Mobile Computing Software Development

Esther V. V. Reed and Matt Mutka

Department of Computer Science and Engineering
3115 Engineering Building
Michigan State University
East Lansing, MI 48824-1226

Abstract

Mobile computing has gained momentum and grown rapidly in recent years. Portable computing devices such as notebooks, palmtops, and handheld devices are readily available and are becoming quite common. Some devices have an embedded, proprietary operating system (OS), while other devices have an embedded, commercially available OS allowing different models to have the same base OS. If development tools exist, developers should find porting an application to a different device using the same OS far easier than to one using a completely different OS.

This paper examines issues for the development of an operating systems’ course laboratory assignment using a commercial OS. The embedded system platform that is targeted for this laboratory assignment is a H/PC device using the Microsoft Windows CE operating system. Commercial developer’s tools for these platforms and environments from Microsoft are used. The primary software result of this assignment is to develop parts of an application that are used to create a time client and an FTP client using the Windows CE Toolkit for Visual C++ 6.0.

This paper discusses the software development products that are explored for developing the laboratory assignment and the target platforms and environments. We discuss the paths and pitfalls for developing software in this environment and how this will effect a course laboratory assignment.

Keywords: embedded system, H/PC, Windows® CE, Visual C++®, HP Jornada.

I. Introduction

Mobile computing has gained momentum and grown rapidly in recent years. Portable computing devices such as notebooks, palmtops, and handheld devices are readily available and are becoming quite common. Some devices have an embedded, proprietary operating system (OS) with each model having its own system. However, other devices have an embedded, commercially available OS, which allows many different models to have the same base OS. Microsoft® Windows® CE is such an OS. Using the same base OS across different platforms provides advantages to both users and developers. Users get a similar look and feel, which makes using different models of devices easier. Provided that development tools exist, developers should find porting an application to a different device using the same OS far easier than to one using a completely different OS.
This paper presents an exploration of software development tools for embedded systems using a commercial OS, which may be appropriate for laboratory development. The embedded system platform and environment that was investigated is a Handheld Personal Computer (H/PC) device using the Windows CE operating system. The primary device and OS, for which software was developed, was the Hewlett Packard (HP) Jornada 820 Handheld PC Professional (H/PC Pro) running Windows CE 3.0. The secondary device and OS was the HP 620LX Palmtop PC running Windows CE 2.0. Commercial developer’s tools for these platforms and environments from Microsoft® were investigated. The C++ and Java programming languages were to be included in this project. Consequently, the following packages were planned for investigation: Windows® CE Toolkit for Visual C++® 5.0, Windows® CE Toolkit for Visual J++™ 1.1, Windows® CE Toolkit for Visual C++® 6.0, and Windows® CE Toolkit for Visual J++™ 6.0. It was found that the primary development device was only supported by the latter two packages. Additionally, Windows CE Toolkit for Visual J++ 6.0 was never released and has since been removed from Microsoft’s list of products. Thus, the primary software tool that was investigated was the Windows CE Toolkit for Visual C++ 6.0. The results of this exploration were used to create a time client and an FTP client for both the primary and secondary devices.

We first review the growth of mobile computing and how that has affected the evolution of H/PCs and Windows CE. Next, a discussion is presented of the software development products that were explored, plus our target platforms and environments. We will then focus on the paths and pitfalls of development that were discovered during the course of this exploration. Lastly, we will go over the software that we developed, our recommended changes for the Windows CE Software Development Kits (SDKs), and future work for this project.

II. What are Windows® CE and H/PCs?

The demands of mobile computing have established the need for small, lightweight devices. If a device is too heavy or bulky, a user will not be inclined to take it anywhere. These two requirements dictate directly and indirectly the resources available in that device, resulting in the fact that mobile computers or devices are resource poor relative to static units. Much work has been done researching the needs of mobile devices and what changes are needed in a traditional OS to accommodate them.

II. A. Growth and Trends of Mobile Computing

In 1990, the first notebook computers were introduced. These machines were primarily used as secondary tools for traveling or for taking work home. Now many people use their mobile computer as their primary and only machine in the office, and many work places encourage the use of mobile computers. In 1998, 20 percent of all personal computers installed were mobile or portable computers. The sales of portable computers increased by 24.6 percent from 1996 to 1997, while desktop computer sales increased by 14.6 percent over the same period. The annual growth rate for portables through 2001 has been forecast at 16.2 percent by International Data Corporation. This steady growth has sometimes been attributed in part to the growth of the Internet and E-Business. Both consumers and businesses have an interest in anytime anywhere computing and Internet access. The growth in mobile computing and the change in consumer computing needs have led companies such as Microsoft to reevaluate their development strategy.
The old mantra for many computer-related companies was “a computer in every home and on every desktop”. The focus is changing to include wireless computing and Internet-oriented applications.

Researchers soon recognized that traditional operating system designs assumed a static system, and were inadequate to handle the unique events encountered in the mobile computing environment. The consumer desire for easy mobility causes a compromise between lighter, smaller devices and available system resources. Mobile devices have lower power, slower processors, smaller storage capacity and a smaller area for user input than their static cousins. Although a long battery life is desirable for mobile use, much of the total weight is the battery. Thus, a manufacturer faces a trade-off between either a larger battery that makes the weight inconvenient, or a lighter device that needs to be recharged more frequently. Since hard drives are relatively heavy and need a large amount of power, a device may opt for RAM and ROM chips that have less storage capacity. In addition, the OS for a mobile device also faces different network and security situations than those faced by a traditional OS.

Much work has been done evaluating and investigating solutions to various mobile OS topics, such as security, network, wireless operation, adaptive OS, and adaptive applications. The uniqueness of this research prompted the ACM and IEEE to create a separate Mobile Computing and Networking Conference in 1994. This annual conference continues to grow and attract new research.

II. B. Evolution of Windows CE

Commercial vendors and developers also looked at changes needed to create an OS that could handle the constraints of the resource poor embedded devices. Microsoft’s answer was Windows CE.

Windows CE 1.0 was first unveiled at Comdex 1996. The first generation Windows CE devices, and all later generation devices to date, are PC companions meaning they are designed to exchange and synchronize data with a desktop computer. Windows CE 1.0 includes a suite of applications, and both the OS and applications are embedded in the device’s ROM. While documents and other programs may be downloaded and used on the device, in the event of power-loss, only the original OS and applications are in permanent storage. Although it has the same look and feel, Windows CE is not a ported version of Window 95. Windows CE was designed from the ground up to work with resource poor mobile devices.

Windows CE is a small, scalable, 32-bit Windows-compatible OS intended to work on a wide range of embedded devices with variable user input methods. It supports the Win32 programming model, and exports a subset of the Win32 API. Windows CE is targeted at small systems with limited memory. To enable accommodating variation in hardware and software features, Windows CE is modular. Thus, a vendor may only include the needed parts of the OS. This results in a separate SDK for each platform.

Windows CE 1.0 supports the Philips, the NEC MIPS, and the Hitachi SH3 processors, and it supports a monochrome 480 X 240 screen size. Windows CE 1.0 provides a nice assortment of
embedded applications such as Information Manager, which includes Calendar, and Tasks and Contacts, which can automatically synchronize with Microsoft Schedule+ 7.0a. In addition, Pocket Word, Pocket Excel, Pocket Internet Explorer and an Inbox for email are provided. Although the devices rely on a serial cable to synchronize with their host desktop computer, Windows CE 1.0 includes remote dial-up supporting TCP/IP and PPP protocols. These devices have 4 MB to 8 MB of on-device ROM, and from 2 MB to 4 MB RAM 19.

Initially, Windows CE 1.0 devices were met with excitement and high expectation. However, by mid-1997, a common reaction seemed to be that these devices were too slow, the screens were too small and there was not sufficient memory. At a Microsoft Windows CE conference in May 1997, Microsoft announced its plans for Windows CE 2.0, which was formally announced to the public in September 1997 at the Embedded Systems Conference in San Jose 18.

Some of the improvements in Windows CE 2.0 are support for 256 color, support of 640 X 240 screen size, support for direct printing, cascading menus, support for microphones, and improved Pocket applications. In addition, Pocket PowerPoint and Pocket Outlook have been added. Pocket Internet Explorer has been integrated with the Explorer shell. Synchronization with the desktop computer has been improved with Windows CE Services, which handles document transferal and conversion between the different operating systems and different versions of an application, e.g. Pocket Word to Microsoft Word. Windows CE 2.0 also supports Ethernet LAN, SLIP protocol for wireless Cellular Digital Packet Data (CDPD), and built-in modems. These devices have from 8 MB to 16 MB RAM 22, 23.

In 1998, Microsoft announced Windows CE, Handheld PC Edition Version 3.0 having the Core OS System Version 2.11. Windows CE 2.11 or the H/PC Pro Edition includes Pocket Access, which includes ActiveX Data Objects (ADO) allowing synchronization with ADO-based databases on desktop or network servers. Improvements were made in the existing Pocket applications, plus color displays of 640 X 480, or 800 X 600 are supported. Processor support has been increased. In addition to the NEC MIPS and the Hitachi SH3 processors, Windows CE 2.11 also supports the Hitachi SH4 and the StrongArm 1100 processors. Communication support has been increased to included CompactFlash Type II, PCMCIA, and infrared (IrDA standard) protocols. The H/PC Pro Edition includes support for wired and wireless Ethernet adapters, parallel ports, fast infrared (Fast IR) and Universal Serial Bus (USB). These devices have 16 MB of on-device ROM, and from 16 MB to 32 MB RAM 24.

Windows CE 2.11 supports a subset of Winsock 1.1, but still has good network support. Stream and datagram connections are supported, as are WNetAddConnection3. The WinCE API is taken from the Win32 API, but it is not a subset. Graphical support is quite reduced relative to the Win32 API, and the database API is completely different in Windows CE. Files are specified by their complete path, and the idea of a current directory does not exist. Unicode is used in Windows CE, although data conversion functions are provided since some functions, such as the Winsock functions, need to work with char. Windows CE 2.11 is a multi-tasking / multi-threaded OS, that is limited to 32 separate processes. Processes use much less state information, for example, than what is used in Windows 95 4, 20, 21.
Two new releases of Windows CE are forthcoming from Microsoft – Windows CE 2.12 and Windows CE 3.0. An amusing but accurate observation is that “the only Microsoft product that seems to have been updated faster than Microsoft Internet Explorer is Windows CE” 25.

Windows CE 2.12 is an update to Windows CE 2.11 and should be released soon. Some of the expected changes include added security features to prevent the loading of untrusted DLLs and OEM modules. The shell is being improved, and a second, more powerful web browser is being added. The notification subsystem is being redesigned, although the former functions will be maintained for compatibility 25.

Windows CE 3.0 is codenamed Cedar and is a major version change. The scheduler is being redesigned to allow much improved real-time support. Nested interrupts will be supported and the number of thread priority levels is being increased. Four new thread functions will be added to support these changes. This version includes the features of the previous Windows CE versions plus the improvements of Windows CE 2.12. Windows CE 3.0 is expected to add TAPI and support incoming calls, in addition to the Simple Network Management Protocol (SNMP). An HTTP server component will be added, that will include both basic and NTLM client authentication support. In-process and out-of-process COM objects will be supported with all threading models. Windows CE 3.0 will implement a version of Microsoft Message Queue Service (MSMQ) to support distributed applications. Changes to the file system include increasing the object store from 16 MB to 256 MB. The database API is adding features to create, open and seek databases. The accessibility functions are being added back to the graphics API. A pre-release of Windows CE 3.0 is expected soon, but will not likely have all of the intended changes. Windows CE 3.0 will continue to be a modular OS. If the memory requirements of the new features are too high, a vendor does not need to include them on their version of Windows CE 3.0 25.

II. C. Evolution of H/PCs

Using the Microsoft Windows CE Platform Builder, Windows CE can be compiled for a wide range of embedded system devices. The modular design of the OS allows a developer to select what to include. After generating the custom Windows CE OS, the Platform Builder can create an SDK to support software development. Despite the ability for Windows CE to be compiled for existing devices, the development of Windows CE was paralleled by the development of new embedded system devices, designed around Windows CE specifications and abilities 18.

The new devices, Handheld Personal Computers (H/PCs), also called PC Companions, were aimed at the niche between a Personal Digital Assistant (PDA) and a notebook computer. The H/PCs were designed to be lighter, smaller, and less costly than a notebook computer, yet still have more powerful applications and more sophisticated connectivity than a PDA. The actual size of the device is approximately 7 inches long, 4 inches deep and 1 inch thick when closed.

NEC, Philips, LG Electronics, Casio and Hewlett Packard (HP) were the original licensed vendors producing new Windows CE devices. When Windows CE 1.0 was unveiled in 1996, only the NEC MobilePro 200 and the Casio A-10 were shipping 18. The Casio and NEC models followed the strict Microsoft specifications, resulting in little difference between them 26, 27.
Philips and HP released Windows CE 1.0 devices in 1997, and both varied from the Microsoft specifications to include additional features\(^\text{18, 28, 29}\). Casio responded by introducing the A-11\(^\text{27}\), and NEC also improved its product line releasing the MobilePro 400 and MobilePro 450\(^\text{26}\). Table 1 shows a comparison chart of these devices.

<table>
<thead>
<tr>
<th>Vendor and Model</th>
<th>Display</th>
<th>Memory</th>
<th>Input</th>
<th>Extra Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEC Mobile Pro 200</td>
<td>grayscale 480 X 240</td>
<td>2 MB ROM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>Casio A-10</td>
<td>grayscale 480 X 240</td>
<td>2 MB ROM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>Philips Velo 1</td>
<td>grayscale 480 X 240</td>
<td>4 MB RAM, and 5 or more MB ROM</td>
<td>keyboard, touchscreen</td>
<td>voice-recording</td>
</tr>
<tr>
<td>HP 320LX</td>
<td>grayscale 640 X 240</td>
<td>4 MB ROM</td>
<td>keyboard, touchscreen</td>
<td>direct printing from device</td>
</tr>
<tr>
<td>Casio A-11</td>
<td>grayscale 480 X 240</td>
<td>4 MB ROM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>NEC Mobile Pro 400</td>
<td>grayscale 480 X 240</td>
<td>4 MB ROM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>NEC Mobile Pro 450</td>
<td>grayscale 480 X 240</td>
<td>4 MB ROM</td>
<td>keyboard, touchscreen</td>
<td>backlit display</td>
</tr>
</tbody>
</table>

Table 1: Comparison of Some Windows CE 1.0 H/PCs

An interesting side note is that, prior to Windows CE, HP released the HP 200LX that used MS DOS 5.0 as its OS. This device had a CGA compatible 640 X 240 display and included Pocket Quicken and Lotus 1-2-3 release 2.4\(^\text{30}\).

When Windows CE 2.0 was released in 1997, NEC, Casio, Philips, HP and LG Electronics announced new hardware. Other vendors such as Sharp also joined the Windows CE group and produced new Windows CE devices. Casio released the A-20, and introduced the Casio Cassiopeia E-10, a Palm-Size Personal Companion (PPC)\(^\text{31}\). Philips produced the Velo 500\(^\text{33, 32}\), and NEC unveiled two Windows CE 2.0 devices, the MobilePro 700 and the MobilePro 750c\(^\text{33, 34, 35}\). HP first released the HP 340LX and the HP 360LX\(^\text{29}\), and later produced the HP 620LX Palmtop PC and HP 660LX Palmtop PC\(^\text{53}\). Sharp introduced the Mobilon HC-4100 and Mobilon HC-4500, followed by the Mobilon HC-4600\(^\text{33, 36}\). Table 2 presents a comparison chart of these devices.
### Table 2: Comparison of Some Windows CE 2.0 H/PCs

<table>
<thead>
<tr>
<th>Vendor and Model</th>
<th>Display</th>
<th>Memory</th>
<th>Input</th>
<th>Extra Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>H/PC Casio A-20</td>
<td>4-bit grayscale, 640 X 240</td>
<td>8 MB ROM, and 8 MB RAM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>Philips Velo 500</td>
<td>16-bit monochrome, 640 X 240</td>
<td>16 MB RAM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>NEC Mobile Pro 700</td>
<td>4-bit grayscale, 640 X 240</td>
<td>16 MB ROM</td>
<td>keyboard, touchscreen</td>
<td>backlit display</td>
</tr>
<tr>
<td>NEC Mobile Pro 750c</td>
<td>256 color, 640 X 240</td>
<td>16 MB ROM</td>
<td>keyboard, touchscreen</td>
<td>backlit display</td>
</tr>
<tr>
<td>HP 340LX</td>
<td>4-bit grayscale, 640 X 200</td>
<td>4 MB RAM, and 10 MB ROM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>HP 360LX</td>
<td>16-bit grayscale, 640 X 240</td>
<td>8 MB RAM, and 10 MB ROM</td>
<td>keyboard, touchscreen</td>
<td></td>
</tr>
<tr>
<td>HP 620LX</td>
<td>256 color, 640 X 240</td>
<td>10 MB ROM, and up to 16 MB RAM</td>
<td>keyboard, touchscreen</td>
<td>backlit display</td>
</tr>
</tbody>
</table>
| HP 660LX | 256 color, 640 X 240 | 10 MB ROM, and up to 32 MB RAM | keyboard, touchscreen | • backlit display  
• external 56Kbps modem |
| Sharp Mobilon HC-4100 | 16-bit grayscale, 640 X 240 | 12 MB RAM | keyboard, touchscreen | |
| Mobilon HC-4500 | 256 color, 640 X 240 | 16 MB RAM | keyboard, touchscreen | |
| Mobilon HC-4600 | 256 color, 640 X 240 | 32 MB RAM | keyboard, touchscreen | |

Prior to the release of Windows CE 2.0 devices, consumer interest in Windows CE devices seemed to be faltering. This trend changed with the release of H/PC devices with larger, color screens and more memory, coupled with a richer OS having new and improved applications. Consumer interest in Windows CE devices increased, and critical opinion of Windows CE devices seemed to improve greatly [18].

Windows CE 2.11, released in fall of 1998, was much anticipated. The new OS improved communications, added applications, and – possibly most importantly – gave vendors much greater design flexibility. The size of Windows CE 2.11 devices can range from small PPCs (Casio Cassiopeia E-105) to medium H/PC Pro’s (HP Jornada 680) to large H/PC Pro’s (Sharp Mobilon PV-6000). Screen displays can be monochrome or color, and vary from 480 X 240, to 640 X 240, to 640 X 480, to 800 X 600 [38]. NEC, Casio, HP, LG Electronics and Sharp were joined by Compaq, Everex, Fujitsu, Hitachi, NTS Computer Systems, Samsung and Vadem in producing Windows CE 2.11 devices [38]. Philips has not yet produced new hardware using the new OS [32]. Casio has not produced H/PC Pro hardware, but has gone with PPC devices continuing with the Cassiopeia line [39]. In late 1998, HP unveiled the HP Jornada 820. Geared
towards traveling business people, the Jornada 820 feels like a mini-notebook computer. HP later added to its Windows CE 2.11 devices with the introduction of the Jornada 420, a PPC, and the Jornada 680, a mid-size H/PC Pro. In early 1999, NEC released the MobilePro 770, and later released the MobilePro 800, which is a large size H/PC Pro device. Sharp produced the Mobilon PV-5000 followed by the Mobilon PV-6000, which are both are large size H/PC Pro devices. Thus far, all H/PC and H/PC Pro devices have been a clam shell design. The Mobilon PV-6000 introduced a new easel design that allows it to have one of the largest screens at 9.4", but also makes it one of the heaviest devices at 3.2 pounds. Table 3 presents a comparison chart of these devices.

<table>
<thead>
<tr>
<th>Vendor and Model</th>
<th>Display</th>
<th>Memory</th>
<th>Input</th>
<th>Extra Features</th>
</tr>
</thead>
</table>
| **HP Jornada 820** | 256 color, 640 X 480 backlit | 16 MB ROM, and up to 32 MB RAM | keyboard, touchpad | • 90% of full size keyboard  
• VGA-Out  
• speaker and microphone  
• internal 56 Kbps modem |
| **HP Jornada 680** | 256 color, 640 X 240 backlit | 16 MB ROM, and up to 32 MB RAM | keyboard, touchscreen | internal 56 Kbps modem |
| **NEC MobilePro 770** | 256 color, 640 X 240 | 32 MB ROM | keyboard, touchscreen | internal 56 Kbps modem |
| **NEC MobilePro 800** | 64K colors, SVGA 800 X 600 | 24 MB ROM, and 32 MB RAM | keyboard, touchscreen | • internal 56 Kbps modem  
• VGA out |
| **Sharp Mobilon PV-5000** | 640 X 480 VGA | 16 MB RAM | keyboard, touchscreen | • internal 33 Kbps modem  
• speaker |
| **Sharp Mobilon PV-6000** | 640 X 480 VGA | 16 MB RAM | keyboard, touchscreen | • internal 33 Kbps modem  
• speaker |

Table 3: Comparison of Some Windows CE 2.11 H/PC Pro’s

If Windows CE 2.0 devices favorably changed the growth of Windows CE, then Windows CE 2.11 Palm-Size Personal Companions (PPCs) and H/PC Pro devices caused it to soar. The advantages of the medium and large size H/PC Pro devices over conventional portable computers are smaller size, lighter weight, instant power-on, longer battery life and better durability due to far fewer moving parts. Although Windows CE PPCs are competing against an older, strong PDA market, a recent PC Magazine review states that PPCs, such as the Cassiopeia, are easier to use and have better applications. Businesses that do much of their work in a mobile setting, such as the healthcare industry, have long looked at the use and benefit of PDAs and electronic devices. In 1997, the Visiting Nursing Association of NY purchased 1400 Fujitsu pen tablets for its home healthcare nurses to input patient demographics and procedures. While the medical profession is beginning to use and rely on these devices, a common opinion seems to be that such devices take too long to learn and proprietary operating systems are costly to build and maintain. The arrival of the Windows CE OS and Windows CE devices presented big advantages to businesses. Training time is reduced since users have a familiar Windows desktop and applications. The H/PC allows a user to make notes, make
appointments, and update records away from their desk, and later easily synchronize this information with their desktop computer via Windows CE Services. Documents created with Microsoft desktop products can also be converted via Windows CE Services and used in Windows CE Pocket applications. These advantages have caused businesses, such as the healthcare industry, to shift to Windows CE devices. In 1998, Physix, Inc. introduced PocketChart, a Windows CE virtual charting application, that allows users to update charts, document treatment plans, and send prescriptions to the pharmacy. Physicians take the H/PC devices on their rounds and upload the new information to the primary computer later. In 1999, the Stanford University Cardiology faculty announced that they are developing Windows CE applications to be used for entry and reporting of patient data and test results, and are incorporating the use of Microsoft Access databases. They are also working on wireless communication of patient monitoring information, such as ECG data, to the host computer and Windows CE devices.

Although Windows CE 2.12 and Windows CE 2.0 have been announced by Microsoft, at the time of this paper no new Windows CE devices for these OS’s have been announced by vendors.

III. Software Development Products Evaluated

Windows CE was chosen as the target environment for this project. Four software development packages were considered for evaluation. These software packages are Microsoft products.

III. A. Windows CE Toolkit for Visual C++ 5.0 and Windows CE Toolkit for Visual J++ 1.1

Both the Windows CE Toolkit for Visual C++ 5.0 and the Windows CE Toolkit for Visual J++ 1.1 were released as part of the July 1998 MSDN distribution. Optionally, one could have purchased these packages from a reseller. A computer running Windows NT 4.0, Service Pack 3 is the required OS and development platform. The development platform used for this project is a 200 MHz Pentium with MMX and 96 MB RAM, running Windows NT 4.0 at SP 4.0. Both Toolkits were designed to be installed as add-ons to Microsoft Visual Studio 97, Professional or Enterprise versions. One could also add these Toolkit packages to the Professional or Enterprise versions of, respectively, Microsoft Visual C++ 5.0 or Microsoft Visual J++ 1.1. Thus, either Visual Studio or the appropriate programming language had to be installed prior to Windows CE Toolkit installation. We used Visual Studio 97 Professional, and added Service Pack 3 for that product prior to Toolkit installation. Windows CE 1.0 and Windows CE 1.01 are supported directly by this Toolkit, plus cross-compilers are provided that support the following processors: NEC MIPS and Hitachi SH3. Windows CE 2.0 support could be added by obtaining and installing a free download, Microsoft Windows CE Platform SDK, Handheld PC Edition, Version 2.0.

After installation of the above software, all three versions of Windows CE were supported from within either Visual C++ 5.0 or Visual J++ 1.1. A developer simply chooses to create a WCE application, selects which platforms to include and clicks ‘OK’ to create the application. After the application workspace is created, the Windows CE version may be easily selected and changed from the workspace menu. Likewise, the target platform or processor may be easily selected and changed from the menu, provided that the platform was included when the WCE
application was created. Both Toolkits also provide an emulator to allow the WCE application to be run on the desktop development computer in place of a real Windows CE device.

These two Toolkits were only investigated briefly at the beginning. Since the primary target platform, the HP Jornada 820, was not supported by these toolkits, work did not continue with these Toolkits.

III. B. Windows CE Toolkit for Visual J++ 6.0

Although the release of this product was briefly advertised on the Microsoft web site, in March of 1999 only a picture of the product could be found on the Windows CE developers web site. In April of 1999, the Windows CE Toolkit for Visual J++ disappeared from the Microsoft products list, and the picture was removed. This may have coincided with various court cases in which Microsoft was then involved. Microsoft Visual J++ 6.0 is still sold by Microsoft 49. However, all current searches on the Microsoft web site for the Windows CE Toolkit for Visual J++ direct the user either to the Windows CE Toolkit for Visual C++ 6.0 or to the Windows CE Toolkit for Visual Basic. These are now the only two Windows CE Toolkits discussed on the Windows CE web site 50, 51. Hence, the Visual J++ toolkit and the Java language were not included in the project.

III. C. Windows CE Toolkit for Visual C++ 6.0

The Windows CE Toolkit for Visual C++ 6.0 was first released as a beta in October 1998. This product was the first Windows CE Toolkit to support the ARM processor and the HP Jornada 820. We used the beta, until the Toolkit was released in March 1999. At that time, the released Toolkit was only available from resellers, but has since been released in the MSDN distribution.

Again, a computer running Windows NT 4.0, Service Pack 3 is the required OS and development platform. The development platform used for this project is, once again, a 200 MHz Pentium with MMX and 96 MB RAM, running Windows NT 4.0 at SP 4.0. This Toolkit was designed as an add-on to Microsoft Visual Studio 6.0, Professional or Enterprise versions. Again, Visual Studio had to be installed prior to the Windows CE Toolkit installation. We used Visual Studio 6.0 Enterprise and added Service Pack 2 for that product prior to installation of this Toolkit. Windows CE 1.0 and Windows CE 1.01 support have been dropped from this Toolkit. The Windows CE Toolkit for Visual C++ 6.0 supports Windows CE 2.0 and 2.01, plus cross-compilers are provided that support the following processors: NEC MIPS, Hitachi SH3, Hitachi SH4, and StrongARM 1100. Although later releases of this Toolkit may include Windows CE 2.11 support, initially support for the H/PC Pro Edition could only be added by obtaining and installing a free download, Handheld PC Professional 3.0 SDK.

After installation of the above software, all three versions of Windows CE were supported from within Visual Studio’s Visual C++ 6.0. As with the earlier Toolkits, a developer simply chooses to create a WCE application, selects which platforms to include and clicks ‘OK’ to create the application. After the application workspace is created, the Windows CE version and the target platform may be easily selected and changed from the workspace menu. The Windows CE Toolkit for Visual C++ 6.0 again provides an emulator to allow the WCE application to be run
on the desktop development computer in place of a real Windows CE device. This Toolkit was the primary software development package that was investigated and used for software development.

IV. Target Hardware Platform and OS Environment for Developed Software

The *Windows CE Toolkit for Visual C++ 6.0* supports five different processors and allows easy cross-compilation of processors and versions of Windows CE. We were able to compile the developed software for all supported Windows CE versions and processors, and tested the software on two actual devices.

IV. A. Primary platform – the HP Jornada 820

The HP Jornada 820 was chosen as the primary platform for software development since it runs the most recent version of Windows CE. The intent was to develop network utilities for Windows CE, and it was felt that Windows CE 2.11 was most likely to support the necessary functions. The HP Jornada 820 used in this project has a 32-bit, 190 MHz StrongArm SA-1100 processor, 16 MB ROM, and 16 MB RAM. The OS is Windows CE 3.0, Core System Version 2.11. User input is done through a keyboard and a touchpad. The display is 640 X 480 X 256 colors, with a built-in VGA-Out port where the external monitor display can be 1024 X 768 X 256 colors. The Jornada 820 has an integrated 56Kbps, v. 90 modem, a USB port, a serial port, a Fast IrDA infrared port plus a built-in speaker and microphone. The rechargeable battery lasts around 10 hours. The Jornada 820 is light, weighing around 2.5 lbs., and it is small with its size being about the same as two CD jewel cases laid side by side.

The device comes preinstalled with all of the applications previously mentioned with Windows CE 2.11. HP also includes some utilities to configure such settings as the screen brightness, and some communications tools, e.g. HP dialup. However, we found that the standard Windows CE communications tools were easy to use. HP also included a CD-ROM with the latest version of Windows CE Services to allow the Jornada 820 to synchronize with a desktop computer.

Since the development platform had a serial mouse and a modem, we found that getting the synchronization to work between the Jornada 820 and the desktop computer was non-trivial. We suspect that we would have had no problems had the system used a PS/2 mouse. We found that the Jornada 820’s synchronization software had to be set to use the serial port COM 1 at a baud rate of 19200. We then set the modem to use COM 3 and the serial mouse to use COM 2. Once the serial port settings were in place, the synchronization using the Windows CE Services worked fine.

A minor problem that bears mentioning is the occasional freeze or hanging of communication between the Jornada 820 and the desktop computer at the initialization of synchronization. When this occurred, simply removing the serial cable from the Jornada 820 momentarily and then reseating it was enough to reset the connection. This could be done while *Windows CE Services* was attempting to communicate with the Jornada 820 without stopping and restarting Windows CE Services. Reseating the serial cable is actually recommended by HP as a possible fix for this problem.
IV. B. Secondary platform – the HP 620LX Palmtop PC

The HP 620LX Palmtop PC was chosen as the secondary platform for software development, since it was available in-house and runs Windows CE 2.0. Thus, the two platforms will test the ease of cross-compilation for two different processors and two different versions of Windows CE.

The HP 620LX used in this project has a 75 MHz Hitachi SH3 RISC processor, 10 MB ROM, and 16 MB RAM. The OS is Windows CE 2.0. User input is done through a keyboard or a Pen & Touch interface. The display is 640 X 240 X 256 colors. The 620LX has a serial port, and an IrDA infrared port plus a built-in speaker and microphone. The 620LX is very light, weighing about one pound, and it is small with its size being slightly larger than a checkbook. The device comes preinstalled with all of the applications previously mentioned with Windows CE 2.0. HP again includes some utilities and tools.

V. Paths and Pitfalls of Development

V. A. Sources of Information and Help

Information on Windows CE development is growing. However, when we started this project it was somewhat limited. The first helpful source of information that we found was the Microsoft Windows CE web site for Developers. This provided a checklist on software necessary to develop for platform X using environment Y, overviews of Windows CE versions, white papers on Windows CE architectural designs, technical papers on Windows CE development issues and suggestions on books for developers. Much of the information here is high-level. For example, The Windows CE Communications Model provides a nice architectural overview of communications in Windows CE, but gives no specifics to help figure out why the ICMP functions do not work.

In addition to web help, Microsoft also provided two mailing lists with Windows CE information, The Windows CE Developer News Flash and the Windows CE News Wire. These two lists have since been combined under the Windows CE News Wire. Subscribing to this mailing list is very worthwhile. While the bulk of the announcements are of little interest at least one announcement in every other issue is extremely helpful.

Windows CE newsgroups are available on the Internet news servers, and on the MSDN news server. While both can be helpful, the MSDN newsgroups for developers filter out the non-developer questions and seem to have a more knowledgeable group of developers reading it. In addition, some Microsoft developers occasionally answer or provide insight on various problems. The downside of the MSDN newsgroups is that one must be a MSDN member to read them, although that may no longer be true.

All of the Windows CE Toolkits provide release notes. While reading through them the first time is rather dull, when one hits the first compilation or link error it is a good idea to look...
through them. Several oddities occur with certain libraries, certain processors, and certain versions of Windows CE. The release notes do help with debugging.

Each SDK also comes with release notes. It is definitely worthwhile to read these. If nothing else, one begins to realize that just because a program does or does not compile for the emulator can have little bearing on whether or not the program will work for a specific platform. One also realizes that differences may exist in the same function used in different versions of Windows CE.

If one has access to the MSDN library, it should be installed on the development platform. The help files that come with Windows CE are tailored to that OS. These help files often provide little or no help on general Win32 functions or procedures.

If one has little or no experience with Win32 programming, the book *Programming Microsoft Visual C++ Fifth Edition*[^55] is a must. We worked through the first six chapters to get a good foundation in Win32 development. This book stresses the use of MFC, which turns out not to be recommended for Windows CE. Still, the book is still very helpful as a Visual C++ programming reference. Another Visual C++ reference is *Microsoft Visual C++ 6.0 Programmer’s Guide*[^56]. While lacking in the great examples found in the Kruglinski book, the Zaratian book is somewhat helpful, too.

High-level information on Windows CE can be found in *Inside Microsoft Windows CE*[^21]. This was helpful with getting familiar with what Windows CE is, but provided no help for applications development. The best book and reference for Windows CE application development is *Programming Microsoft Windows CE*[^20]. This book has great examples, clear explanations, good hints on problems to avoid and helpful suggestions. If you are doing Windows CE development, get this book! We worked through chapters 1-5, and 10 to get a good foundation in Windows CE style programming. This book stresses the use of the Win32 API functions, and avoids the use of MFC. The reasons for this will be discussed later. *Microsoft Windows CE Programmer’s Guide*[^57] is also a nice programming reference. Early in the project, this book provided the only easily referable and complete list of Windows CE API functions.

*Windows Sockets Network Programming*[^58] provides very solid information on Winsock concepts and programming. We also found a very helpful web site on Winsock and Windows programming, *Vijay Mukhi’s Technology Cornucopia*[^59]. This site has simple and clear explanations and examples, and is a worthwhile visit for anyone beginning Winsock programming. Lastly, We found some very helpful public domain or GNU source code, which assisted me greatly with the FTP client. These were *dftp* written by D. C. Asonye, *NcFTP* written by M. Gleason, and *WS_FTP* written by J. A. Junod.

V. B. Similarities To and Differences From Win32 Development

The look and feel of Windows CE software development is very much like Win32 development. The windows and dialog box procedures function the same way, many of the functions have the same name, and the same development interface, Visual Studio, can be used by either

[^55]: Programming Microsoft Visual C++ Fifth Edition
[^56]: Microsoft Visual C++ 6.0 Programmer’s Guide
[^57]: Microsoft Windows CE Programmer’s Guide
[^58]: Windows Sockets Network Programming
[^59]: Vijay Mukhi’s Technology Cornucopia
development environment. But in reality, Windows CE development is more of a second cousin to Win32, rather than a sibling. They may look and feel the same, but the details of “how do I get X to happen” can range from slightly to very different. A developer will be much further along assuming that Windows CE development is its own beast to be learned, than assuming they just have to do things a little differently from Win32 development.

The Microsoft Windows CE web site lists the technical paper, *Porting Windows 95 Programs to Windows CE* 60. This paper claims that Windows 95-based applications can be ported with less effort than writing from scratch. This may be true at times. We tried porting a Win32 Windows for Workgroups/DOS program, and found we had far less problems just writing the program from scratch. The *Porting*… paper lists the major porting concerns as follows: understanding the differences between the Win32 API and the WinCE API; understanding the differences between the standard MFC and MFC for Windows CE; memory limitations; energy limitations; varying hardware; and differences in testing.

The WinCE API is smaller than the Win32 API and supports a reduced feature set. For example, fewer windows styles, colors and fonts are supported. The WinCE API has specific extensions to support the varying hardware capabilities of different devices, such as touch screens. While designed to match the functionality of standard MFC, MFC for Windows CE has differences in available classes and supported features. RAM is generally limited on a Windows CE device. A Windows CE application should be written to use as little memory and storage as possible. A Windows CE program should also be written to conserve on energy consumption. Unnecessary computations should be removed, and an application should shut down unneeded features when idle. To best handle hardware differences on various target platforms, an application should be written for the lowest common denominator of functions and features. Testing with the emulator may only be helpful for devices using standard Windows CE. Nonstandard hardware platforms will need another method for application testing 60.

We found most of the information in *Porting Windows 95 Programs to Windows CE* to be helpful. Several areas are mentioned as potential problems: API functions, MFC, testing and emulation, using Unicode, and communications issues. Although the information is accurate, we found some either misleading or not sufficiently stressed. Specifically, the porting issues occurring with API functions, MFC and Unicode need to be stressed since the resulting problems can be considerable.

The *Porting*… paper states that only a subset of Win32 API is supported. However, we find this very misleading since almost every Windows CE API function has a usage definition that is slightly different from the Win32 definition. We also found that some functions change between versions of Windows CE. Either different names for the same function occur, or functions with the same names do different things. Although more information is now available, we found it difficult to get accurate information on the functions that make up the WinCE API for different versions of Windows CE. We did find a complete list of Windows CE API functions 57 that appeared to coincide with the information in the Windows CE help. Unfortunately, we discovered that functions were listed which were in fact not defined. For example, the following FTP related functions are listed in the help and the in API lists, but are not defined in a header file or library: `FtpCommand()`, `FtpCreateDirectory()`, `FtpDeleteDirectory()`,
**FtpDeleteFile(), FtpFindFirstFile(), FtpGetCurrentDirectory(), FtpGetFile(), FtpOpenFile(), FtpPutFile(), FtpRemoveDirectory(), FtpRenameFile()**, and **FtpSetCurrentDirectory()**. We also found functions that did have a prototype in a header file, but were not defined in the corresponding library. The NT utility DUMPBIN was very helpful in determining if a function was defined.

MFC is stated to have significant differences between Windows CE and Win32. However, what is not really mentioned is that MFC can automatically pull in various libraries that can cause compilation or linking failures. We found this out via link errors and Windows Knowledge Base articles.

The need for string conversion from Unicode to **char** is easily the most understated problem. A new paper, *Ten Tips for Programming for Microsoft Windows CE* ⁶¹, does a better job of addressing this issue, but fails to mention some critical functions and macros. Even if one is not porting a Windows 95 program, one will have to deal with string conversion. For example, all of the low level Winsock API commands take and return **char** strings, while the screen out commands take Unicode. Unicode / **char** conversion is compounded by developing for different versions of Windows CE, while trying to maintain one version of code. This issue will be discussed below in section D.

V. C. MFC versus Win32 API

MFC is provided for Windows CE and one can find MFC examples that compile, link and work on a Windows CE device. Nonetheless, our primary programming reference, *Programming Microsoft Windows CE*, warns against using MFC and encourages using API functions. Boling states that most Windows CE devices do not include the MFC library in their ROM in order to save space. Thus, the MFC and OLE32 DLLs required by a MFC application must be downloaded with the application. Even then, using MFC is problematic. For example, Windows CE 2.0 PPCs do not support MFC. Another consideration is whether the target device will have sufficient RAM to handle an MFC application. An API program is typically smaller, more efficient and has a smaller footprint when running than an MFC program. Since Windows CE devices are resource poor compared to desktop computers, the amount of necessary RAM for an application is a far bigger concern for a Windows CE developer than for a NT developer ²⁰.

Although *Microsoft Windows CE Programmer’s Guide* did not specifically warn against using MFC, we noticed that most of their example code used API functions and not MFC ⁵⁷. We took the hint and chose to develop our applications using API functions.

V. D. Development Problems for Different Windows CE Versions

The various SDKs and Toolkits do a reasonable job alerting one to potential difficulties in compiling the same code under different versions of Windows CE. The obvious cause is the change in supported functions and features among the different versions. For example, ICMP functions, which may be used for a ‘ping’ program, are supported in Windows CE 2.11 but not in Windows CE 2.0.
Since our primary environment was Windows CE 2.11, we wrote and tested our code for Windows CE 2.11 and then ported our application to Windows CE 2.0. We ran into one serious problem caused by Unicode / char conversions. As previously mentioned, some Windows CE functions use char strings while others require Unicode strings; hence the need for string conversion. This detail was compounded by the fact that, sprintf(), the function to write a formatted char string, is not supported until Windows CE 2.11.

In Windows CE 2.0, if one would like to print to the screen and to a log file something like:

    sprintf(outbuf, “Log in failed. Server response was %s”, returnMsg)

one must:

- convert returnMsg to Unicode using mbstowcs();
- create the formatted string with wsprintf() using the TEXT() macro within wsprintf();
- print that Unicode string to the screen;
- convert the string back to char with wcstombs();
- and lastly print that char string to a file.

If one wishes to send a command to a socket in early versions of Windows CE, prior to Windows CE 2.11, one must:

- create the command as a Unicode string with wsprintf();
- convert the string to char with wcstombs();
- and give the char string to send().

This may not seem like a huge problem, but it is a tedious and pervasive process that unnecessarily increases the size of the code. Our solution was to write our own, limited sprintf function.

Another problem that occurred was caused by the changes in functions handling files. While fopen() and fclose() are defined in Windows CE 2.11, one must use the functions CreateFile(), WriteFile(), and CloseHandle() in the older versions of Windows CE.

We suspect that fewer problems would occur if a program were first written for an older version of Windows CE and then ported to a newer version. The drawback to this approach would be that a developer may not be able to easily take advantage of features in the newer version.

V. E. Emulator Testing versus Hardware Testing

The Windows CE Toolkits do provide an emulator to allow a WCE application to be run on the desktop development computer in place of a real Windows CE device. This is quite convenient as it allows a developer to test a program on the different versions of Windows CE without having different Windows CE devices. The drawback occurs in the fact that a program is not tested on a real device. Some Microsoft information states that emulator testing is only a problem for applications aimed at nonstandard hardware devices. However, a developer is first alerted to potential problems in the Windows CE Toolkit SDK release notes. The release notes warn that the ability of a program to run on the emulator does not mean it will run on a device X. In addition, successful tests on device X do not guarantee that a program will run on device Y. Problems occur because different devices are using different processors and have different
hardware designs. The SDK release notes and the Windows CE help files list many compilation and link errors that are processor specific.

We found an interesting variation to this problem. Building our programs that included **winsock.h** failed for the emulator in Windows CE 2.00 and 2.01, but succeeded for 2.11. Even so, the build succeeded for the release versions of all target processors. Specifically, we received a linking failure on **winsock.h** for the X86em debug and release platforms. These programs linked correctly and work correctly on the primary and secondary test devices for this project. Although the cross compilations completed with no errors and no warnings, we do not know if our programs work correctly on the remaining Windows CE platforms and environments as we do not have access to these devices.

V. F. Development Problems for Different Hardware Platforms

Windows CE devices using the same version of Windows CE can have different processors, different display sizes and have different user input. For example, Windows CE 2.11 currently supports four different processors, four different display sizes and three different input methods – keyboard, touchpad and touchscreen. When developing a program, a Windows CE developer must decide what platforms will be supported. Once the target platforms are known, a developer needs to consider the variations in screen display, input methods and processors that occur. If all variations are to be supported, a developer must then decide how best to design the program.

A developer has essentially three options. The first option is to create one program, which accommodates all platforms without limiting the functionality of any platform due to another. For example, multiple dialog boxes could be included and displayed only on the appropriate screen size. While providing the best experience to a user for all platforms, the resulting source code will be complex and the executable could be larger than necessary if extra functions or dialog boxes are included.

Another design option is to create multiple versions of the program – one version for each platform. Thus, no device would be limited by another, the source code for each version will be simpler than that from option one, and the size of the program will be as small as possible. The drawback here affects the developer, who would have to maintain multiple versions of code. Multiple versions of source code for one program makes errors of omission probable – meaning a fix or change is added to one version, but accidentally omitted from another.

The third design option is to maintain one version of code, but limit the program by the device supporting the least functions and features. This option is recommended by Microsoft – write an application for the lowest common denominator of functions and features. Although a program may not use all available features on all target devices, this approach keeps the program small, keeps the source code simple, and makes code maintenance easier on a developer. I chose this design method.

Once the target platforms are chosen and a design method is selected, a developer must ensure to accommodate the various targets. A program will need to handle the different input methods and
the different Windows messages generated by the OS. Dialog boxes will need to fit within the screen display of the device having the smallest resolution.

A developer should also be aware that different errors can occur with different targets. As noted in section E, processor specific compilation and link errors are already known to occur. Processor specific problems are nice because these generally cause the program build to fail. Problems caused by differences in screen display or input methods are more difficult to spot. A program will compile, and problems can remain hidden until a developer or a user runs the program on a real device. Nothing we read stressed the problems that could occur if a developer does not take into account the variations in screen size or input methods on different Windows CE devices. If a program uses a function that depends upon a specific input method, for example a touchscreen message, this part of a program may freeze on a Windows CE device that does not have a touchscreen. Problems caused by differences in screen display may be revealed by use of the emulator provided in the Toolkit. Yet, the emulator cannot be depended upon completely. As previously mentioned, even if a program compiles and works for the emulator, it may behave differently on an actual device.

When we ported our programs from the Jornada 820, with its 640 X 480 display, to the 620LX, I found that I did need to resize the dialog boxes for the 640 X 240 display. We could have created two versions of the programs—one for each display type. Since we wished to maintain just one version of the program, we chose to resize the boxes for the smaller display. While this is easier on the developer, some of the resulting boxes do feel cramped on the larger display of the Jornada 820.

V. G. Problems with the SDKs

The SDK that we used for this project was the *Handheld PC Professional 3.0 SDK*, since our primary device is an H/PC Pro. We were able to cross compile for the older Windows CE versions using this SDK. For the most part, we had few problems with this SDK. However, it had the feeling of a late beta. The help is not well organized, and, as we previously mentioned, functions are listed in the help that are not actually defined in the existing libraries. We suspect that these missing functions will be implemented in the future, but the help files were not edited to reflect this.

One interesting problem that we discovered occurred with our first network program. We attempted to write a ‘ping’ using Windows CE ICMP functions, but received link errors on the symbols for the functions `IcmpCloseHandle()`, `IcmpSendEcho()`, and `IcmpCreateFile()`. We verified with the `/verbose` link setting that the linker did find the library `icmp.lib`, searched it, and failed to find the corresponding symbols. Using DUMPBIN, we verified that the needed symbols were in fact defined in `icmp.lib`. We posted our findings to the MSDN new groups and emailed the Microsoft Windows CE bug report. A bug report support person acknowledged the problem and stated they would “look into it”, but never responded further. We received one suggestion from the MSDN newsgroup that did not work, and received another later that we did not implement as we had moved on to other network clients.
VI. Developed Software

The programming goal of this project was to produce network utilities for Windows CE devices. Two utilities were successfully written in the course of this project, a time client and an FTP client. We will describe some of the work to develop the FTP client.

VI. A. E_FTP – a FTP Client

After entering the fully qualified domain name, user name and password, E_FTP allows a user to log onto any FTP server using port 21. E_FTP creates a log file, e_ftplog.txt, to store connection information and command messages to assist with troubleshooting. This client also provides a test routine, which makes a control connection to Microsoft’s FTP site, changes the transfer mode and requests the connection status. This routine logs its activity in a file, e_chklog.txt, which can also be helpful to troubleshoot connection problems. Once connected to a server, a user may change, create or delete remote directories, plus list the contents of a directory. When viewing the contents of a directory having a large number of files, the number of files may exceed the screen buffer size, which is set at 100 lines. To assist with this problem, the directory contents are also saved in a file, LIST.TXT. A user may retrieve and store one or multiple files, abort transfers, and delete remote files one at a time. E_FTP includes a large amount of error checking for both the network connection and file handling functions. We also attempted to provide useful error statements for different problems. To assist with troubleshooting problems occurring during a ‘Get Multiple Files From Host’, the list of names matching the file name pattern is saved in NLIST.TXT. There a user can see exactly what the function attempted to retrieve. User help is incorporated into the program as brief comments on each dialog box. To help keep the size of the program small, we did not include a detailed help file. Detailed help is provided in a readme.txt file, which a user may choose to download.

The only known limitation in E_FTP occurs with number of files that may be retrieved by mget or ‘Get Multiple Files From Host’. If the file name pattern generates a list of matching names larger 4K (4096 bytes), only the final 4K will be retrieved. This limit is not affected by file sizes, but only by the number of and size of the file names. To determine if this problem has occurred, a user only needs to check the size of the file NLIST.TXT. If the size of that file is greater than 4096 bytes, then the user should split the transfer by using patterns that will select a smaller number of files. For example, instead of using *.*, a user could do multiple transfers – first using a*.*, then b*.*, and so on. This limitation only occurs with mget.

VII. Recommended Changes for CE SDKs

After working with the Windows CE Toolkits and SDKs, we found this software development package to be very useful. Overall, the Windows CE Toolkits worked quite well. Nonetheless, we did find a few items that should be improved to better serve the developer.

As noted previously, functions are listed in the help that are not defined in the libraries. The Windows CE help files should correctly reflect the functions currently available in and supported by the SDK. We found the inaccurate information quite frustrating at times – especially when we were working with the various FTP functions. These functions were beautifully defined in the VI. Developed Software
help, and listed in one of our programming references. They just did not make it into the libraries. Microsoft should consider improving the accuracy of the help files.

In addition, the help files within the Windows CE Toolkits are not well organized. The API functions are one large list with no indexing by usage or type. For example, it would be helpful to have a page that listed all API functions related to sockets, and another page listing all functions related to data conversion i.e. char to Unicode. Instead, a user must hunt through the list or get the information from a second source. Our solution was to create our own help by saving pages under topics in the ‘favorites’ section of help. This made it far quicker for me to find related functions. Microsoft should also consider improving the organization of the Windows CE help in the SDK.

In the time since we completed the programming portion of our project, we have noted that Microsoft has reorganized and improved the online help available through MSDN. In early 1999, little information was available beyond the three books and the few technical papers that we have already mentioned. Moreover, the information was somewhat difficult to find. Now the MSDN online library has a huge selection of technical references and background papers specifically for Windows CE. We have not reviewed the content of most of the papers, but the selection of topics is excellent. Microsoft also now offers many online seminars that cover user and developer Windows CE topics.

VIII. Summary

The objective was to determine what challenges are present before we would provide a laboratory experience in an operating systems course. During the course of this project, Windows CE Toolkit for Visual C++ 6.0 was investigated as a development tool for Windows CE applications targeted at embedded devices. The original plan for this project was to include the Java programming language. This was not done since the corresponding Windows CE Toolkit for Visual J++ 6.0 was not released by Microsoft.

Windows CE Toolkit for Visual C++ 6.0 was found to be a very useful tool for Windows CE application development. The Toolkit and SDK are installed on a desktop computer running Windows NT as an add-on to either Microsoft Visual Studio 6.0 or Microsoft Visual C++ 6.0. The Toolkit runs within Visual C++ 6.0 adding options to create a Windows CE application project, and allowing a developer to choose which processor platforms to include in the project. Once the project is built, the cross compilation feature of the Toolkit allows a developer to easily change the target processor or the target version of Windows CE used for the program build. Windows CE Toolkit for Visual C++ 6.0 supports all current versions of Windows CE and all current processor platforms. In addition, an emulator is provided to allow a Windows CE program to be run on the development desktop computer instead of the actual Windows CE device.

The primary Windows CE target device for this project was the HP Jornada 820, which has a StrongARM processor and runs Windows CE 2.11. The secondary device was the HP 620LX, which has a SH3 processor and runs Windows CE 2.0. Thus, the cross compilation feature of the
Windows CE Toolkit for Visual C++ 6.0 could be investigated, plus the ease of porting an application from one version of Windows CE to another.

Various paths and pitfalls of Windows CE development were discovered in the course of this project. At the outset of this project, information on Windows CE programming was limited, but several good references were found and are listed in the body of this report. Although Windows CE development looks and feels like Win32 development, a developer will be much further ahead if Windows CE development is treated like a new and separate beast to be learned. Porting a 16-bit Win32 API Winsock program to Windows CE proved extremely difficult, and was abandoned in favor of development from scratch. Programming references favored using the Windows CE Win32 API functions and recommended cautious use of MFC functions. When porting a program from Windows CE 2.11 to Windows CE 2.0, differences in file handling functions and string functions had to be overcome. Although testing a program on the built-in emulator can be helpful, the true test is to try a program on the actual device. Developing programs for different Windows CE devices require a developer to be aware of how a program is affected by the screen size, input method and processor on each target device. A developer must decide between maintaining multiple versions of code that use the separate features on each device, or have one version of code that works on the lowest common denominator between the target devices. Although it is a nice development tool, the Windows CE Toolkit for Visual C++ 6.0 has the feel of a beta product. Help files list functions that are not defined in libraries, and some libraries do not link with their defined symbols correctly.

Bibliography
12. Stangel, M., and Bharghavan, V. Improving TCP Performance in Mobile Computing Environments. IEEE International Conference on Communications '98.
   http://users.neca.com/vmis/wsockexp.htm
60. Microsoft Corporation. *Porting Windows 95 Programs To Windows CE*. 
   http://msdn.microsoft.com/library/techart/msdn_10ce_tips.htm
63. *MSDN Online Seminars*. http://msdn.microsoft.com/training/seminars/events_ce.asp

ESTHER V. V. REED

MATT MUTKA
Mobile Computing - Brief Overview - Mobile Computing is a technology that allows transmission of data, voice and video via a computer or any other wireless enabled device without having to be conn. Mobile software is the actual program that runs on the mobile hardware. It deals with the characteristics and requirements of mobile applications. This is the engine of the mobile device. We examine the applicability of mobile software codes to perform networking tasks in wireless and mobile computing environments. We contend that the advent of wireless technologies during the past decade has turned computer networks increasingly complex to manage. In particular, factors such as context awareness and user mobility are now crucial in the design of communications protocols used by portable devices with moderate to severe bandwidth and battery power limitations. Unlike hard-coded communication protocols that fulfill a specific need, mobile software agents can be deployed to deal with mobile app development tools allow developing and creating native mobile apps as well as cross-platform mobile apps with various functionalities as we need. These tools also help to develop codeless mobile apps for organizations that cannot afford to hire a developer.

Why use App Development Software? App Development software helps to customize the functionalities of mobile applications and allows adding new features easily. It also improves the user experience and navigation of the mobile application.

How to select the Best Mobile App Development Tool? This document describes the mobile software development lifecycle, taking a look at UX design, UI design, development, stabilization, distribution, and more. Development is the most resource intensive phase, this is the actual building of the application. Stabilization is when development is far enough along, QA usually begins to test the application and bugs are fixed. Often times an application will go into a limited beta phase in which a wider user audience is given a chance to use it and provide feedback and inform changes. Deployment Mobile devices open up a whole new way to interact with computing, the web, and even corporate infrastructure. The inception stage is all about defining and refining the idea for an app.