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
Гладышевой Юлии Сергеевны

Научный руководитель В. В. Оносовский

Санкт-Петербург

2010

Оглавление

1. Введение и постановка задачи 3
1.1. Введение 3
1.2. Постановка задачи 5
2. Обзор существующих технологий и подходов 8
3. Реализация 9
3.1. Общее описание решения. 9
3.2. Структуры данных 11
3.3. Протокол 12
3.4. Webcam сервис 16
4. Реализация компоненты User Application 18
5. Реализация компоненты Driver 19
6. Сложности в процессе разработки 20
7. Заключение 21
8. Список литературы 22

1. Введение и постановка задачи

1.1. Введение

Данная курсовая работа заключается в разработке универсального структурного шаблона для реализации мобильных сервисов управления удаленнымиустройствами на базе платформы Ubiq Mobile. В качестве примера таких сервисов был реализован Webcam сервис, который позволяет пользователю получать изображения с удаленной веб-камеры на мобильный телефон. Webcam сервис обладает большинством структурных и технических особенностей, то есть является полноценным представителем сервисов для управления удаленными устройствами.

Описанный проект выполнен совместно с Кристиной Тумановой (студенткой СПбГУ, 445 группы) как часть исследовательского проекта Ubiq Mobile.

Платформа Ubiq Mobile инкапсулирует большинство деталей, специфичных для конкретных мобильных телефонов с помощью предоставления разработчикам простого API. Вследствие чего программисты без опыта разработки мобильных приложений могут использовать Ubiq Mobile для создания высокопроизводительных мобильных сервисов и приложений с богатой функциональностью и низкими требованиями к ресурсам. Эти сервисы будут эффективно работать на различных моделях телефонов и в различных условиях связи, включая относительно медленные каналы(GPRS, EDGE).

Ключевая идея платформы Ubiq Mobile - терминальная (“mainframe-like”) архитектура, где вся бизнес-логика приложения выполняется на стороне сервера, а мобильные телефоны выступают больше в роли графических терминалов, чем клиентов. С одной стороны, такой подход сильно сокращает требования к ресурсам, упрощает инкапсуляцию зависимых от устройств технических деталей и делает всю структуру мобильных приложений более простой и прозрачной. С другой стороны, терминальная архитектура накладывает некоторые ограничения на разрабатываемые сервисы. Наиболее значимое ограничение касается невозможности offline работы.

Обмен данными между сервером и мобильным клиентом выполняется через специальный бинарный протокол, построенный поверх TCP/IP. Использование специального протокола вместо одного из стандартных дает дополнительные возможности для оптимизации трафика и использования нестандартных алгоритмов сжатия графических данных.

Платформа накладывает на разрабатываемые приложения следующие ограничения: относительно статический характер пользовательского интерфейса (среднее время отклика платформы составляет около 1 секунды, что достаточно медленно по сравнению с локальными анимированными приложениями) и невозможность offline работы. Для большинства информационных и “business-related”сервисов пользовательский интерфейс не является важным. Невозможность offline работы частично компенсируется с помощью механизма сохранения пользовательских сессий, реализованного на базе платформы Ubiq Mobile.

Преимущества и ограничения платформы определяют класс приложений, для которых разработка на базе Ubiq Mobile наиболее целесообразна и оправдана, например, приложения для управления удаленными “intelligent” устройствами и системами с помощью мобильного телефона. “Intelligence” означает, что устройство либо само имеет непосредственный доступ в интернет, либо подключено к компьютеру, который имеет постоянное интернет-соединение. Типичные примеры таких устройств и систем являются:

- Веб-камеры, установленные на компьютере пользователя и посылающие на мобильный телефон пользователя изображения.
- Системы видеонаблюдения, передающие видео с нескольких камер на мобильный телефон.
- Система управления загородным домом (офисом, магазином, гаражом и т.п.). Такие системы обычно имеют компьютерный интерфейс и интернет-соединение, но мобильный интерфейс обычно предоставляется через неудобные SMS сообщения. Используя мобильный online сервис, пользователь имеет возможность получать информацию о системе (свет, температура воздуха, влажность) в графическом виде на экране мобильного телефона и посылать некоторые команды уделенной системе.
- Системы безопасности и сигнализации, например, для машины. Если будет обнаружена попытка проникновения, система пошлет специальный “тревожный” сигнал на телефон пользователя с детальной информацией. Пользователь может послать системе специальную команду (например, заблокировать двигатель).

Все мобильные сервисы для управления удаленными устройствами имею ряд общих особенностей:

- Двусторонний обмен информацией (от удаленной системы к мобильному телефону – данные о состоянии системы, от мобильного телефона к пользователю - команды)
- Непрерывный режим работы удаленного устройства
- Возможность «подключения» к устройству в любой момент времени
- Возможность отправки устройством «тревожных» сигналов при наступлении “важных” событий, которые требуют немедленного оповещения пользователя.

Эти общие особенности позволяют при разработке конкретных приложений для управления удаленными устройствами на базе платформы Ubiq Mobile использовать структурные паттерны. В качестве примера таких сервисов был реализован Webcam сервис, который позволяет пользователю получать изображения с удаленной веб-камеры на мобильный телефон. Webcam сервис обладает большинством структурных и технических особенностей сервисов для управления удаленными устройствами. В процессе его реализации мы пытались разработать решение, которое может быть применимо практически ко всем сервисам этого класса.

1.2. Постановка задачи

В основном мы исходил из следующей общей модели сервиса:

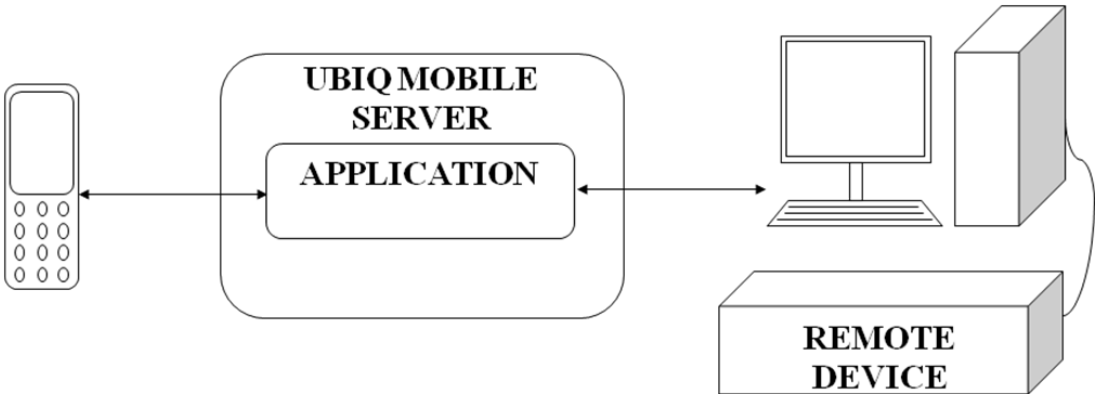

Рисунок 1. Общая модель сервиса

Пусть есть некоторое удаленное устройство, подключенное к компьютеру, имеющему постоянный доступ к интернету. С одной стороны, удаленное устройство производит некоторую информацию (текущее состояние системы или изображения, полученные с камеры), с другой стороны, оно может получать некоторые команды от компьютера, к которому оно подключено, и выполнять их.

Пользователь в любой момент времени подключается к сервису и получает информацию с удаленного устройства на экран своего мобильного телефона. Если необходимо, пользователь может отправлять некоторые управляющие команды устройству, используя элементы графического интерфейса.

Также важно обеспечить “standby” режим для сервиса - с минимальной затратой энергии и трафика, сервис может в любой момент быть активированным при получении специальных “alert” сообщений от удаленного устройства (например, при выходе некоторых измеряемых параметров за установленный диапазон).

Когда компьютер, к которому подключено удаленное устройство, соединяется с Ubiq Mobile сервером в первый раз, выполняется процедура регистрации. Ее результатом является уникальный регистрационный номер, присвоенный данному устройству. Если пользователь хочет получить доступ к конкретному устройству, он должен выбрать соответствующий сервис из списка всех сервисов и ввести регистрационный номер нужного устройства. Если устройство с таким номером существует и доступно в данный момент, то пользователь соединяется с устройством имеет возможность получать данные с устройства. Если устройство с данным номером недоступно (выключено или нет доступа в интернет), то регистрационный номер сохраняется и может быть использован снова после включения устройства. Регистрационные номера могут быть освобождены и использованы снова только после отмены регистрации на сервере Ubiq Mobile.

В разработанном нами приложении в качестве удаленного устройства выступает удаленная веб-камера, установленная на компьютере пользователя. Сервис предоставляет следующие возможности:

- Получение статических изображений с веб-камеры и отображение их на экране мобильного телефона.
- Изменение интервала, через который с веб-камеры отправляются изображения
- Возможность останавливать/возобновлять отправку изображений

В рамках данной курсовой работы мной были реализованы следующие компоненты сервиса:

- Серверное приложение, отвечающее непосредственно за связь с пользователем и реализующее всю бизнес-логику, связанную с обработкой информации, получаемой от удаленных устройств
- Программная компонента, устанавливаемая на пользовательском компьютере, отвечающая за работу с веб-камерой и за отправку данных на сервер.

2. Обзор существующих технологий и подходов

Проблема обмена информацией между удаленными устройствами и мобильными телефонами привлекает внимание многих разработчиков мобильных приложений. В настоящее время существует несколько общих подходов к решению данной проблемы: создание “ad hoc” native приложений, использование SMS/MMS интерфейса или Web/WAP интерфейса. Рассмотрим создание специализированных мобильных приложений. С одной стороны, такие приложения обладают высокой интерактивностью, богатой функциональностью и оптимально используют трафик. С другой стороны, огромное разнообразие мобильных устройств и платформ делает их разработку очень дорогой. Сервисы, использующие SMS/MMS или Web/WAP интерфейс, характеризуются отсутствием интерактивность и бедной функциональностью.

Еще один подход, сочетающий в себе богатые пользовательские возможности, высокую эффективность и низкие требования к ресурсам – это создание таких приложений на базе Ubiq Mobile платформы.

3. Реализация

3.1. Общее описание решения.

Структурное решение для представленной задачи изображено на рисунке 2.

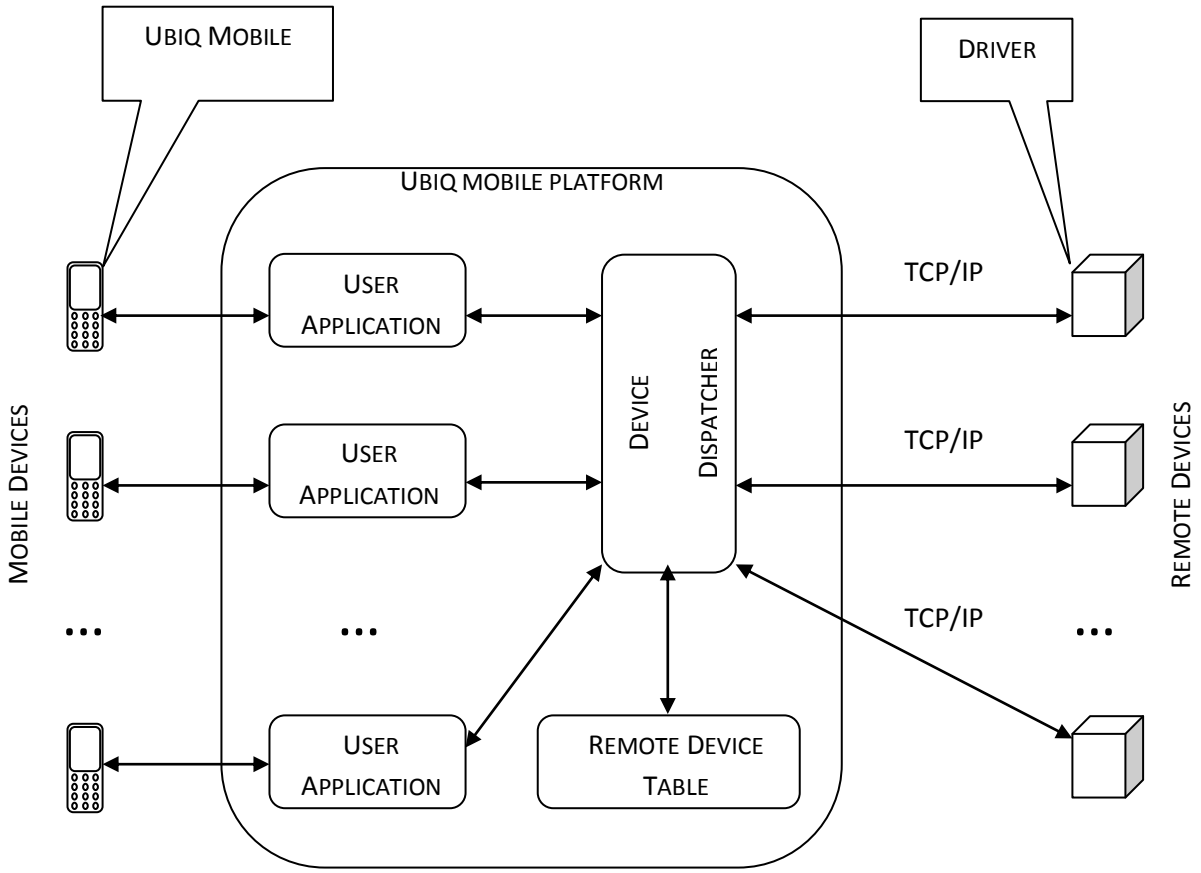

Рисунок 2. Архитектура.

Для реализации этой схемы используется два типа серверных приложений. Приложения первого типа, User Applications, 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 (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), которую пользователь устанавливает на своем 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, они могут содержать ссылки на объекты, что избавляет от необходимости сериализации. 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 (RDT). Каждая запись таблицы содержит регистрационный номер устройства, ссылку на удаленное устройство в виде сокета и идентификатор пользовательского приложения.

Когда к Ubiq Mobile Server подключается новое удаленное устройство, ему присваивается уникальный регистрационный номер, по которому пользователь с мобильного телефона сможет к нему обратиться. 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 Webcam сервисе – это четырехзначное число.

Для более крупных сервисов с высокими требованиями к безопасности возможна реализация полноценного механизма аутентификации, но в нашем приложении этого не требуется.

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, 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 единой структуры.

Одной из 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.

Все команды можно разделить на «general» и «device-dependent». «Device-dependent» команды зависят от особенностей конкретных устройств, к ним относятся, например, команды, связанные с передачей информации о текущем состоянии и настройках устройства. «General» являются команды, одинаковые для всех устройств, например, запрос на регистрацию устройства или сообщение об ошибке.

В общем случае логическая структура команды имеет следующий вид:

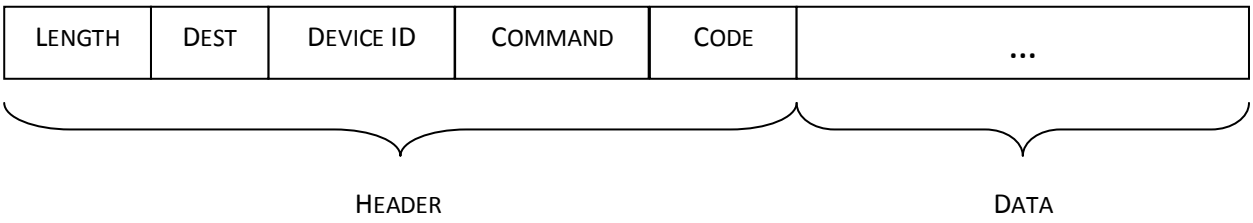

Рисунок 3. Структура протокола.

Каждая команда содержит заголовок, состоящий из 5 полей:

- «LENGTH» содержит общую длину команды
- «DEST» определяет получателей команды. Команда может предназначаться Device Dispatcher, определенному User Application или драйверу определенного удаленного устройства. Только для Device Dispatcher это поле является

существенным, по его значению он определяет получателя. В случае если получатель - не сам Device Dispatcher, он пересылает сообщение указанному адресату

- «DEVICE ID» содержит регистрационный номер устройства, который ему выдается при первом подключении к серверу
- «COMMAND» представляет собой код команды
- «CODE» - дополнительное поле. В некоторых командах оно используется для хранения статуса передаваемой информации (обычная, тревожная, настройки), в сообщениях об ошибках для передачи кода случившейся ошибки

«DATA» - поле, непосредственно содержащее данные, зависящие от конкретных команд и устройств. Отдельного поля, содержащего длину данных не предусмотрено, так как эта величина может быть вычислена исходя их общей длины команды и фиксированной длины заголовка.

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 уже получило регистрационный номер и готово к работе, драйвер отсылает эту команду диспетчеру. Параметров команда не имеет

- RegistrationCancel

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

- UserSessionEnd

  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.

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

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

При завершении пользовательской сессии User Application оповещает Device Dispatcher. 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. Webcam сервис

Описанная структура системы и структура протокола являются универсальными и не зависят от специфики конкретных устройств. В разработанном сервисе передачи изображений с удаленной веб-камеры реализована данная 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.

Специфичными командами протокола для данного сервиса являются:

- SetInterval

  Команда используется User Application для установки нового значения временного интервала, через который на сервер будут отправляться изображения. Параметр команды - значение интервала.

- GetImage

  Команда используется User Application для запроса на получение изображения с веб-камеры. Параметров команда не имеет.

- SendImage

Команда используется Драйвером для отправки изображения, захваченного веб-камерой, на сервер. Изображение в формате JPEG записывается в виде массива байт в поле «DATA».

Все остальные команды, а именно, команды, связанные с регистрацией веб-камер, сообщениями о готовности к работе и сообщениями об ошибках, были реализованы с помощью “general” команд.

4. Реализация компоненты User Application

Мною была реализовано серверное приложение User Application, которое создается в отдельном экземпляре для каждой пользователя. User Applications отвечают непосредственно за связь с пользователем и реализуют всю бизнес-логику, связанную с обработкой информации, получаемой от удаленных устройств.

User Application взаимодействует с Драйвером через Device Dispatcher. Взаимодействие User Application с Device Dispatcher осуществляется с помощью механизма обмена сообщениями. User Application может принимать сообщения от Диспетчера, формировать и отсылать их Драйверу или Диспетчеру.

Команды, формируемые User Application для Device Dispatcher:

- UASuspend
- UAResume
- EndUserSession
- DeviceRequest

Команды, формируемые User Application для Driver:

- DataRequest
- SetSettings
- SetInterval
- GetImage

Дизайн интерфейса пользовательского приложения был реализован с помощью низкоуровневых графических примитивов платформы (нарисовать прямоугольник, выбрать цвет, вывести текст и др.).

5. Реализация компоненты Driver

В прошлом году в качестве курсовой работы был реализован прототип Драйвера и оттестирован в автономном режиме. В этом году он был существенно переработан с учетом требований платформы Ubiq Mobile и в соответствии с разработанным структурным шаблоном для реализации сервисов управления удаленными устройствами.

Driver устанавливается на компьютере, имеющем постоянное подключение к интернету. С помощью Driver удаленное устройство взаимодействует с сервером. Обмен данными происходит по внутреннему протоколу, построенному поверх TCP/IP. 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, в случае которых отправляет сообщение на сервер для немедленно оповещения пользователя.

Команды, формируемые Driver для User Application

- Data

Команды, формируемые Driver для Device Dispatcher:

- RegistrationRequest
- RegistrationCancel
- DeviceReady

6. Сложности в процессе разработки

В процессе разработки были обнаружены следующие сложности:

- Недостаточный уровень отлажненности и устойчивости самой платформы Ubiq Mobile (в части messaging API и TCP/IP API) .

- Сложность настройки среды для отладки (Visual Studio с многоязыковым solution – C++ и C#, Carbide C++ на базе Eclipse для имитатора клиента и java для имитации драйвера, все они общаются через локальный TCP/IP)

7. Заключение

При выполнении данной курсовой работы был решен ряд архитектурных и технических задач:

- Спроектирован и разработан универсальный двоичный протокол для обмена данными с удаленными устройствами, допускающий специализацию для учета особенностей конкретных устройств
- Разработан программный интерфейс для работы с протоколом, позволяющий адаптировать его к использованию для управления другими удаленными устройствами
- Разработана архитектура и дизайн User Application
- Существенно переработана компонента Driver с учетом требований платформы Ubiq Mobile

Разработанный сервис передачи изображений с удаленной веб-камеры в целом интегрирован и отлажен в составе платформы Ubiq Mobile. Данная версия сервиса поддерживает работу в режиме “одно устройство – один пользователь”. В будущем мы планируем добавить поддержку одновременной работы нескольких пользователей с несколькими устройствами.

Был разработан универсальный структурный шаблон для реализации мобильных сервисов управления удаленными устройствами на базе платформы Ubiq Mobile. На основе данного шаблона в будущем планируется создать общую библиотеку классов, которая может быть использована для разработки подобных сервисов.

Также в будущем планируется поддержка передачи данных не только в дискретном режиме (например, статических изображений с веб-камеры), но и поддержка передачи потоковых данных (например, видео).

8. Список литературы

[1] **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.

[2] **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.