросу, но и автоматически, при наступлении определенных событий, например, при выходе одного из контролируемых параметров за границы установленного диапазона. В этом случае передаваемая информация должна иметь статус «сигнала тревоги», который должен активизировать мобильное устройство, даже если оно находится в режиме ожидания, и привлечь внимание пользователя.

Общий сценарий работы сервиса должен быть следующим: при подключении удалённого устройства к сервису происходит его регистрация на сервере Ubiq Mobile, в результате которой устройству присваивается уникальный регистрационный код. Пользователь при подключении к сервису в начале сеанса работы вводит этот код со своего мобильного телефона, после чего получает возможность обмениваться информацией с удалённым устройством через сервер Ubiq Mobile.

В разрабатываемом приложении в качестве удаленных устройств выступают веб-камеры, установленные на компьютерах пользователей. Обмен данными с севером должен выполняться по протоколу TCP/IP и сервис должен предоставлять пользователю следующие функции:

- «Подключение» к веб-камере с мобильного телефона в любой момент времени;
- Получение статических изображений с веб-камеры на экран мобильного устройства через фиксированные интервалы времени;
- Задание интервала, через который с веб-камеры должны отправляться изображения;
- Остановка/Возобновление передачи изображений;
- Получение тревожных сигналов и активация мобильного телефона пользователя, даже если он находится в режиме ожидания.

Данная работа выполняется совместно с Юлией Гладышевой. Моя часть заключается в разработке фрагментов сервиса, отвечающих за

- Обеспечение возможности подключения удаленных устройств к серверу в любой момент времени
- Взаимодействие с удаленными устройствами (передача устройству управляющих команд от мобильного пользователя; прием и обработка информации, полученной от удаленных устройств)

2. Обзор существующих технологий и подходов

Обмен информацией между различными «умными» системами и мобильными устройствами, например, управление удаленными устройствами посредством мобильного телефона – задача, на которой акцентируют свое внимание многие разработчики мобильных приложений.

На данный момент наиболее распространенными подходами существует несколько основных подходов решению данной задачи и реализации мобильных сервисов в целом являются:

- Разработка специализированных мобильных приложений, предназначенных для работы на конкретных моделях телефонов;
- Использование SMS/MMS интерфейса;
- Использование Web/WAP интерфейса.

Рассмотрим, насколько каждый из них удовлетворяет следующим требованиям, которые существенны для активного использования мобильных сервисов в реальных условиях:

- Поддержка различных мобильных устройств и платформ;
- Поддержка медленных каналов связи;
- Сервисы должны быть интерактивными, обладать богатой функциональностью и хорошо проработанным интерфейсом;
- Обеспечение экономии трафика (в случаях тарификации по объему переданной/полученной информации).

Специализированные мобильные приложения зачастую обеспечивают достаточно высокий уровень интерактивности и богатую функциональность, но существенным недостатком таких приложений является сложность поддержки большого числа мобильных устройств и платформ.

Сервисы, основанные на SMS/MMS или WAP интерфейсе, обладают очень бедной функциональностью и отсутствием интерактивности.

Все больше разработчиков мобильных сервисов используют технологии Mobile Ajax, которая требует установки на телефоне одного из мобильных браузеров (Opera Mobile, Internet Explorer Mobile, Minimo, OpenWave, …). Данная снижает трату времени, но требует высокой производительности и мощности пользовательского мобильного устройства.

Соответственно, платформа Ubiq Mobile ориентирована на то, чтобы предоставить разработчикам возможность создания кросс-платформенных «легких» и в то же время интерактивных сервисов с богатой функциональностью и низкими требованиями к ресурсам.

3. Реализация

3.1. Общее описание решения.

Структурное решение для поставленной задачи изображено на рисунке рис. 2.

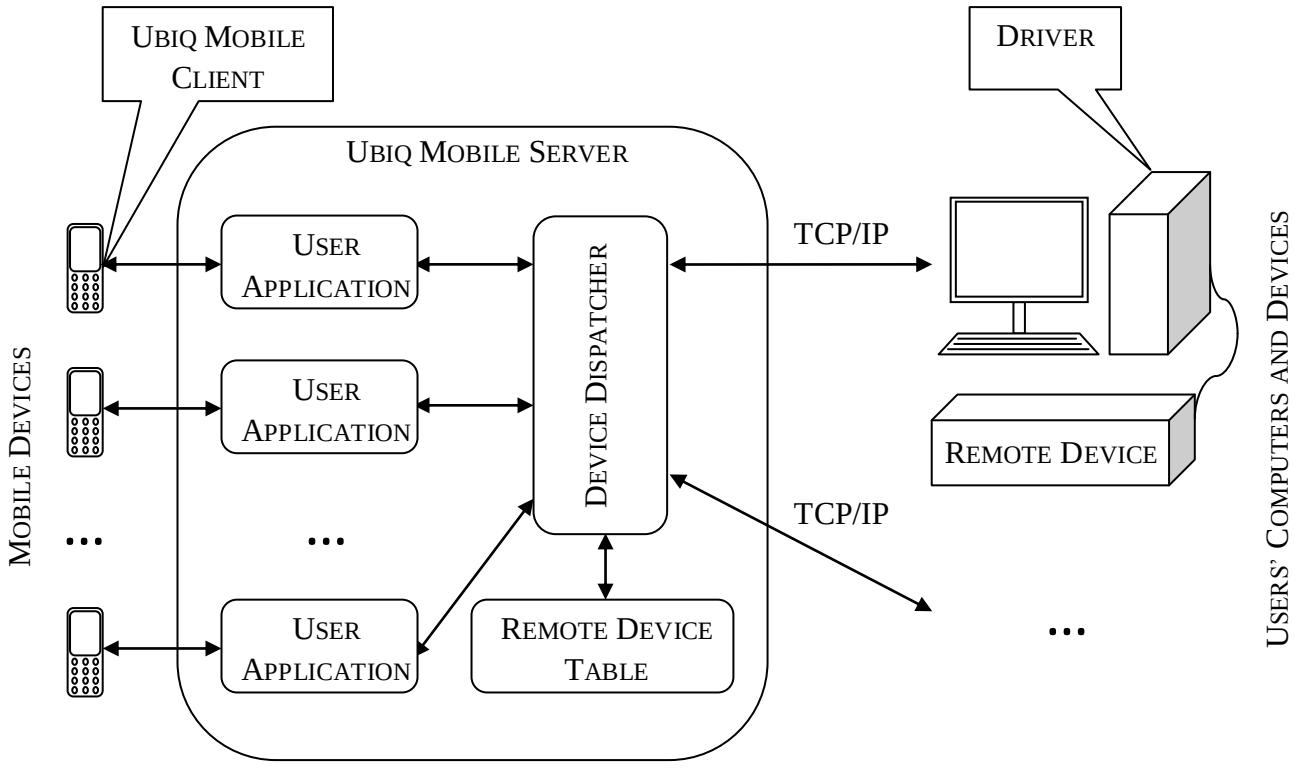

Рисунок 2. Структура мобильного сервиса.

Для реализации этой схемы используется два типа серверных приложений. Приложения первого типа, User Application, отвечают непосредственно за связь с пользователем и реализуют всю бизнес-логику, связанную с обработкой информации, получаемой от удаленных устройств. User Application запускается на сервере в отдельном экземпляре для каждого пользователя и соответствует понятию «пользовательской сессии». Приложение второго типа, Device Dispatcher, существует в единственном экземпляре и играет роль коммутатора между удаленными устройствами и соответствующими User Applications.

Device Dispatcher (или просто Диспетчер) запускается при инициализации сервера и постоянно находится в активном состоянии. Это необходимо для того, чтобы обеспечить возможность подключения к серверу удаленных устройств в любой момент времени, независимо от наличия соединений с пользователями, работающими с этими устройствами. Основными функциями Device Dispatcher являются регистрация устройств,

осуществление коммутации между ними и User Applications, а также удаление информации об устройствах при отмене пользователями подписки на сервис. При временном отключении устройств от сервера информация о них не удаляется и при повторном подключении соединение восстанавливается.

После запуска User Application устанавливает соединение с Device Dispatcher и передает ему регистрационный номер устройства, с которым пользователь хочет взаимодействовать посредством мобильного телефона. $Далее Device Dispatcher проверяет наличие соединения с данным устройством, готовность его к работе и в случае успеха организует обмен данными между User Application и этим удаленным устройством. Главные задачи User Application:

- Обрабатывать данные и отображать их в удобном виде на мобильном телефоне пользователя при помощи графического API;
- Получать команды от пользователя и пересылать их удаленному устройству.

Удаленное устройство взаимодействует с сервером через программную компоненту (драйвер), устанавливаемую на пользовательском компьютере. Обмен данными происходит по внутреннему протоколу, построенному поверх TCP/IP. Для этого компьютер должен иметь постоянное подключение к интернету, но выделенного IP-адреса не требуется, так как для полноценного функционирования сервиса достаточно наличия выделенного IP-адреса у сервера. Драйвер обрабатывает пользовательские команды и отправляет на сервер информацию о текущем состоянии устройства. Кроме того он устанавливает новые настройки и режимы работы и осуществляет контроль за возникновением «тревожных» событий, например, выход одного из параметров устройства за границы установленного диапазона. В этом случае драйвер должен автоматически отправить сообщение на сервер для немедленного оповещения пользователя.

Для взаимодействия между приложениями внутри сервера платформа Ubiq Mobile предоставляет стандартный механизм обмена сообщениями через почтовые ящики. Сообщения представляют собой структуры данных, произвольные с точки зрения языка программирования; в частности, они могут содержать ссылки на объекты, что избавляет от необходимости сериализации и предоставляет разработчикам бóльшую свободу в представлении информации. Поддерживается как синхронная, так и асинхронная отправка сообщений. Для приложений существует возможность автоматического оповещения при появлении новых сообщений в почтовом ящике.

3.2. Структуры данных

Для взаимодействия между удаленными устройствами и пользовательскими приложениями сервер поддерживает специальную структуру данных – таблицу зарегистрированных устройств (Remote Device Table). Каждая запись таблицы содержит

- Регистрационный номер устройства;
- Ссылку на удаленное устройство в виде сокета;
- Идентификатор пользовательского приложения.

Когда к серверу Ubiq Mobile подключается новое удаленное устройство, ему присваивается уникальный регистрационный номер, по которому пользователь с мобильного телефона в дальнейшем сможет к нему обращаться. Регистрационный номер вместе со ссылкой на текущий сокет, который выделяется при подключении каждого удаленного устройства, заносится в таблицу. При повторном подключении устройства его идентификационный номер не меняется, а ссылка на сокет может поменяться, что отражается в таблице.

Регистрационный номер представляет собой случайно выбираемое целое число, в реализованном демо-сервисе – это четырехзначное число.

Для более крупных сервисов с высокими требованиями к безопасности возможна реализация полноценного механизма аутентификации.

Когда пользователь с мобильного телефона подключается к сервису и вводит регистрационный номер определенного устройства, User Application посылает Device Dispatcher запрос на коммутацию с этим устройством. Device Dispatcher, в свою очередь, заносит в таблицу идентификатор User Application, и в случае, если в данный момент устройство соединено с сервером (то есть поле «Сокет» в соответствующей записи таблицы заполнено), отправляет пользовательскому приложению сообщение о готовности устройства к работе, в противном случае - сообщение об ошибке.

3.3. Протокол

Взаимодействие между Device Dispatcher и User Applications осуществляется посредством механизма обмена сообщениями через почтовые ящики, а между Device

Dispatcher и драйверами устройств - по специальному двоичному протоколу, построенному поверх TCP/IP. Но и в том, и в другом случае логической единицей обмена данными является команда унифицированной структуры.

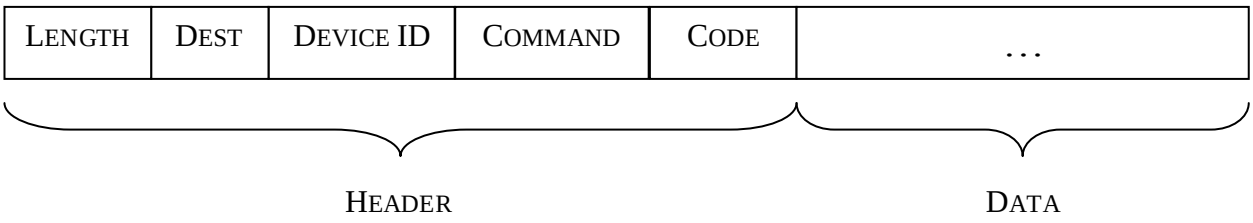

Рисунок 3. Структура команды.

Каждая команда содержит заголовок фиксированной длины, состоящий из 5 полей:

- **«LENGTH»** содержит общую длину команды (включая заголовок и данные). Под это поле отводится 4 байта.
- **«DEST»** - 1-байтовое поле, определяющее получателей команды.

  Одной из основных функций Device Dispatcher является коммутация между User Applications и драйверами устройств, поэтому команды от драйвера, предназначающиеся пользовательскому приложению должны автоматически, без детальной обработки, пересылаться получателю. Аналогично Device Dispatcher должен поступать с командами для драйвера от User Application.

  Данное поле используется Диспетчером для описанных целей, по его значению он определяет получателя команды. В случае если получатель - не сам Диспетчер, он пересылает сообщение указанному адресату.
- **«DEVICE ID»** содержит регистрационный номер устройства, представленный в виде 2 байтов.
- **«COMMAND»** представляет собой 1-байтовый код команды, определяющий тип и структуру передаваемой информации.
- **«CODE»** - дополнительное поле длиной в 1 байт. В некоторых командах оно используется для хранения статуса передаваемой информации (обычная, тревожная, настройки), в сообщениях об ошибках для передачи кода случившейся ошибки.

В итоге, общая длина заголовка каждой команды составляет 9 байт.

«DATA» - поле, непосредственно содержащее устройство-зависимые данные произвольной длины. Отдельного поля, содержащего длину данных не предусмотрено, так как эта величина может быть вычислена исходя из общей длины команды и фиксированной длины заголовка.

Все команды можно разделить на *команды общего назначение* и *команды, зависимые от устройств (device-dependent)*. Общими являются команды, одинаковые для всех устройств, например, запрос на регистрацию устройства или сообщение об ошибке. Устройство-зависимые команды зависят от особенностей конкретных устройств, к ним относятся, например, команды, связанные с передачей информации о текущем состоянии и настройках устройства. Предполагается, что каждый сервис может использовать дополнительные, устройство-зависимые команды для определенных целей.

Рассмотрим общие команды протокола.

3.3.1. Взаимодействие между Driver и Device Dispatcher

- RegistrationRequest

  При первом подключении к сервису устройство отправляет на сервер запрос на регистрацию, при этом в команде никаких параметров не передается, а поле **«DEVICE ID»** является пустым.

- RegistrationConfirmation

  Диспетчер отсылает эту команду драйверу удаленного устройства в ответ на его запрос о регистрации. Новый регистрационный номер содержится в поле **«DEVICE ID»** .

- DeviceReady

  Когда устройство готово к работе после получения регистрационного номера или после переподключения к серверу, драйвер отсылает эту команду Диспетчеру. Параметров команда не имеет.

- RegistrationCancel

При отмене пользователем подписки на сервис драйвер устройства отсылает на сервер запрос на отмену регистрации. Параметров команда не имеет.

- EndUserSession

  Диспетчер отправляет эту команду драйверу удаленного устройства, чтобы сообщить ему о завершении пользовательской сессии. Драйвер при получении данной команды должен остановить передачу данных. Параметров команда не имеет.

- Error

  Команда используется для отправки сообщения об ошибке. В поле **«CODE»** хранится код ошибки, и в случае необходимости в поле данных - дополнительная информация.

3.3.2. Взаимодействие между Driver и User Application

- DataRequest

  В любой момент User Application может запросить у драйвера данные о текущих параметрах устройства и его настройках. Поле **«CODE»** используется для определения типа запрашиваемой информации (информация о текущих значениях характеристик устройства или настройки). Дополнительные параметры, зависящие от конкретного устройства (например, идентификатор под-устройства, чьи настройки запрашиваются пользователем) содержатся в поле данных.

- Data

  С помощью команд этого типа Driver отправляет на сервер данные о текущем состоянии устройства, о его настройках и об установленном режиме работы. Поле **«CODE»** используется для определения статуса информации (обычная, сигнал тревоги или настройки). Сами данные хранятся в поле **«DATA»** в формате, зависящем от конкретного устройства.

- SetSettings

User Application может менять текущие настройки удаленного устройства посредством отправки команд данного типа. Новые значения параметров хранятся в поле данных в формате, зависящем от конкретного устройства.

- UASuspend

  В случае возникновения кратковременного разрыва связи с мобильным пользователем User Application отправляет Диспетчеру данную команду. Диспетчер пересылает её драйверу устройства, но не удаляет из таблицы зарегистрированных удаленных устройств, так как предполагается, что связь с пользователем в скором времени восстановится. Параметров команда не имеет.

- UAResume

  Когда после кратковременного разрыва связи с пользователем подключении с ним восстанавливается, User Application отсылает Диспетчеру данную команды. Диспетчер пересылает эту команду драйверу соответствующего устройства, чтобы тот в свою очередь возобновил передачу данных с интервалом, установленным пользователем до разрыва связи.

- Error

  Команда используется для отправки сообщения об ошибке. В поле **«CODE»** хранится код ошибки, и в случае необходимости в поле данных - дополнительная информация.

3.3.3. Взаимодействие между Device Dispatcher и User Application

- DeviceReady

  В случае готовности определенного удаленного устройства к работе эта команда отсылается Диспетчером пользовательскому приложению в ответ на его запрос этого устройства (команда DeviceRequest). Параметров команда не имеет.

- EndUserSession

User Application отправляет данную команду Диспетчеру при завершении пользовательской сессии (платформа Ubiq Mobile позволяет приложениям определять свою собственную реакцию на возникновение данных событий). Диспетчер при получении данной команды удаляет идентификатор приложения из таблицы зарегистрированных устройств и пересылает эту команду драйверу соответствующего удаленного устройства, чтобы тот в свою очередь остановил передачу данных. Параметров команда не имеет.

- Error

  Команда используется для отправки сообщения об ошибке. В поле **«CODE»** хранится код ошибки, и в случае необходимости в поле данных - дополнительная информация.

3.4. Сервис передачи изображений с удаленной веб-камеры

Описанная архитектура построения системы и структура протокола являются универсальными и не зависят от специфики конкретных устройств. В разработанном сервисе передачи изображений с удаленной веб-камеры реализована именно эта структура. Драйвер, установленный на пользовательском компьютере, отправляет захваченные камерой изображения на сервер через определенный временной интервал. Этот интервал является настраиваемым параметром устройства, который пользователь может изменять. Если задать нулевой интервал, передача изображений приостанавливается до тех пор, пока не будет получена команда с ненулевым интервалом. В сервисе используются три дополнительные, устройство-зависимые команды:

- SetInterval

  С помощью этой команды устанавливается значение временного интервала, через который драйвер устройства будет отсылать изображения на сервер. В качестве единственного параметра команды выступает целое число, обозначающее значение нового интервала.

- GetImage

  С помощью этой команды User Application может отправить запрос на мгновенное получение нового изображения. Параметров команда не имеет.

- SendImage

  С помощью этой команды драйвер устройства отправляет на сервер изображения, захваченные веб-камерой. Изображения в формате JPEG сохраняются в поле данных в виде массива байт.

Для общих целей таких как, регистрация, сообщения о готовности устройства и ошибках, использовались общие команды. Вместо описанных устройство-зависимых команд также можно использовать команды общего назначения. Таким образом, команда установки интервала могла быть реализована с помощью команды SetSetings, где единственным параметром было бы значение интервала, через который драйвер должен отсылать изображения на сервер. А вместо команд GetImage и SendImage можно использовать команды DataRequest и Data соответственно.

3.5. Реализация компонента Device Dispatcher

Мною был полностью реализован компонент сервиса, Device Dispatcher.

Device Dispatcher представляет собой приложение постоянно запущенное на сервере, поддерживающее работу со специальной структурой данных, таблицей зарегистрированных устройств. Каждая запись таблицы содержит

- Идентификатор устройства, представленный как целое число
- Номер сокета представленный как целое число
- Идентификатор пользовательского приложения. Для идентификации удобно использовать ссылку на почтовый ящик приложения.

Взаимодействие Диспетчера с драйверами удаленных устройств по протоколу TCP/IP происходит посредством TCP/IP сервиса, встроенного API, предоставляемого платформой Ubiq Mobile. Диспетчер находится в режиме постоянного ожидания прихода информации от удаленных устройств. Когда некоторое устройство подключается к серверу и отправляет информацию в виде команд описанного протокола, TCP/IP сервис принимает её и вместе со ссылкой на текущий сокет передает Диспетчеру. Диспетчер в свою очередь обрабатывает эти данные соответствующим образом и снова переходит в режим ожидания сообщений от удаленных устройств.

В то же время сообщения от пользовательских приложений Device Dispatcher принимает асинхронно с помощью механизма почтовых ящиков.

Основная логика работы Диспетчера заключается в обработке сообщений приходящих к нему, с одной стороны, от удаленных устройств, с другой стороны, от пользовательских приложений.

Независимо от отправителя сообщения, Диспетчер обязательно считывает два поля «LENGTH» и «DEST». В зависимости от значения поля «DEST» Диспетчер либо пересылает команду целиком адресату, либо продолжает считывать остальные поля команды и выполнять соответствующие действия.

Общие команды, предназначающиеся Диспетчеру:

- RegistrationRequest
- RegistrationCancel
- DeviceReady
- DeviceRequest
- EndUserSession

Общие команды, формируемые Диспетчером:

- RegistrationConfirmation
- EndUserSession
- DeviceReady
- Error

3.6. Сложности в процессе разработки

Относительно серьезные сложности при разработке сервиса передачи изображений с веб-камеры возникли только при интеграции его с платформой по следующим причинам:

- Сложность настройки среды для отладки

Для полноценной отладки сервиса необходимо на одной машине использовать одновременно несколько сред разработки, это Visual Studio, Carbide C++ на базе Eclipse для имитатора мобильного клиента и виртуальной машины Java для выполнения компонента «Драйвер» для веб-камеры.

- Платформа Ubiq Mobile (в частности TCP/IP API и API для почтовых ящиков) имеет недостаточный уровень отлаженности и устойчивости.

- Так как клиентская часть сервиса, драйвер, реализована на языке программирования Java, то на сервере при обработке информации, приходящей от веб-камеры, возникает необходимость преобразования порядка байтов в числах.

3.7. Режим работы сервиса «много устройств – много пользователей»

Описанная структура решения предполагает работу сервиса в режиме «одно устройство – один пользователь».

Реализация режима «много устройств – много пользователей» выходит за рамки данной курсовой работы, но было проведено предварительное изучение этой проблемы. Результаты предварительного анализа показали, что режим «много устройств – один пользователь» может быть реализован достаточно легко, так как в этом режиме работы не затрагивается структура протокола и структура данных (таблица зарегистрированных устройств). На мобильном телефоне пользователя отображается несколько вкладок, соответствующих разным удаленным устройствам, каждое из которых имеет уникальный идентификатор. В зависимости от того, с какой вкладкой в данный момент оперирует пользователь, User Application посылает команды драйверу соответствующего устройства. В заголовке каждой команды существует поле с идентификатором удаленного устройства, поэтому командам, приходящим от драйверов устройств и Диспетчера, пользовательское приложение может однозначно сопоставить вкладку, на которой необходимо отобразить полученную информацию. Очевидно, в этом случае нужна переработка структуры пользовательского приложения, в то время как изменений в логике работы компонента Device Dispatcher а также в используемых структурах данных не требуется.

Более сложная ситуация оказывается, когда несколько пользователей хотят работать с одним устройством (режим «одно устройство – много пользователей»). Возникает вопрос о разграничении прав пользователей на доступ к устройству. Очевидно, что в общем случае не должно быть ситуаций, когда несколько пользователей пытаются установить каждый свои настройки (например, текущее значение какого-либо из параметров или границы, при выходе за которые, драйвер формирует «тревожное» сообщение). Поэтому полный доступ к устройству должен получать только один пользователь. Для других пользователей возможно два варианта:

- Они имеют право получать информацию о состоянии устройства и его настройках, запрос на которые отправляет выделенный пользователь, но сами не могут устанавливать настройки или интервал, через который драйвер отправляет информацию о текущем состоянии устройства;
- Пользователи данной группы имеют право устанавливать свой собственный интервал, но не имеют право настраивать устройство;
- Пользователи данной группы имеют право устанавливать свой собственный интервал, а также выполнять некоторый набор действий безопасных с точки зрения многопользовательности.

Для первого варианта оптимальным решением является организация широковещательной рассылки (broadcasting). Выделенный пользователь устанавливает интервал, через который драйвером на сервер отправляется информация о текущем состоянии устройства, может устанавливать новые настройки и т.д. Остальные пользователи являются лишь «подписчиками на рассылку» и имеют возможность только получать и просматривать данные, запрошенные выделенным пользователем.

Для реализации второго варианта в общую модель нужно ввести понятие «интервал», через который драйвер отправляет данные на сервер.

Если обработка интервалов и отправка информации определенным пользовательским приложениям осуществляется с помощью драйвера, то необходимо вводить новые идентификаторы для пользовательских приложений, так как существующие внутренние для этих целей не подходят. В качестве идентификатора можно использовать интервал, через который данный конкретный пользователь хочет получать информацию, или некоторый уникальный номер. Стоить отметить, что в этом

случае драйвер должен хранить информацию о пользовательских приложениях и соответствующих интервалах и уметь «накладывать» разные интервалы друг на друга.

Если данную функциональность реализовывает Диспетчер, тосуществующих внутренних идентификаторов пользовательских приложений окажется достаточно, но возникает вопрос о способе получения данных от устройства: либо получать их только по запросу, либо установить такой интервал передачи данных с устройства, чтобы каждый пользователь мог получать изображения в своем режиме, однако при этом, придется организовывать буферизацию данных на серверной стороне.

Поддержка режима «много устройств – много пользователей» является логичным и полезным усовершенствованием разработанного сервиса, поэтому в будущем будет производиться проработка архитектуры реализации этой схемы на платформе Ubiq Mobile.

4. Заключение

В рамках данной курсовой работы на базе платформы Ubiq Mobile был реализован сервис передачи изображений с удаленной веб-камеры на мобильный телефон пользователя, работающий в режиме «одно устройство – один пользователь». В частности, был реализован компонент сервиса – серверное приложение Device Dispatcher и произведена его интеграция с платформой.

Также был разработан структурный шаблон для реализации целого класса приложений, обеспечивающих доступ к удаленным устройствам. Этот шаблон не зависит от особенностей конкретных устройств и систем, и в дальнейшем на его основе предполагается создать универсальную библиотеку классов, которая упростит разработку других сервисов данного типа и сэкономит время программистов.

Интересными направлениями в дальнейшем развитии сервиса являются поддержка режима «много устройств – много пользователей» и реализация полноценной системы аутентификации.

На данный момент сервис способен работать лишь с теми устройствами и системами, которые передают информацию в дискретном режиме (например, посылают изображения через определенный интервал времени). В будущем планируется осуществить поддержку потоковых данных (в том числе видео), что существенно расширит возможности сервиса.

5. Список литературы

[1] **Реализация связи клиента с сервером для платформы Ubiq** [Дипломная работа] / авт. Левкина М. А. - Санкт-Петербург : Санкт-Петербургский Государственный Университет, Математико-механический факультет, Кафедра системного программирования, 2009.

[2] **Mobile Services for Access to Remote Devices on the Basis of Ubiq Mobile Platform** [Conference] / auth. Gladisheva Yulia, Onossovski Valentin, Tumanova Kristina // Proceedings of 7th Conference of Open Innovation Framework Program FRUCT. - Saint-Petersburg : [б.н.], 2010.

[3] **Ubiq Mobile – a New Universal Platform for Mobile Online Services** [Конференция] / авт. Onossovski Valentin Terekhov Andrey // Proceedings of 6th Seminar of Finish-Russian University Cooperation (FRUCT) Program. - Helsinki : [б.н.], 2009.