Differences Between Mobile and Desktop Computers

Mobile computers (smartphones and tables) have both commonalities with desktop computers (they are computers) and differ widely in several different respects. We want to identify these differences, and discuss what implications they have, from a perspective of application development, and in a broader context. We can raise the question if we are using mobiles for the same purposes as desktops, should they be treated equally, and if not, why not?

Technology has only recently converged on mobiles (previously considered as telephones) to make them into computers. They are addressing many tasks that in the past belonged to the domain of the desktop. There has been a shift, and in many cases sharing, of responsibilities. Common tasks that are conveniently addressed by both classes are sending emails, reading news and other online content, interacting with a variety of web-services, and a host of other information services.

Mobiles, because they are mobile, small, and carry sensors, have also enabled a domain of new abilities. Taking photographs for instance, or being reminded by the use of a calendar service of ones tasks, belong uniquely to the mobile. On the other hand, most of us spend more time interacting on a desktop than a mobile because many tasks are more suitably achieved as such. These involve tasks that are computationally more demanding, rely on network stability and bandwidth, use greater screen-area for visualisation, and are better addressed by more efficient interaction techniques (the keyboard and mouse).

Considering the question of equality by comparison, mobiles have recently experienced a much more rapid series of improvements than the desktop. The desktop can do pretty much the same as it could a couple of years ago, which is far from the case with mobiles. The recent iPhone 4S for instance has a 1GHz dual-core processor and 640x960 screen-resolution compared with a 620 MHz single processor, 320x480 screen of the original iPhone from 2007. *Device Heterogeneity* is even more pronounced on the Android system, that run on devices manufactured by over 10 mainstream technology companies[1]. These companies are free to vary any number of intrinsic properties such as screen-sizes, processing capabilities, sensor ranges and networking support. While the rate of development is different from desktops, note also that general processing capability are several years behind between latest high-end systems. Comparisons in processing performance is difficult to make in terms of just quantities such as processing speed, because it depends on additional factors such as power-supply and bus-architecture. A useful method are compute-oriented benchmarks such as Linpack[2] which will give a fair comparison in terms of a specific problem (in this case solving sets linear equations). The iPhone 4S scores comparably to a Pentium III from 2002[3]. The sufficient point to appreciate is not *exactly* how they compare, but that they are significantly less capable.

Figure 1: The mobile era has just begun, completely new domains of current and near-future applications are emerging [1].

Further, considering how mobiles have evolved in terms of software systems. The first significant entry (of a phone that seriously started to contend with the desktop) was the iPhone. The software model of the iPhone is hugely proprietary, and initially dominating, in a similar way that Windows has been for the desktop. The iPhone has since lost much of its dominating position to other similar software models,

[1]http://en.wikipedia.org/wiki/Comparison_of_Android_devices#Other_manufacturers

[2]http://www.roylongbottom.org.uk/linpack%20results.htm

[3]http://www.xyster.net/blog/?p=40

most notably Android. It is natural to ask why has this happened, and what are the differences that make phones, as a computing platform, lend itself more readily to a diversity of different operating systems. One reason may be due to their networked nature and general adoption of the thin-client oriented approach to tasks, incompatible devices can still interface in the same way with available services. Another reason is probably because of the more rapid evolution of phones have allowed multiple entrants early on. Since Windows which had time to establish itself and became generally more useful because of a critical mass of users and was widely supported by a rich set of applications.

In terms of *programming environment* and application support, mobiles have adopted a radically different approach to application distribution, and application policy. Due to the safe assumption that mobiles have network connectivity, applications are not downloaded and installed in the conventional sense. Instead Apple revolutionised the concept of application marketplaces that allow third party developers to distribute their mobile applications, and which in addition (apart from development) is the *only* supported interface for adding new software functionality to a mobile. This paradigm is principally mediated by a top-down approach to the design of the entire OS with regards to this purpose. All mainstream mobile systems support a high-level runtime environment that are accessed from standard development libraries. This restricts development of applications for a specific system to a single language (e.g. Java for Android, and Objective-C for iPhone). This may have drawbacks if there are inherent limitations in the language (for example parallel programming may be better done in functional languages), but has benefits in its singular focus and wide community support. It also restricts the programmer to obey by the functionality supported by the environment, for example low-level device manipulation or memory operations are not possible. But the functionality is in general well maintained by the providers to cater for virtually any conceivable task, for example allowing general access to network interfaces and sensors.

Returning to the topic of *sensors and mobility*, we identified that these were leading properties that enable the mobile with uniquely interesting applications. Since we are traditionally dependent on the mobile for making phonecalls they are always with us in our pockets, this has made them an attractive platform for a variety of sensors. As mentioned, a convenient place to carry a camera, but more generally the phone is subjected to the same environment as the owner, making their context extremely relevant. With the ability to sense geographical location (using the GPS), we never need to get lost again, and can navigate with ease in completely unfamiliar environments. An incredible useful innovation which became commoditized over just the succession of a couple of years. These are the kind of developments that signal the advent of further significant improvements. A near-future candidate is augmented reality (figure 1 for inspiration), which will be enabled by a combination of the motion sensors, vision processing (by the camera), and networked access of knowledge about the geographical vicinity.

Inherent mobility raises some concerns that are not present with desktops. *Wireless network interfaces* are their primary means of communication, and the same assumptions about connectivity cannot be made compared to a desktop with a fixed connection. The landscape of protocols and connection points are diverse. This means that the application developer must take into account variable rates of bandwidth and latency, and in many cases use timeframes with increases connectivity opportunistically such as caching data or performing a synchronisation with a server. Mobile roaming induces handovers between networks, and this is concern both for client and server side applications in developing seamless applications - the common challenge is to develop a functional piece of software that is generally dependent on connectivity but also resistant to potential downtime.

Another direct consequence of mobility is a limited resource of *power*. Power is for one thing an important reason why mobiles are not as fast as desktops because mobiles don't have available the necessary voltage and current levels to drive a high processing frequency. In terms of applications development power consumption places constrains, and introduces a new objective, the constraint is for the application not to kill the battery, and the new opportunities are to find ways to limit consumption as much as possible. There are numerous ways to this effect, one example is that the Android SDK offers applications to issue intents to the OS. An intent is a signal that the application wants a particular resource when it becomes available, for example a GPS fix. In this way power hungry operations can be mitigated across several applications at once.

Another aspect of mobility is a consequence of size, which in turn implicates the methods of *interaction*. The mobile touch-screen represents both the primary means of output (visual) and input (touch-based). The screen are similar to desktops except for their size, and they vary widely between devices as implied by the number of manufacturers. The most significant difference compared to desktops is the touch-based input that effectively replaces both the keyboard and mouse. Both these features of interaction places a demand to completely rethink many applications. Application features that are for instance dependent on the mouse right-click or hover need to be rewritten.

summary

The mobile computing scene is very young and volatile, just consider the rapid turnaround of players (Android, iPhone, Windows and Blackberry) and x-players (MeeGo, Symbian, webOS), which is showing great evolutionary innovation, and that we are no longer dependent on a single provider (such as Windows on the PC). Services have moved into the cloud, and formats have become more universal and defined. Many more players have the ability to be involved, applications on the mobile phones are for instance more recreational and experimental than desktops. In terms of forward thinking development, turnover of mobile systems is so high (consider the iPhone 4S vs original iPhone), development may not need to worry too much about legacy problems. As we have seen mobiles differ in many respects from desktops, but fundamentally they are actually the same, and in many cases lend themselves the same type of applications. In terms of capabilities, I would argue that the mobiles is actually *more* enabled than the desktop. While we can identify many applications that have unique interest on a mobile, it is more difficult to turn the argument the other way. The reason desktops are ahead in many areas are mostly related to challenges that are the mobile is catching up on. The gaps between the limitations of what a mobile can do, and a desktop, seem to be closing rapidly - perhaps soon we can dispense of the desktop all-together?

References

[1] Satyanarayanan M. Mobile computing: the next decade. *ACM Workshop on Mobile Cloud Computing & Services*, 2010.