Mobile Services for Access to Remote Devices on the Basis of Ubiq Mobile Platform

Yulia Gladisheva, Valentin Onossovski, Kristina Tumanova
St.-Petersburg State University,
St.-Petersburg, Russia
yulia.gladisheva@gmail.com, v.onossovski@gmail.com, kristina.tumanova@gmail.com

Abstract

The paper describes a prototype of the mobile service that lets users to receive images from remote webcams to their mobile phones. The implementation of particular webcam service is considered as an illustration of more general approach that is applicable to the whole class of mobile applications designed to manage remote devices. This work has been conducted as part of the Ubiq Mobile research project that is being executed in St.Petersburg State University. Information exchange between remote "intelligent" devices of various kinds and users' mobile devices as well as control of remote devices and systems via mobile handsets is an important problem that attracts close attention of mobile applications' developers. At present, the two most popular ways to solve this problem are either development of "ad hoc" native mobile applications or using functionally poor SMS/MMS interface. Creation mobile applications on the basis of Ubiq Mobile platform can be considered as an efficient compromise combining rich user experience, high efficiency and low development cost. The general scheme of implementation of mobile services for access to remote devices on the basis of Ubiq Mobile platform is presented in the paper; detailed structural patterns are proposed.

INDEX TERMS: MOBILE ONLINE SERVICES, REMOTE DEVICE MANAGEMENT, STRUCTURAL PATTERNS, UBIQ MOBILE PLATFORM

I. INTRODUCTION

The project described in the paper is a part of Ubiq Mobile research project [1]. The main objective of Ubiq Mobile project is to create easy-to-use universal framework for developing mobile online applications. The Ubiq Mobile platform encapsulates most mobile-specific technical details by providing developers with simple and straightforward API so that many programmers without special mobile experience could use Ubiq Mobile to create modern distributed mobile applications and services. Ubiq Mobile platform allows developers to easily create highly efficient, functionally rich services with low resource requirements that are able to work efficiently on wide range of mobile devices in different mobile network conditions including relatively slow channels (GPRS, EDGE).

The key idea of Ubiq Mobile platform is terminal ("mainframe-like") architecture where all the business logic of the applications is implemented on server side and mobile devices act as graphical terminals rather than clients. On the one hand, such an approach sufficiently reduces resource requirements (particularly, provides mobile traffic savings), simplifies encapsulation of "device-dependent" technical details and makes the whole structure of distributed mobile application more transparent and straightforward. On the other hand, terminal architecture imposes several restrictions on developing mobile applications. The most significant restriction relates to practical impossibility of offline work.

The data exchange between server and mobile client is performing via proprietary binary protocol built over TCP/IP. The usage of custom binary protocol instead of one of standard

protocols gives additional opportunities to mobile traffic optimization by minimizing redundancy and using non-standard algorithms of graphical data compression.

The main restrictions that the platform imposes on developing applications are: relatively static character of user interface (the average response time is around one second that is slow enough, especially in comparison with “local” animated applications) and inability to work offline. Both of these restrictions are fundamental properties of any terminal system. On the other hand, for most information- or business-related online services animated UI is not so important, and inability of offline work is partially compensated by user sessions persistence mechanism implemented in Ubiq Mobile platform.

All the considered advantages and restrictions of the platform make it well suited for implementation of important class of target mobile applications, related to remote control of various “intelligent” devices and systems from users’ mobile phones. The “intelligence” of the remote device means that either the device itself is permanently connected to the Internet (e.g., via HTTP protocol), or the device is connected to PC that has permanent Internet connection. The typical examples of such devices (and appropriate services) are the following

- Webcams connected to user’s stationary computers that send images on user’s mobile phone
- Video surveillance systems that continuously broadcasts video from several cameras on user’s mobile device
- Remote control for house (storehouse, garage etc.) automation systems. Such systems usually have computer interfaces and Internet connections but mobile interface is typically provided via SMS that is not so convenient. Using mobile online service, the user would be able to receive information about the system (light, air temperature, air moisture, fire system etc.) in graphical mode on the screen of his mobile device, and send commands to the system using UI controls
- Security and warning systems, for example, for a car. When detecting intrusion attempt, the system sends alert signal to the user’s mobile device with detailed information. The user can send an appropriate command to the system (e.g., to block car engine) immediately after receiving alert signal

Most existing solutions in this area implement one of two basic approaches: either development of “ad hoc” native mobile applications for access to remote devices (that is expensive and hard to support different mobile devices and platforms), or using universal SMS/MMS-based interface that evidently provides too poor functionality. Mobile web-based solutions are less popular and rarely used due to their lack of interactivity.

All the services related to access to remote devices have some set of features in common – information exchange between mobile user and remote system in both directions (from remote system to the user – information about system state, from user to the remote system – control commands), work of the remote system persistent mode and ability for the user to connect to the system from mobile device at any time. In case of important events that require immediate user intervention, the remote system should be able to send alert signals.

Such commonality of features lets developers to use common structural patterns when developing concrete mobile services on Ubiq Mobile platform. We chose webcam service for implementation as demo application. In spite of its relative simplicity, this service possesses most of structural and technical peculiarities that are typical for all services of the discussed class. So, during implementation of this specific application, we tried to form more general approach that can be applied to most of services related to management of remote device from mobile phone.

II. TASK OVERVIEW

Basically, we came from the following general service model:

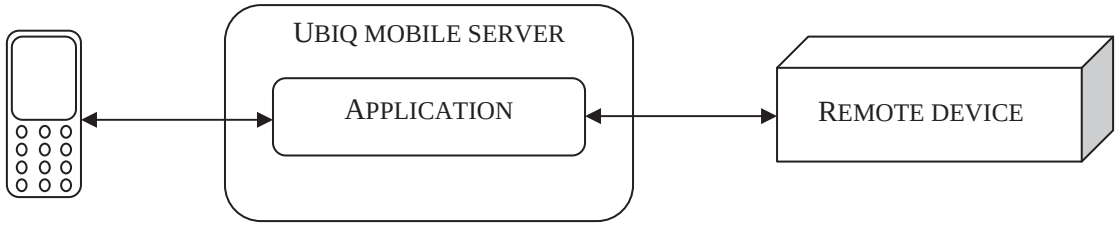

Fig. 1. Concerned model

There is some remote device connected to a computer that has permanent Internet access. On the one hand, the remote device produces some information (like current state of home automation system or images received from a webcam); on the other hand, it can receive control commands from the computer and execute them. The typical scenario of mobile user's work with device looks like following: the user connects to the service at any time and receives information from the remote device presented on the screen of his mobile phone or communicator in appropriate graphical form. If necessary, the user can send control commands to the device using control elements of graphical interface (e.g., change some parameters and press the button on the screen) and see the results on his mobile device. In addition, it's quite important to provide "standby" mode for the service – with minimal energy and traffic consumption, but ready to be activated when receiving special "alert" messages from the remote device – for example, in case of going some parameters of the device beyond the established boundaries.

The general workflow of the service is the following. When PC that connected with the remote device connects to the Ubiq Mobile server first time, a registration procedure is being fulfilled. The result of registration is the unique registration code assigned to this remote device by dispatcher application running on Ubiq Mobile server. When the mobile user wants to get access to the device, he choose appropriate service in the list of all available services displaying on his phone screen and, after service's request, enters registration code of the device he needs to access. If the device with such code exists and is available at the moment, the service "connects" users' mobile phone to the remote device allowing it to receive data as well as send control commands. If the device is turned off or disconnects from the Internet, its registration code is still preserving and may be used again for access to the device after its turning on. The registration codes can be released and re-used only if the device is explicitly unregistered on Ubiq Mobile server.

For work with the service, the remote device should be provided with a special program component (let's call it driver – don't confuse with OS device drivers) installed on the PC to which device is connected. Drivers of remote devices communicate with special application – device dispatcher – that is permanently running on Ubiq Mobile server. Device dispatcher application is responsible for performing registration functions and monitoring current state of all remote devices registered in the service.

In the particular case of webcam service we have web-camera connected to user's PC as remote device. The service provides the following main features:

- Acquiring static images from the webcam and displaying them on the screen of user's mobile device in fixed time intervals;
- Setting value of time interval from user's mobile device;
- Ability to stop/resume image transfer by user's request.

III. SOLUTION DESCRIPTION

The general structural scheme that can be used for implementation of services of the discussed class on Ubiq Mobile platform is shown on Fig. 2:

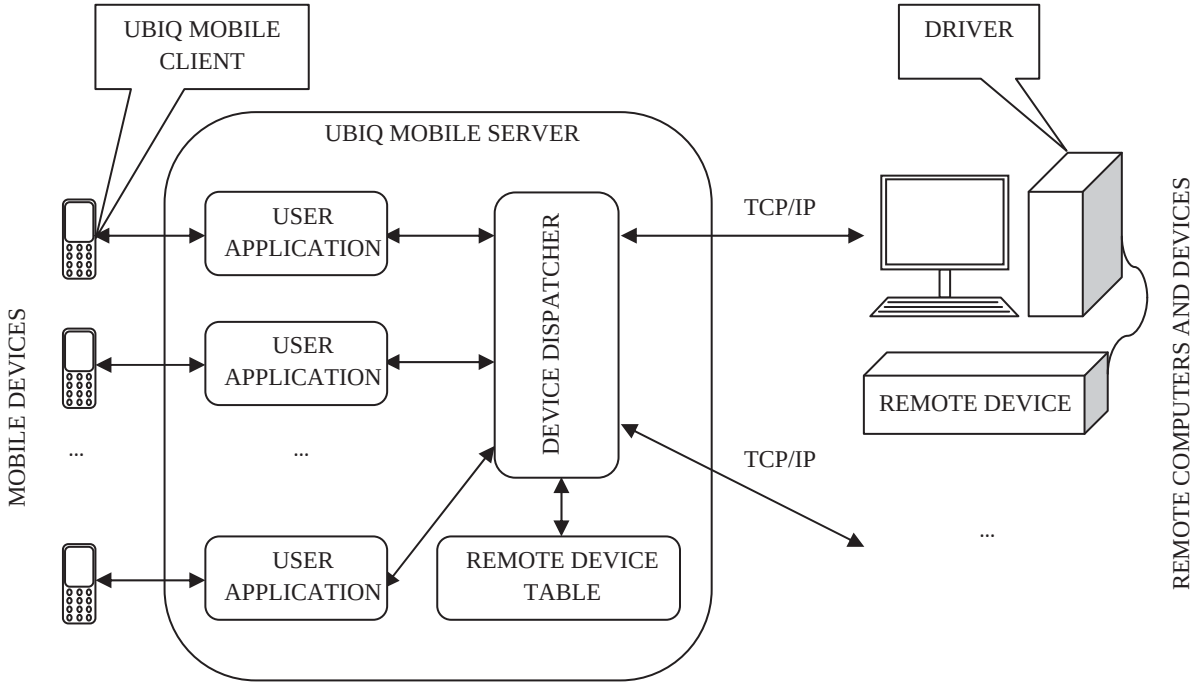

Fig. 2. Mobile service implementation structure

This implementation scheme assumes two main types of server applications. Applications of the first type, User Applications, are responsible for interaction with mobile users. They implement all business logic and user interface related to processing and displaying of information receiving from remote devices. At any time, many User Application instances can run simultaneously on Ubiq Mobile server, corresponding to different user sessions. The application of the second type - Device Dispatcher (or just Dispatcher) exists in a single instance. The Dispatcher is started during server launch and works all the time until server stops. The main functions of the Device Dispatcher are registration of remote devices, checking their current availability and providing data exchange between remote devices and User Applications. The Device Dispatcher is also responsible for deletion of all information about remote device when registration of this device is being cancelled. When short disconnections of remote devices occur, the information isn't deleted and the device is getting available after re-connection.

After starting, User Application establishes new connection with the Dispatcher and sends registration number of the device that user wants to access. Then the Device Dispatcher checks, if the device is connected to the server and ready for work. If everything is OK, Dispatcher is ready to perform data transfer between User Application and the remote device. User Application's functions are

- Processing data from remote device and represent it on user mobile's screen in appropriate graphical form
- Receiving commands from the user and passing them to remote device

Remote device interacts with the server through a program component, driver, installing on the computer to which device is connected. Data exchange is performing via proprietary binary protocol built over TCP/IP. In order to perform this interaction the computer should have a permanent Internet access, but dedicated IP address is not required – it's sufficient to have dedicated IP for Ubiq Mobile server. The device driver processes user's commands and sends to the server information about current state of the device. Also the driver can set device settings and send control commands to the device. In urgent situation (for example, if one of device characteristics is going beyond fixed bounds), the device driver should send alert message to the server for immediate user notification.

For inter-application cooperation inside the server Ubiq Mobile platform provides a standard mechanism of message exchange through mailboxes. In terms of programming language, these messages may contain arbitrary data structures, particularly object references, so programmers have more freedom in data representation in comparison with "linear" messages. The mechanism of mailboxes supports both synchronous and asynchronous messaging models. There is an opportunity for applications of automatic notification when new messages appear in mailbox.

A. Data structure

Ubiq Mobile server supports a special data structure – Remote Devices Table (RDT) – for interaction between remote devices and Users Applications. Each RDT record contains a registration number of specific remote device, a reference to this device (in the form of socket number of current device connection) and an identifier of corresponding User Application.

When new remote device connects to Ubiq Mobile Server, a unique registration number is assigned to it. Thereafter user can refer to the device by this number from his mobile phone. Registration number and a reference to current socket (extracting when some device connects) are added to RDT. When device reconnects, its registration number isn't changed but reference to socket can be changed.

Registration number represents a random integer number. In realized demo-service it is a random four-digit number.

For larger services with high security requirements, it's possible to realize full-fledged mechanism of authentication but in our application it isn't needed.

When user connects to the service from mobile phone and enters the registration number of required device, User Application sends to the Device Dispatcher request for establishing commutation with this device. The Device Dispatcher adds to RDT the identifier of User Application and if the device is connected to the server at this moment, Device Dispatcher sends a message about readiness of device to User Application. Otherwise, it sends error message.

B. Protocol

The interaction between Device Dispatcher and User Applications is performing via inter-application messaging mechanism while for data transfer between Device Dispatcher and remote devices the custom binary protocol built over TCP/IP is used. But, in both cases the logical structure of data exchange is the same – the passed information is presented in form of logical commands of uniform structure.

One of important functions of the Device Dispatcher is to provide commutation between User Applications and drivers of remote devices, so the commands that Driver sends to a certain User Application should be transferred to the recipient automatically, without detailed

processing. In the same manner, the Device Dispatcher should handle commands sending from User Applications to remote device's drivers.

All commands can be divided into two categories – general commands and device-dependent ones. The first category includes all commands whose structure is the same for all types of remote devices, such as registration requests or error messages. The structure of commands of the second category is device-specific and can sufficiently differ for different kinds of remote devices. For example, all commands related to information exchange between User Applications and remote devices are obviously device-dependent.

The general logical structure of the command is shown on the Fig. 3.

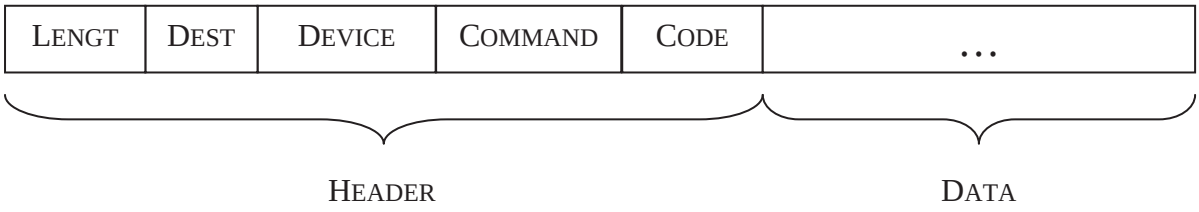

Fig. 3. Structure of command.

Each command includes a standard header of fixed structure consisting of 5 fields:

- ***Length*** field contains length of the whole command (including header and data)
- ***Dest*** field determines the recipient (destination) of the command. The command can be destined to the Device Dispatcher, User Application or remote device Driver. The information from this field is used by Device Dispatcher for the commutation purposes.
- ***DeviceID*** field contains registration code of the device
- ***Command*** field contains command code, which determines type and structure of transferring information.
- ***Code*** is an auxiliary field that may contain either status of transferring information (regular data, urgent data, settings) or error code, depending on type of command

"DATA" is a section of command containing device-dependent data of variable length. There is no special *Data length* field – the length of data section is calculated using *Length* field.

General commands

The protocol assumes the following set of general commands that are device-independent.

Interaction between remote device drivers and Device Dispatcher:

- RegistrationRequest
  This command is sent by remote device's Driver to the Device Dispatcher during first connection of the device to the service.
- RegistrationConfirmation
  The Device Dispatcher sends this command back to the remote device's Driver as response to registration request. The unique registration code is stored into *DeviceID* field.
- RegistrationCancel
  When the user doesn't want to access remote device via mobile service anymore, the remote device driver sends to the server a request for cancellation of its registration.
- DeviceReady

When the remote device is ready to work with the service after either registration or reconnection of the device, the device driver informs Device Dispatcher about this fact by sending DeviceReady command.

- EndUserSession
  This command is sent from Device Dispatcher to the remote device driver to inform the latter that the mobile user session has been suspended or terminated. After receiving this command, the device driver stops data sending from the device.
- Error
  The command is used for sending error messages between Device Dispatcher and remote device drivers. This command uses *Code* field for storing error code and, if necessary, data section for detailed information about error.

Interaction between User Applications and device drivers:

- DataRequest
  User Application can submit a request to the device driver for retrieving information about current values of device parameters, settings and data at any time. The *Code* field is used to determine the type of requested information (either settings or regular data). The additional parameters depending on particular device (for example, an identifier of the sub-device whose settings are requested) are stored in the data section.
- Data
  The command is used for sending remote device parameters, settings and data from device Driver to User Application (for instance, in respond to DataRequest command). The *Code* field is used to determine the status of information (regular data sending, urgent data sending or settings). The data itself is stored in the data section of the command in device-dependent format.
- SetSettings
  User Application may change current settings of the remote device by sending this command to the device driver. The new values of device parameters are stored in the data section in device-dependent format. For example, for webcam service the only parameter is used – interval of sending images from webcam to User Application.
- Error
  The command is used for sending error messages between Device Dispatcher and remote device drivers. This command uses *Code* field for storing error code and, if necessary, data section for detailed information about error.

Interaction between User Applications and Device Dispatcher

- DeviceRequest
  User Application submits this command to the Device Dispatcher to get access to the remote device (assuming that the device is already registered in the service). The registration code of requested device is stored in *DeviceID* field.
- DeviceReady
  This command (already described above) is sent by Device Dispatcher to User Application in response to DeviceRequest command.
- EndUserSession
  The User Application sends this command to the Device Dispatcher in case of its suspending, disconnection or termination (Ubiq Mobile platform API lets applications to define their own reactions to such kind of events). Furthermore,

Device Dispatcher passes this command to the remote device driver as described above.

- Error
  The command is used for sending error messages between User Applications and the Device Dispatcher. This command uses *Code* field for storing error code and, if necessary, data section for detailed information about error.

Device-dependent commands

The protocol structure described above is universal and doesn't depend on device-specific details. The protocol assumes that each concrete service may use additional, device-dependent commands for special purposes. In our developed webcam demo-service three additional commands are used:

- SetInterval
  The webcam Driver sends images to the User Application with fixed intervals. The user can set new value of interval. If the new value is equal to zero, the webcam Driver suspends image transferring until getting new non-zero value.
- GetImage
  Request from the User Application to the webcam Driver to send image from the webcam immediately.
- SendImage
  Driver sends an image to the server with fixed time interval or after user's request. The image is stored in data section.

For general purposes – like webcams registration, connection to a particular webcam, error messages etc. – general commands are used.

Actually, there were no needs for this particular webcam demo service to introduce device-specific commands – it would be possible to use general commands instead. So, function of setting interval could be implemented using general SetSettings command, with an interval value as the only parameter. Instead of GetImage and SetImage commands one could use general DataRequest and Data commands correspondingly. We chose to use device-dependent commands here for illustration purposes.

IV. CONCLUSION

The work has been performed in the Department of Software Engineering of St. Petersburg State University within a framework of Ubiq Mobile research project. Currently there is a working version of webcam service that supports "one webcam – one user" working mode.

The plans for the nearest future include extending webcam service functionality by adding support for "multiple webcams – multiple users" working mode. In addition, we plan to assemble all device-independent components in the universal class library that could be used as a template for development mobile services providing access to the remote devices.

REFERENCES

[1] Onossovski Valentin, Terekhov Andrey, "Ubiq Mobile – a New Universal Platform for Mobile Online Services", Proceedings of 6th Seminar of Finish-Russian University Cooperation (FRUCT) Program, Helsinki, Finland, 3-6 November 2009, pp. 96-105.