RemoteUI: A High-Performance Remote User Interface System for Mobile Consumer Electronic Devices

Daniel Thommes[*†], *Student Member, IEEE,* Qi Wang[*], *Member, IEEE,*
Ansgar Gerlicher[†], *Member, IEEE*, Christos Grecos[*], *Senior Member, IEEE*
[*]University of the West of Scotland, Paisley, UK and [†]Stuttgart Media University, Germany

***Abstract*—Innovative applications become feasible with solutions for remotely controlling mobile devices over the air. To realize these applications, efficient technologies for transferring the devices' user interfaces are required. Existing remote user interface (UI) solutions however were not built for the mobile world. Rich, touchable user interfaces on battery-powered devices combined with low available bandwidth and high network latency will highlight their problems. We propose a new solution called RemoteUI, which works with abstract UI descriptions and their remote replication. Experiment results show that RemoteUI significantly outperforms the existing popular Virtual Network Computing (VNC) approach, and it is highly efficient in terms of required bandwidth when compared with VNC.**

I. Introduction

Today we are facing a growing mobile computing market, and solutions for remote control of mobile equipment such as smartphones or other consumer electronic devices introduce numerous promising applications. Fig. 1 shows such an application in the car integration industry. The driver controls his/her sophisticated smartphone via the car's onboard "thin client" computer with larger screen. The latter mirrors the mobile phone's user interface, transmitted via Bluetooth or WiFi, and allows controlling it with touch gestures. Whilst providing a safer and more convenient way for the driver to enjoy the applications running in the mobile phone, the thin client computer is affordable since no expensive cellular communication facilities or other smart functionality is required.

Fig. 1. In-car-remote scenario

The aim of our *RemoteUI* project is to develop technologies for realizing a high-performance remote user interface system and protocol for mobile scenarios. In these scenarios special conditions like low available bandwidth, high network latency and low energy availability exist. Under these conditions existing protocols do not perform well enough because they were designed for mouse and keyboard based input methods and wired networks. In this paper we prove the concept that a mobility-optimized protocol can outperform an existing solution regarding bandwidth requirements while still transferring the rich user interfaces of modern smartphone applications.

Currently only remote UI server implementations based on Virtual Network Computing (VNC) [1] are available for all major mobile platforms. VNC uses the Remote Framebuffer Protocol [2] to transmit the server's frame buffer content as pixel information to the client. For the in-car scenario shown above the CE4A consortium specified a VNC-based solution called *Terminal Mode* [3]. In this context, VNC is the most reasonable technology to compare with our implementation.

II. The RemoteUI System

The *RemoteUI system* is designed to optimally fit into the mobile world. As a first approach for reducing the required bandwidth of the system, we use view descriptions and graphic primitives to describe and transfer the user interface. These graphical objects build the so-called *UI-tree*, which is replicated between server and client. The replication process typically starts with the initial transfer of the complete tree, e.g. when a new screen is opened on the server. Each node in the tree is carrying a unique ID with which server- and client-initiated manipulations are realized. Fig. 2 exemplifies the process of *remote tree manipulation*. The left tree is transformed into the right by replacing text and image information in different nodes. To accomplish this, the IDs and the corresponding attributes of the nodes are addressed in the commands.

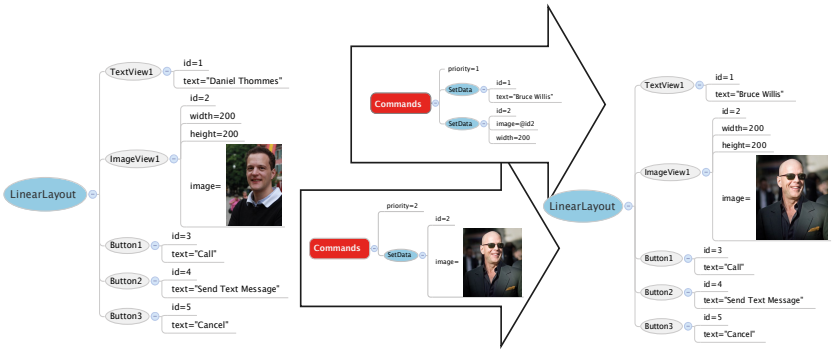

Fig. 2. UI-tree and tree manipulation

Additionally the commands carry *priority information*. The server transmits objects with high priority first, which intentionally causes a delay for lower prioritized objects especially in low bandwidth scenarios. Because the client sequentially processes received objects this results in an *incremental rendering* of the remote user interface. The aim of this technology is to increase the speed with which the system reacts on user input. Part of our research is the development of algorithms to determine the optimal priority from input parameters such as visibility, data amount, operational importance and available bandwidth. An example for a possible prioritization is shown in Fig. 3.

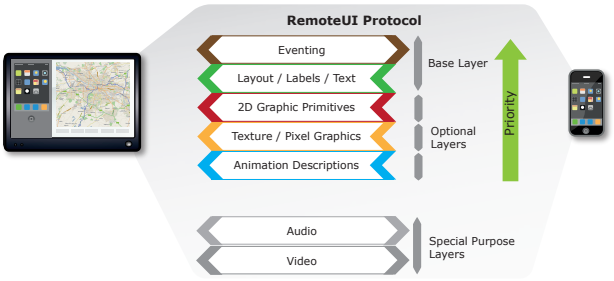

Fig. 3. Example of prioritization in RemoteUI's hierarchical protocol

III. Implementation and Experimental Results

We have developed a RemoteUI framework for the Android platform. Applications that shall be remotely controlled must be linked and compiled with the *RemoteUI App Library*. These "remotable" applications on startup locally connect to the *RemoteUI Server*, which handles all network connections. Remotable applications give the RemoteUI Server access to their user interfaces. The system this way works *without root privileges*. The *RemoteUI Client* application is used to connect via network to a RemoteUI Server and is able to control its remotable applications. For the serialization of the UI-tree and manipulation commands, we use the Hessian 1.0 protocol [4], which is platform independent and produces very compact binary representations.

To demonstrate the advantages of the UI-tree approach over existing solutions, we carried out an experiment comparing the required bandwidth of the RemoteUI protocol with that of VNC. For this experiment, we implemented two exemplary Android applications: a simple calculator application and a list demo application displaying a list of 50 items. We then simultaneously connected to the device over WiFi using RemoteUI and VNC. We executed the test applications and captured the data traffic between servers and clients.

Both applications were started as part of the experiment causing the initial transfer of their UIs to the connected clients. The calculator was used to carry out a simple equation, whilst the list demo application was used to scroll down the list of items. Both actions were taken on the serving device to exclude traffic from touch events on one of the clients only.

The results of the bandwidth consumption and data measurements are shown in Fig. 4, Fig. 5 and Table 1. The experiments show that the RemoteUI protocol has significantly lower bandwidth requirements than VNC does. One reason for this is the usage of the UI-tree description and tree manipulation mechanisms instead of transferring pixel information. Additionally the server-initiated updates used by RemoteUI are a superior approach compared with VNC's request-response mechanism that caused traffic during our experiments even if there were no server side updates.

While Fig. 4 shows small peaks in the RemoteUI curve that result from the calculator's text field being updated when entering the equation (seconds 15-40), Fig. 5 only shows one peak produced by the initial transfer of the list view. This can be explained with RemoteUI's smart handling of lists. All items, also the invisible ones, are transferred initially. Scrolling down the list does not cause any network traffic and can

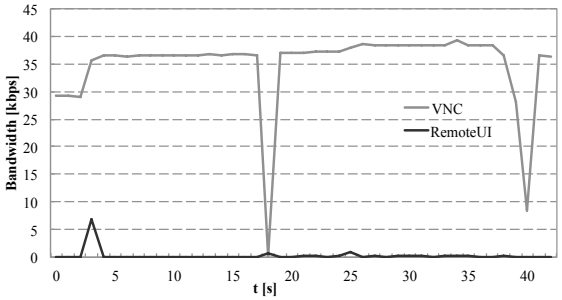

Fig. 4. Bandwidth comparison for the calculator application

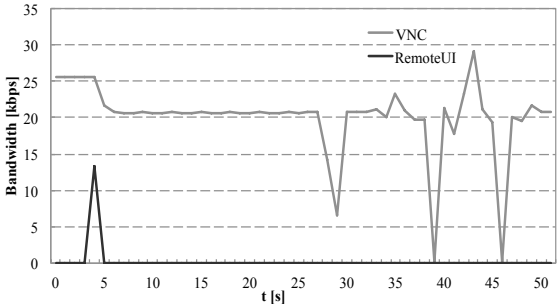

Fig. 5. Bandwidth comparison for the list demo application

TABLE 1 . ABSOLUTE DATA REQUIREMENTS OF VNC AND REMOTEUI

Experiment	Duration [s]	VNC data [bytes]	RemoteUI data [bytes]	RemoteUI / VNC ratio
Calculator	42.08	1,503,311	11,814	0.79%
List demo	51.56	1,047,687	13,265	1.30%

be executed independently on the client and server side. VNC in contrast has to transmit many pixel data for this use case.

IV. Conclusion

We have presented a new system for remote user interfaces specialized for mobile scenarios. We have shown that the proposed RemoteUI system outperforms the existing solution VNC substantially. The experiments show that our protocol used only 0.78-1.3% of VNC's bandwidth.

In our future work, we will continue with the implementation of prioritization, caching and prefetching mechanisms. The latency problem will be addressed by introducing special client-only view variants for a better usability in high-latency scenarios. We also assume that there is a relation between required bandwidth and energy consumption. Analyzing this relationship will also be a part of our future work.

References

[1] T. Richardson, Q. Stafford-Fraser, K. Wood, and A. Hopper, "Virtual network computing", *IEEE Internet Comput.,* vol. 2, no. 1, pp. 33-38, 2002.

[2] T. Richardson and J. Levine, "The Remote Framebuffer Protocol", IETF RFC 6143, Mar 2011.

[3] Nokia Inc. and CE4A, "TerminalMode v1.0 Specification." [Online]. Available: http://www.terminalmode.org/files/Zipped_Release.zip. [Accessed: 26-Jul-2011].

[4] Caucho Technology, Inc., "Hessian 1.0.2 Specification." [Online]. Available: http://hessian.caucho.com/doc/hessian-1.0-spec.xtp. [Accessed: 25-Jul-2011].

Picture credits:

- Fig. 1: Car dashboard W123 230 Automatic from 1977, first series, Wikipedia user HLW, licensed under CC-BY 3.0
- Fig. 2: Bruce Willis at a premiere in London, Caroline Bonarde Ucci, licensed under CC-BY 3.0