An Alternative to Thick Clients for Personal Wireless Devices

Dragos A. Manolescu
dragos.manolescu@acm.org

Draft of February 6, 2002

1 Introduction

Thad Starner's article from the January 2002 issue of Computer focuses on improving the functionality of software for personal wireless devices [7]. Starner proposes a shift from thin to thick clients as an easy route to achieving this goal. He argues that if the technology trends continue to evolve as they have done throughout the last decade, wireless devices will have comparable amounts of disk storage, RAM, and CPU power with primary general-purpose computing devices.

While technology will undoubtedly continue to advance, Starner's analysis doesn't take into account important factors like software development costs, application deployment, security, integration with existing infrastructure, and licensing costs. In this article I revisit the topic and examine the characteristics of thin and thick clients in the context of personal wireless devices. I argue that the thick client approach won't cause thin clients to go extinct. Neither approach is perfect, and I'll point out several characteristics of thin clients that make them more attractive than thick clients. As an example, I discuss how Mobile Classic Blend, a Java technology for personal wireless devices exploits the characteristics of the thin client approach while dealing with the constraints imposed by current technology.

2 Background

Thin and thick clients are different sides of the same coin. The coin is client-server, and deals with the partitioning of functionality between the client and the server. The server side provides services and "serves" requests from clients. Depending on the amount of processing that it performs, clients are either thin of thick.

The thin client approach dates back from the days of mainframes. In those days hardware was locked behind doors in air-conditioned rooms. People didn't have direct access to it. Instead, they used "dumb" terminals. All processing took place on the mainframe, and the terminals simply handled paper (and later on CRT) output and keyboard input. Because of the very little amount of processing carried out by these terminals, they were called *thin clients*.

The dawn of the PC era opened the doors to new possibilities. The CPUs, memory, and storage capabilities of personal computers enabled them to run their own programs, independently from the server. Gradually, as PCs processing capabilities increased, software moved from the server to the client. Currently servers host only a few enterprise-wide applications (e.g., databases). The bulk of the software runs on *thick clients* and interacts with dedicated servers only for special services.

3 Thin Clients for Personal Wireless Devices

With the advent of affordable and powerful personal computing, some people believed that thin clients were gone. They were wrong. The Web has resurrected the thin client approach. Server-side programming is back in

style again. With the Web making inroads in the wireless market, our personal wireless devices are going to act like 21st century 3270s as long as the Web remains sever-centric.

However, after more than a decade of experience with thick clients we are now in a better position to evaluate the two approaches. While the thick client approach has opened new doors, we have also discovered problems for which the thin client approach remains a better fit. Therefore, we should not dismiss the thin client approach and wait for technology to catch up. Instead, we should rethink it and capitalize on its benefits while incorporating the lessons we've learned since it went out of style.

Let's consider the following characteristics:

Development cost Building software for PDAs is quite different compared to building software for desktop computers. Things that are different include the programming language [8], the programming style [4], and the development tools [3, 1]. This translates into developer retraining and retooling—an expensive proposition. Through leaving the application on the server, the thin client approach minimizes client-side development.

Integration with existing infrastructure Corporations have invested large amounts of money in wireline technology for their Web-enabled applications. Acquiring new infrastructure to add support for wireless devices represents an expensive endeavor. Businesses would rather amortize their current technology investments and gradually expand their market to wireless customers. Through reusing existing server-based applications, thin clients can integrate with the current infrastructure and allow for a smooth expansion at a fraction of the cost.

Instantaneous deployment Software changes in time. Developers add new functionality or fix security holes. Updating software over the Internet generally requires user involvement and poses security risks. Through using server-based applications, the thin client approach allows developers to deploy new applications as well as update existing applications instantaneously.

Improved security Starner suggests that in the future users may replace their general-purpose computers with mobile systems. In fact, this depends on the type of data you're dealing with. Corporations won't allow their employees to carry sensitive enterprise data on mobile devices. Instead, they would prefer them connecting to enterprise servers, where the corporation has direct control over regulating and monitoring the access. Additionally, should the device fall into the wrong hands, the liability is higher if it contains full-fledged applications or confidential records. Even with encryption, the chances of it being broken while in the possession of a cracker are much higher [6]. Through holding mainly remote GUIs for server applications instead of enterprise data and applications, thin clients have lower liability from a security standpoint.

Reduced cost of ownership Typically software licenses charge a per-CPU fee. Thick clients require a license for each client. For large installations this may yield an expensive bottom line. Through having many users share server-side applications, the thin client approach reduces the cost of ownership.

The above characteristics suggest that the thin client approach has several advantages over the thick client approach. Generally the challenge lies in striking the right balance between the type of processing that takes place on the client, and the type of processing that takes place on the server. In the context of personal wireless devices additional challenges stem from the peculiarities of wireless connectivity and the limitations of current technology. Let's look at how a thin client for personal wireless devices addresses these issues while exploiting the above characteristics.

4 Mobile Classic Blend—A Thin Client for Personal Wireless Applications

A successful thin client for personal wireless applications must deal with an array of constraints. A first set of constraints pertains to pocket-sized devices in general: limited processing power and memory space [4]; reduced screen real estate and limited input capabilities [5]. Another set of constraints pertains to wireless devices: limited bandwidth; disconnected operation; expensive airtime. While the constraints specific to small devices have been dealt with in the past, Web-enabled mobile phones are bringing the second set of issues under the spotlight.

The above constraints make building Internet applications for mobile phones a challenging task. Several technologies attempt to address these challanges. [Briefly discuss WAP and Web clipping.] However, these attempts leave a wide gap between what these technologies provide and what mobile users really want.

Mobile Classic Blend is a Java technology that closes this gap by leveraging the processing capabilities of mobile clients and a bidirectional data transfer protocol. The technology lets you develop and run server-based Internet applications on personal wireless devices. By exploiting the thin client approach, developers build wireless applications with as little regard as possible for the idiosyncrasies and limitations of the target device.

The Mobile Classic Blend architecture consists of a presentation server and a thin client. The presentation server can run standalone or within an existing application server such as BEA's WebLogic or IBM's WebSphere. It integrates with existing server-side technologies such as servlets, JSPs, and RMI. The thin client consists of GUI specifications which describe how applications interface with the users. Additional server and client components implement the bidirectional data transfer protocol and other services like object serialization, transparent for the application programmer. For maximum portability, the architecture leverages the Java platform on the client (J2ME) as well as on the server (J2SE/J2EE). The current version Mobile Classic Blend uses IBM's J9 virtual machine [2] on the client, which adds around 100K to the client side. Figure 1 shows the server and client components, as well as their position next to wireline browser-based and thick client applications.

Next let's examine how Mobile Classic Blend solves the problems specific to personal wireless devices while providing an attractive platform for building and deploying mobile Internet applications.

5 Technology and Business Issues

What makes a thin client approach like the one described in the previous section attractive? The key factor stems from the thin client approach being feasible now, with current technology. In contrast, Starner doesn't provide an immediate solution; moving to thick clients requires waiting for the technology to become available. Other factors include gradual expansion to new markets, real-time notifications, and reduced development effort. Here's how these factors line up.

Compatibility with current wireless technology Due to its server-centric model, thin-clients typically involve more client-server traffic than thick clients. This traffic is subject to the bandwidth limitations specific to wireless devices. Current wireless networks (i.e., 2G) have bandwidth in the range of 9.6 through 19.2 Kbps. Mobile Classic Blend addresses this limit in two ways. First, it uses a communication protocol optimized for low-bandwidth connections. The protocol provides bidirectional, live data exchange between the client and the server. Second, the client side uses the native widgets of the wireless device. This preserves the look-and-feel of the deployment platform while eliminating full-page refreshes. The combination represents a radical departure from the unidirectional, heavyweight solutions typical of HTTP/HTML-based applications.

Real time, wireless servers The Mobile Classic Blend doesn't aim at replacing wireline access. Instead, it augments it when end users need wireless, real time connectivity. The integration with existing infrastructure lets users quickly transform existing J2EE servers into real time wireless servers.

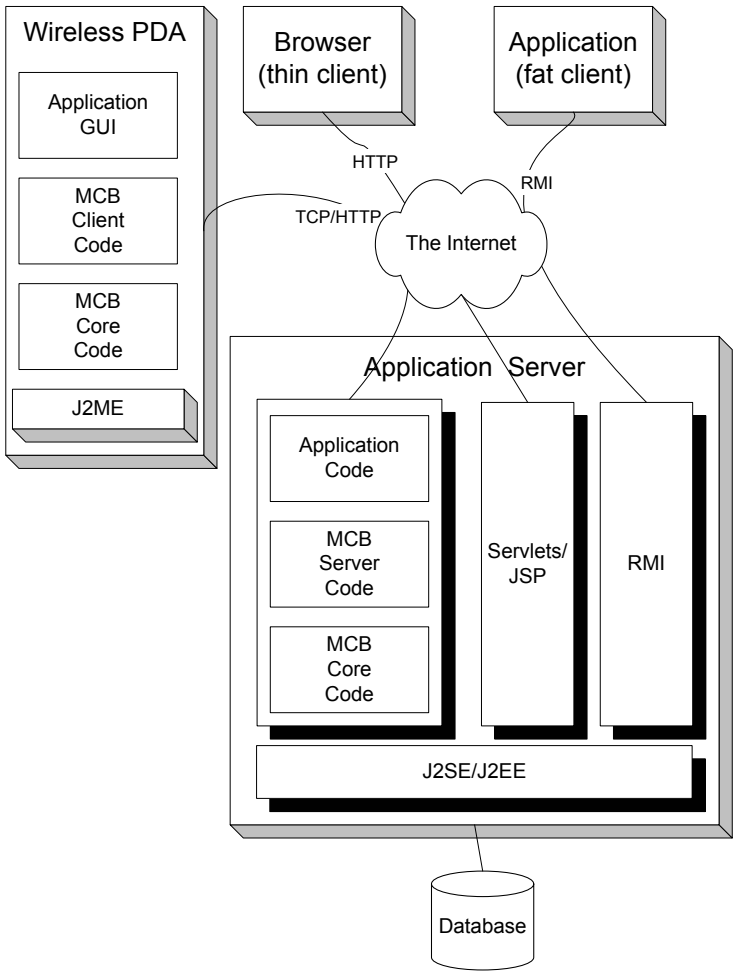

Figure 1: The Mobile Classic Blend architecture consists of a presentation server (depicted here within an Application Server) and a thin client.

Application wake-up and server push People with personal wireless devices need them because they can benefit from being mobile and connected. Sometimes they need to know promptly whenever time-sensitive information changes, and act upon it. Mobile Classic Blend leverages carrier-supported messaging to notify devices in real time, even when they are disconnected from the Internet. The wireless device wakes up and requests a connection to the server in response to the notification. Once the connection is established, the server can push data to the device. In effect, this mechanism gives the illusion of always-on applications and allows for location-based services. Additionally, since messaging costs less than regular airtime, using the messaging service significantly lowers the operational costs.

No device programming Mobile Classic Blend reduces client side programming to composing widgets. The only programming takes place on the server, where developers write Java code with the tools and within the development environments they already know. In effect, this programming model lets you reuse existing resources, knowledge, tools, and environments.

6 Other Challenges

Several characteristics of personal wireless devices make the thin client approach a hard problem. First, unlike wireline computers, the connectivity is intermittent. Second, the synchronization with users data must take into account changes made on both devices. Let's discuss these challenges in more detail.

Wireless network access means intermittent connectivity. Therefore, applications must provide support for disconnected operation: they must tolerate the temporary loss of connectivity. This must remain transparent for the user, who should still be able to use the application. This involves caching some information on the client (i.e., wireless device). For example, consider a mail agent. Sending and receiving email depends on it being connected to the Internet. But people should still be able to use it when the connection is lost. For example, you should be able to compose messages and queue them locally until the connection is reestablished. But is this a thin client or a thick client?

Synchronization with a user's main computer is another important thing. Once people have more than one computer, they can easily loose track of which one holds the latest versions of their files. They may end up making different updates to the same file. Synchronization should perform a smart merge.

7 Summary and Conclusion

An increasing number of personal devices provide wireless network access. The ability to connect mobile users to the Internet has opened the doors to new possibilities. It has also brought under the spotlight several challenges for people building software for personal wireless devices. Dealing with the constraints typical of portable devices and the constraints typical of current wireless technology makes building software for these platforms a hard problem.

Personal devices trade form factor and convenience for power. Current wireless devices solve the above problems by adopting a thin client approach, which relies on a server that performs most processing. But as technology continues to advance, the processing capabilities of personal devices will improve to the point that they will no longer need server-side processing. They will become thick clients. Would this cause the thin client approach to go extinct?

In this paper I've discussed several characteristics that make the thin client approach a viable alternative to thick clients. Traditionally thin clients act as simple interfaces that merely take the user input to the server, and the server output to the user. But in the context of wireless devices the thin client can capitalize on its processing capabilities and find a balance that works in its advantage. As an example, I have presented Mobile Classic

Blend, a Java technology that uses the thin client approach to provide responsive, Internet applications on current generation wireless phones.

Thin-clients won't go away. Smart thin-clients that run responsive Internet applications are available today and run on the current generation of wireless devices. We should not dismiss this approach and use it whenever we need its benefits.

References

[1] Borland Software Corporation. jBuilder MobileSet. On the Web at http://www.borland.com/jbuilder/mobileset/.

[2] IBM. Ibm j9 virtual machine. On the Web at `http://www.embedded.oti.com/learn/vaesvm.html`.

[3] IBM. VisualAge Micro Edition. On the Web at http://www.embedded.oti.com/.

[4] James Noble and Charles Weir. *Small Memory Software: Patterns for Systems with Limited Memory*. Software Patterns Series. Addison-Wesley, 2000.

[5] James Noble and Charles Weir. A window in your pocket: Some small patterns for user interfaces. In *Proc. European Pattern Languages of Programs*. The Hillside Group, Inc., 2001.

[6] Bruce Schneier. *Secrets and Lies: Digital Security in a Networked World*. John Wiley & Sons, 2000.

[7] Thad Starner. Thick clients for personal wireless devices. *IEEE Computer*, 35(1):133–135, January 2002.

[8] Java 2 platform, micro edition. On the Web at http://java.sun.com/j2me/.