Accelerated Desktop Control – acdeco

Petri Heinilä
0035087

May 30, 2013

1 Introduction

Idea with Accelerated Desktop Control (acdeco) is to remote control desktop- or other fixed computer via wireless radio channel. Control inputs are constructed on mobile device, smart-phone, by sensing movement or position of the device.

The emphasis in input design is not to use conventional mobile device touch-screen display. Use of the touch-screen display requires user visual attention, and on the remote control situations the user attention is on the target device, the desktop computer display. Therefore the control input sensing should relate only for user hand activity.

Mobile device runs on a limited battery energy. The touch-screen display energy consumption is markable, usually second in device operations after wireless connections. This makes two effects in use. First mobile device energy saving activity is aggressive, constantly disabling the touch-screen display. User would need to turn the display continuous on, if using remote control through display. Second if display would be continuously on, the energy drain would shorten the mobile device activity time considerably.

Also, by nature control signal messages are trivial and does not really need touch-screen manipulation.

Targeted audience for acdeco are desktop or laptop users carrying mobile device along them. The mobile device provides additional input methods for controlling desktop. Also short range controlling of the fixed device is feasible, eg. controlling slide-show presentation on laptop computer.

Another audience group is multi-media computer user for the entertainment content. The acdeco would replace conventional remote control device.

2 Design

On 1 is presented the system functional parts. The deployment of the acdeco system is divided into two areas desktop-side and mobile-side.

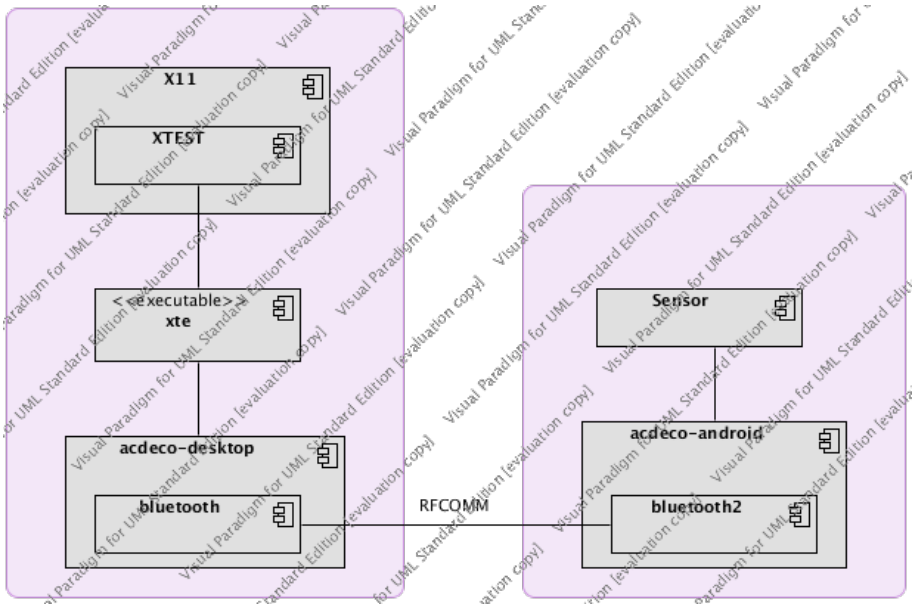

Figure 1: Systems parts

2.1 Desktop-side

acdeco-desktop is a application specific program. The program works as a server in communication where mobile side connects. The programs communicates the mobile-side and decodes the control messages. For a control message external program, xte, is called.

xte is a external program from xautomation package. xte generates XTEST [Drake(1992)]message that can be used to feed synthetic X11 input events.

X11 is a display server with XTEST protocol extension. X11 holds the application windows that are to be remote controlled.

2.2 Mobile-side

acdeco-android is program on mobile device that handles sensor reading, sensor data interpreting, converting sensor signals into control messages and bluetooth RFCOMM [blu(2001)]communication handling.

sensor provides sensor sample flow. Accelerometer sensors are used.

2.3 Protocol design

The developed and used protocol in acdeco is SASCII (Stupid American Standard Code for Information Interchange). The messages interchanged in remote control operation are signal type message carrying no payload information. Two types messages are needed protocol flow control messages and remote control messages. The assumed amount of produced mobile device inputs is under 200 so under 200 control messages are needed.

Used underlaying protocol is RFCOMM[blu(2001)] transport layer in a Bluetooth stack.

Message boundary on SASCII is one octet containing code for flow- or control message.

SASCII is a extension from ASCII protocol codes where codes 32dec – 255dec are used for signal messages and 0dec – 31dec are for flow messages. Flow messages follow ASCII protocol code semantics. Semantics for control messages are free, but it is recommend to use ASCII code that related produced input in desktop-side. For example if space character is assigned to the controlled desktop application then ASCII code 32dec should be used for that.

For flow messages the ASCII SYN 22dec, synchronous idle, is used for connection opening and device vicinity heartbeat signaling. On heartbeat signaling both devices send SYN message once a second, and if receiving device does not get SYN message inside second time frame the connection is considered disconnected. This is because transport layer, RFCOMM, SAP (Service Access Point), socket API, does not provide interface to react transport messaging or timeouts.

3 Implementation

Desktop-side implementation was done with Python 2.7 [van Rossum(1993)] language. PyBlues python module was used for bluetooth connectivity for RFCOMM transport and SDP[blu(2001)] service advertising. PyBluez was the reason to use old python 2.7 instead python 3.3, because PyBluez development is dead (latest update 2009) and was not ported to python 3+ series.

xte command line tool from xautomation debian package was used to send XTEST requests to the X11 server.

On mobile-side android 4.1.2 [and()]environment with Java language[jav()] was used to implemeny acdeco-android part.

Linux distribution for bluetooth management was problematic. At first RFCOMM was not working. After resolving problem the solution was to set: /etc/bluetooth/main.conf: DisablePlugins = pnat, to get RFCOMM to work. Resolving this take about 12 hours time.

On android side, there was problem, the program did SIGSEGV on bluetooth socket close(). Therefore no proper disconnection management were implemented.

4 Conclusions

The demonstration program showed out the viability to use mobile device movement sensors in remote controlling of fixed computers without used touch-screen attention.

References

[and()] Android Web Site. URL `http://www.android.com`.

[jav()] Java Language and Virtual Machine Specifications. URL `http://docs.oracle.com/javase/specs/`.

[blu(2001)] Specification of the Bluetooth System, Volume 1: Core, v1.1. Bluetooth SIG, February 2001.

[Drake(1992)] Kieron Drake. XTEST Extension Library. Technical report, 1992. URL `http://ftp.xfree.org/pub/XFree86/4.4.0/doc/PDF/xtestlib.pdf`.

[van Rossum(1993)] G van Rossum. *An Introduction to Python for UNIX/C Programmers.* PhD thesis, 1993. URL `citeseer.ist.psu.edu/vanrossum93introduction.html`.