

國立交通大學

資訊科學系

碩士論文

ART 可適性及 GPRS 延遲的改善

ART Adaptability and GPRS Latency Improving



研究生：朱文如

指導教授：袁賢銘 教授

中華民國 九十四 年 六月

ART 可適性及 GPRS 延遲的改善
ART Adaptability and GPRS Latency Improving

研 究 生：朱文如

Student : Wen-Ju Chu

指 導 教 授：袁賢銘

Advisor : Shyan-Ming Yuan

國立交通大學
資 訊 科 學 系
碩 士 論 文



Submitted to Institute of Computer and Information Science
College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer and Information Science

June 2005

Hsinchu, Taiwan, Republic of China

中華民國九十四年六月

ART 可適性及 GPRS 延遲的改善

學生：朱文如

指導教授：袁賢銘

國立交通大學資訊科學系

摘要

近年來，行動裝置的普及使得行動應用程式蓬勃發展，原本 J2ME 為行動應用程式執行的主流平台，近來隨著 .NET CF 的崛起，原有的行動應用程式必須移植到 .NET CF 平台，使得移植的需求愈來愈大，但兩平台互不相容，讓應用程式的移植面臨許多挑戰，而且兩平台之間缺乏自動或半自動轉換的工具，使得從 J2ME 移植行動應用程式到 .NET CF 變得費時且困難。

為了解決上述問題，我們將 ART 的應用範圍從原本的 J2ME 擴大到 .NET CF 平台，使用 ART 平台開發的行動應用程式只需要開發一次，便能在這兩個平台上顯示與操作。依 ART 架構，我們從網路連線、資料處理、使用者介面及使用者互動處理四個層面分析 J2ME 和 .NET CF 平台的異同，提出移植 ART 的方法。

在 ART 平台執行的應用程式需要透過網路將程式執行結果以 ART 訊息的方式從 ART server 傳送到 ART client 上顯示。GPRS 是目前行動裝置最普遍的上網方式，但是由於 GPRS 的頻寬小且延遲時間長，使得 ART 訊息無法有效率的傳送到 ART client。我們根據 ART 系統的特性，提出一個演算法來縮短 ART 訊息在 GPRS 上傳送的時間。

ART Adaptability and GPRS Latency Improving

Student: Wen-Ju Chu

Advisor: Shyan-Ming Yuan

Department of Computer and Information Science

National Chiao Tung University

Abstract

Nowadays, the popularity of mobile devices enhances the development of mobile application. J2ME was the most popular platform for mobile applications originally, but the growth of .NET platform requires a lot of porting of mobile applications from J2ME to .NET platform. Since these two platforms are not compatible, it is challenging in terms of porting applications. Lacking of automatic or semi-automatic transform tool between these two platforms makes porting from J2ME to .NET becomes difficult and inefficient.

To solve problems mentioned above, we extend the appliance of ART from J2ME to .NET platform. Therefore mobile applications developed by ART can be built just once and can display and operate in both platforms. We analyze the difference between J2ME and .NET in four parts: network communication, ART message handling, user interface display, and user interaction handling. We provide methods of porting ART depending on these four parts.

Mobile applications running in ART send the executing result from ART server to ART client by ART messages through network. GPRS is the most popular network on mobile devices. Due to the low bandwidth and high latency of GPRS, transferring ART messages to ART client is inefficient. Based on the characteristics of ART system, we represent two algorithms to decrease the transferring time of ART messages through GPRS.

Acknowledgement

首先要感謝我的指導教授袁賢銘教授，謝謝教授這兩年來的指導以及對這篇論文的指引，讓我學習到更多更廣的專業知識及思考方式。感謝各位博士班學長鄭明俊、邱繼弘、高子漢、葉秉哲及吳瑞祥，謝謝你們寶貴的建議及幫助，讓我能順利完成這篇論文。感謝各位實驗室同仁，尤其是慧雯、倫武、上謙、憲昌，謝謝你們平時的協助及鼓勵。感謝我的室友慧文、怡緯及好朋友心怡、筱菁、易虹、雅文，在我煩悶的時候，有你們的陪伴及鼓勵，讓我有動力繼續前進。感謝遠在美國求學的學妹詩妮以及好友鶴文，謝謝你們給我英文論文寫作上的建議，尤其是鶴文，在忙碌的工作之餘還抽空指導我英文。感謝我的妹妹文君，謝謝妳的善解人意。最後要感謝我的父母，謝謝你們從小的教誨，而且永遠無條件的支持我，讓我充份發揮自己的興趣，我能有今天小小的成就都是來自於你們，謹以此篇論文感謝你們的養育之恩，謝謝！



Table of Contents

Acknowledgement	iii
Table of Contents.....	iv
List of Figures	vi
List of Tables	vii
Chapter 1 Introduction	1
1.1. ART (Adaptive Remote Terminal)	2
1.2. Motivation	3
1.3. Objectives	4
1.4. Organization	4
Chapter 2 Background and Related Works.....	5
2.1. J2ME (Java 2 Platform, Micro Edition)	5
2.2. .NET CF (.NET Compact Framework)	6
2.3. GPRS	8
Chapter 3 Porting ART Client from J2ME to .NET	11
3.1. Network Communication	11
3.2. ART Message Handling.....	12
3.2.1. Numeric Data Transformation	13
3.2.2. String Data Transformation	14
3.3. User Interface Display	16
3.4. Event Handling	21
3.4.1. Event Subscription.....	21
3.4.2. Paint Event Handling.....	23
Chapter 4 GPRS Latency Improving.....	27

4.1. Test Bed	27
4.2. ART Applications	28
4.3. GPRS Latency Factors in ART	29
4.3.1. Test Result of Data Size	30
4.3.2. Test Result of the Nagle Algorithm	31
4.4. Algorithms to Improve GPRS Latency in ART system	33
4.4.1. Algorithm 1	33
4.4.2. Algorithm 2	35
Chapter 5 Conclusions and Future Works	38
5.1. Conclusions	38
5.2. Future Works	39
References	42



List of Figures

Figure 1-1: The Architecture of ART system	3
Figure 2-1: Java 2 Platform, Micro Edition.....	6
Figure 2-2: The GPRS network architecture	9
Figure 3-1: Establishing a socket connection in ART client on J2ME platform	11
Figure 3-2: Establishing a socket connection in an ART client on .NET CF platform	12
Figure 3-3: The format of an ART message	13
Figure 3-4: US-ASCII Encoding in Java.....	15
Figure 3-5: ASCII Encoding in .NET.....	15
Figure 3-6: User Interface Components in J2ME.....	17
Figure 3-7: An ART Client in J2ME.....	20
Figure 3-8: An ART Client in .NET CF.....	20
Figure 3-9: Event Model in J2ME and .NET CF.....	22
Figure 3-10: Event Subscription in J2ME.....	22
Figure 3-11: Event Subscription in .NET CF	23
Figure 3-12: Canvas Displaying Process of an ART Client in J2ME.....	24
Figure 3-13: Canvas Displaying Process of an ART Client in .NET CF.....	25
Figure 3-14: Modified Canvas Displaying Process of an ART Client in .NET CF.....	26
Figure 4-1: Test Bed for GPRS latency	28
Figure 4-2: GPRS Latency of Different Packet Size.....	31
Figure 4-3: The effect of the Nagle algorithm for GPRS latency in ART system.....	32
Figure 4-4: GPRS latency of Algorithm 1 with Different Waiting Time in ART system.....	34
Figure 4-5: GPRS latency of Algorithm 2 with Different Waiting Time in ART system.....	36

List of Tables

Table 2-1: The comparison between J2ME CLDC MIDP and .NET CF	7
Table 3-1: The encoding types in Java	15
Table 3-2: The encoding types in .NET	15
Table 3-3: UI Commands in an ART message.....	16
Table 3-4: User Interface Composition in .NET CF corresponds to J2ME.....	18
Table 4-1: The Differences amount ART applications	29
Table 4-2: Improvement of Algorithm 1 for GPRS Latency	34
Table 4-3: Algorithm 2 for GPRS Latency Improvement	36
Table 4-4: Algorithm 2 for GPRS Bandwidth Saving	37



Chapter 1 Introduction

It is more and more common that people are checking emails and downloading other information directly to their cell phones and PDAs. For developers, this means an increased demand to create applications for mobile devices. ART [1] is an adaptable developing platform for mobile applications; it makes the development of mobile applications more rapidly. Mobile applications developed by ART are called ART app. (ART application), which can be adapted to many kinds of displaying and operational methods. Mobile devices which support J2ME MIDP [2] platform can display and operate ART app. by ART client. Mobile devices do not support J2ME MIDP can display and operate ART app. by WAP, html browser. Since the mobile devices of .NET CF [3] (.NET Compact Framework) platform get more and more popular, but ART client can not execute on .NET CF platform. Therefore we develop the ART client for .NET CF platform in order to extend the display capability for mobile devices of ART. Based on the ART architecture, we describe the porting issues in four parts: network communication, ART message handling, user interface display, and user interaction handling. We use C# language [4] in .NET platform because it was designed especially for .NET platform.

Mobile applications running in ART send the executing result from ART server to ART client by ART messages through network. ART client usually executes on mobile devices which usually use IEEE 802.11 or GPRS [5] to communicate with ART server. For now, GPRS is the most often used facility for most cell phones to access network. However, the low bandwidth and high latency of GPRS makes it inefficient to transfer of ART messages. Besides, ART messages are transferred through socket connection, since the default setting of socket and network connection may not suit ART system, we adjust these settings based on

the characteristics of ART system and provide an algorithm to decrease the transferring time of ART messages through GPRS.

1.1. ART (Adaptive Remote Terminal)

ART is a platform used to develop and execute the mobile applications. It is a client-server model system, and clients communicate with servers by means of an asynchronous message-delivery mechanism. Figure 1-1 shows the architecture of ART system.

The purposes of ART system are:

- To save time of developing mobile applications – the mobile applications can be written once and run on every platform. It can reduce providers' porting effort and the cost of program maintenance.
- To extend the computing power of mobile devices – we can serve lots of work which cannot be supported just within the mobile devices by the stronger computing power and richer resources on the server side.

Mobile applications developed by ART system are called ART app. (ART applications), which execute on ART server. The executing results are sent to ART client by ART messages. Users can send requests to the ART server by operating GUIs on the ART client. Then ART server will run the application designate by the users' requests and send the executing results as response back to the client side through network connection. The client of ART generates the new GUIs to display the executing results by server's response.

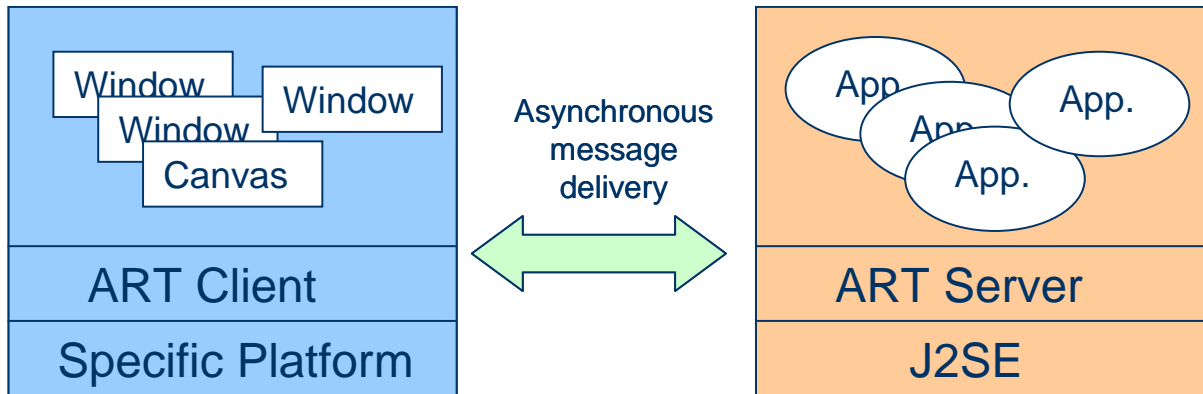


Figure 1-1: The Architecture of ART system

1.2. Motivation

The motivations of this paper are as follows:

- .NET CF platform becomes more and more popular – Owing to the popularity and growing computing power of mobile devices, the needs of mobile applications increase. However, the difference of mobile devices makes the development of mobile applications difficult. J2ME and .NET CF are the main platforms for mobile devices now. They have some similarities, but they are not compatible. They have many differences, such as data format, user interface and supporting packages. If a mobile application wants to execute on both J2ME and .NET CF, it must be written for each platform. One of purposes of ART system is making mobile applications display and operate on every platforms while write once. Therefore we extend the displaying ability of ART to display and operate mobile applications in .NET CF platform.
- High GPRS Latency – GPRS, which is widely used currently, enables mobile devices to surf the Internet. However, low bandwidth and high latency of GPRS makes operations of a mobile application spent most time in GPRS transferring.

Inefficient transferring of ART messages – In scenario 3 of [6], the experimental results say that when the size of a packet becomes larger, its transmission time becomes shorter. Since ART message size is very small, this makes its transferring inefficiently. More over, the default network setting over application layer of OSI network model may not suit ART system.

1.3. Objectives

The objectives of this paper are as follows:

- Expand adaptability of ART – ART system makes developers to write mobile applications once and adapt to different platforms and displaying methods. Now, ART applications can be displayed and operated by WAP, html browser and J2ME MIDP, we expand adaptability of ART to .NET CF platform.
- Decrease GPRS latency of ART transferring – Owing to inefficient transferring of ART messages through GPRS, we observe the characteristics of ART messages transferring in various ART applications and network setting of application layer. An algorithm is provided to decrease GPRS latency of ART transferring.

1.4. Organization

The organization of this paper is as follows: Chapter 2 describes J2ME, .NET CF, GPRS and related works about our research. Chapter 3 represents the issues of porting ART client from J2ME to .NET CF. Chapter 4 shows the test plans and results of GPRS latency and the algorithm to improve GPRS latency.

Chapter 2 Background and Related Works

2.1. J2ME (Java 2 Platform, Micro Edition)

The Java 2 Platform, Micro Edition is the edition of the Java 2 platform targeted at consumer electronics and embedded devices, including mobile phones, pagers, personal digital assistants, set-top boxes, and vehicle telematics systems.

The J2ME technology consists of a virtual machine and a set of APIs defined through the Java Community ProcessSM program by expert groups that include leading device manufacturers, software vendors and service providers.

Figure 2-1 has shown the J2ME architecture, which comprises a variety of configurations, profiles, and optional packets. They cooperate to build complete Java runtime environments. Each combination of the three is optimized for the memory, processing power, and I/O capabilities of a related category of devices. The following describes the three parts:

- Configurations comprise a virtual machine and a minimal set of low-level APIs. They provide base functionalities for a particular range of devices that share similar characteristics. Currently, there are two configurations: the CLDC [7] (Connected Limited Device Configuration) and the CDC (Connected Device Configuration).
- A profile provides a set of higher-level APIs that further define the application life-cycle model, the user interface, and access to device-specific properties. It combines with a configuration to provide a complete runtime environment for a specific kind of device. For example, MIDP [8] (Mobile Information Device Profile) combines with CLDC provides the complete runtime environment for mobile

phones and entry-level PDAs.

- Optional packets offer standard APIs targeted at a certain kind of technology. They extend the capabilities of the Java platform.

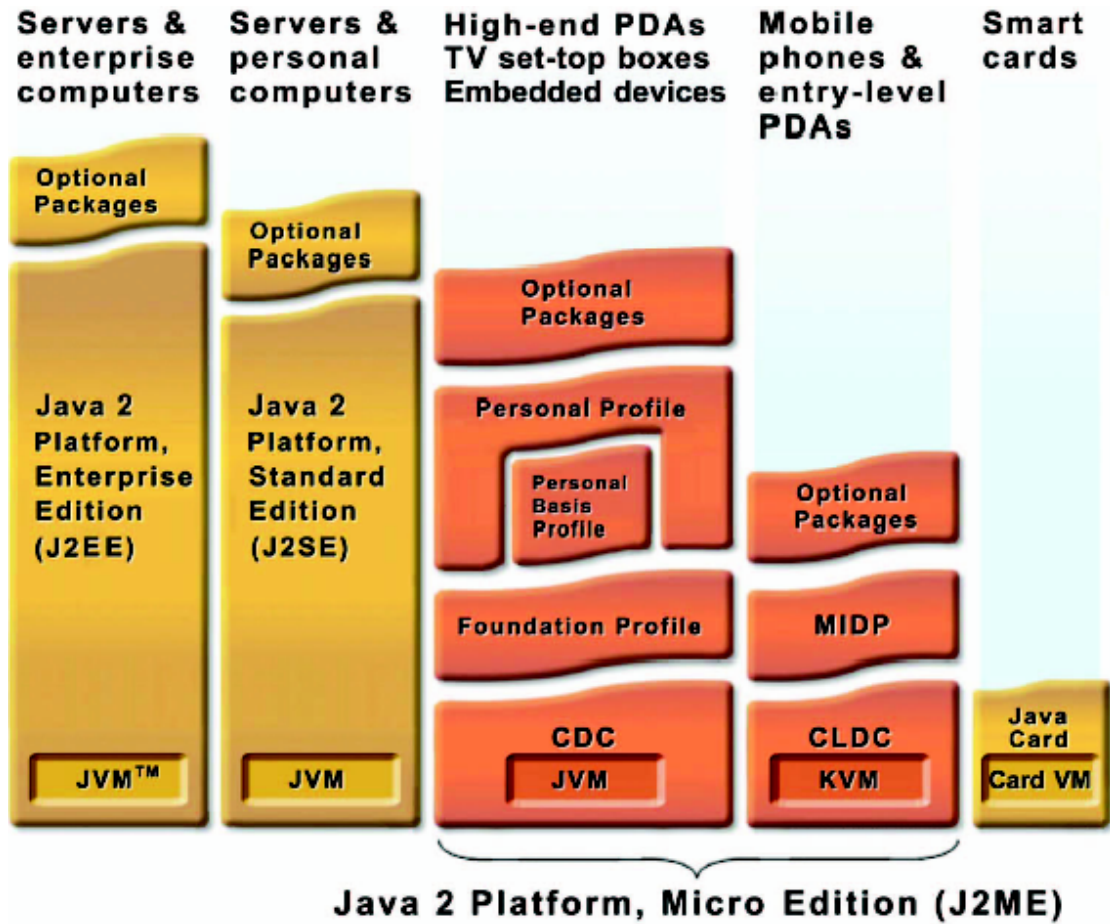


Figure 2-1: Java 2 Platform, Micro Edition

source: <http://java.sun.com>

2.2. .NET CF (.NET Compact Framework)

The .NET CF (.NET Compact Framework) is a development and execution environment for managed applications on Windows Mobile devices, it is designed specifically for

resource-constrained devices, such as PDAs (personal digital assistant) and smart mobile phones. The .NET CF is a subset of Microsoft's .NET Framework.

The Microsoft's .NET Framework is an integral Microsoft Windows® component for building and running Windows applications. It has two main parts: the common language runtime (CLR) and a unified set of class libraries, including ASP.NET for Web applications and Web services, Windows Forms for smart client applications, and ADO.NET for loosely coupled data access.

Items	J2ME CLDC MIDP	.NET CF
Device requirement	Powerless	Powerful
Platforms	Most mobile platform	Pocket PC, Windows CE
Language support	Java	C#, VB.NET
Byte code compatibility	Not compatible with J2SE	Standard .NET CLR
API compatibility	Partial J2SE API with additional optional packages	Subset of .NET
Native APIs	P/Invoke	N/A
Development tools	JDK, JBuilder, etc.	VS.NET 2003
Specification process	Community	Microsoft
User interface	Less UI components	More UI components
Communication ability	HTTP(s)	Sockets, HTTP(s), SOAP

Table 2-1: The comparison between J2ME CLDC MIDP and .NET CF

Code that requires the CLR at run-time in order to execute is referred to as managed code. The CLR is responsible for managing execution of code that runs on the .NET Framework. Managed code is compiled down to a combination of MSIL (Microsoft Intermediate Language) and metadata. These are merged into a Pre Execution Environment (PE) file, which can then be executed on any CLR-capable machine. When you run this

executable, the JIT starts compiling the IL down to native code. The result is that all .NET Framework components run as native code. The table 2-1 shows the comparison between J2ME CLDC MIDP and .NET CF.

2.3. GPRS

The General Packet Radio Service (GPRS) is a packet data service in GSM to access packet data networks across a mobile telephone network. It supplements today's Circuit Switched Data and Short Message Service. The main benefits of GPRS are that it reserves radio resources only when there is data to send and it reduces reliance on traditional circuit-switched network elements.



GPRS facilitates several new applications that have not previously been available over GSM networks due to the limitations in speed of Circuit Switched Data (9.6 kbps) and message length of the Short Message Service (160 characters). GPRS will fully enable the Internet applications you are used to on your desktop from web browsing to chat over the mobile network. Other new applications for GPRS, profiled later, include file transfer and home automation - the ability to remotely access and control in-house appliances and machines.

Theoretical maximum speeds of GPRS up to 171.2 kilobits per second (kbps), This is about three times as fast as the data transmission speeds possible over today's fixed telecommunications networks and ten times as fast as current Circuit Switched Data services on GSM networks.

the addition of Packet Control Units; often hosted in the Base Station Subsystems, mobility management to locate the GPRS Mobile Station, a new air interface for packet traffic, new security features such as ciphering and new GPRS specific signaling.



Chapter 3 Porting ART Client from J2ME to .NET

How an ART app. can be used on mobile devices depends on the capability of mobile devices. For example, on J2ME platform, an ART client is executed as a J2ME MIDP application to show and use ART applications. For some mobile devices without supporting J2ME, ART app. can be shown and used in WAP or http browser as well.

To execute an ART client on .NET CF platform to operate ART applications, this research ports an ART client from J2ME platform to .NET CF platform. According to the architecture of ART clients, four porting issues, network communication, ART message handling, user interface display and event handling, will be discussed next.



3.1. Network Communication

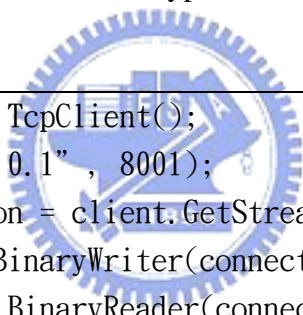
ART is a client-server model system which means an ART client communicates with an ART server through a socket connection. There are some differences between J2ME and .NET at establishing a socket connection. Figure 3-1 shows how to establish a socket connection in an ART client of J2ME platform:

```
1 connection=(StreamConnection) Connector.open("socket://127.0.0.1:8001");  
2 DataOutputStream os = connection.openDataOutputStream();  
3 DataInputStream is = connection.openDataInputStream();
```

Figure 3-1: Establishing a socket connection in ART client on J2ME platform

In J2ME, the IP and port of an ART server in the Connector.open() method are set to establishing a socket connection. The instance of DataOutputStream lets the ART client write primitive Java data types to an output stream in a portable way. The instance of DataInputStream allows the ART client to read primitive Java data types from an underlying input stream in a machine-independent way.

Figure 3-2 shows how to establish a socket connection in an ART client on .NET CF platform. The approach to establish a socket connection in .NET is similar to in J2ME. Among many classes of stream handling in .NET CF, this research chooses BinaryWriter class to replace DataOutputStream class, and BinaryReader class for replace DataInputStream class due to the ability of handling primitive data types in these two classes.



```
1 TcpClient client = new TcpClient();
2 client.Connect( "127.0.0.1" , 8001);
3 NetworkStream connection = client.GetStream();
4 BinaryWriter os = new BinaryWriter(connection);
5 BinaryReader ins = new BinaryReader(connection);
```

Figure 3-2: Establishing a socket connection in an ART client on .NET CF platform

3.2. ART Message Handling

An ART client communicates with an ART server using ART messages, in the format shown in Figure 3-3. An ART message includes the information about how to control and display an ART application. An ART client controls an ART application by generating a related ART messages, transforming it into byte array and sending to the ART server via socket.

isRaw (Boolean)	
Length of sourceID (int)	sourceID (String)
Length of destinationID (int)	destinationID (String)
Length of owner (int)	owner (String)
Length of group (int)	group (String)
Length of appName (int)	appName (String)
Length of uiName (int)	uiName (String)
Function (int)	
If (isRaw) rawData (byte[])	
If (! isRaw) length of data (int)	If (! isRaw) data (String)
If (! isRaw) length of dataExtra (String)	If (! isRaw) dataExtra (String)

Figure 3-3: The format of an ART message

An ART server runs on Java platform, however, an ART client runs either on J2ME or .NET CF platform. Some differences between Java and .NET platform about data transformation influence ART message handlings. The following subsections discuss these differences in two parts: numeric data and string data, and how to handle ART messages correctly between Java and .NET platform.

3.2.1. Numeric Data Transformation

In the process of sending and receiving numeric data via sockets, numeric data in ART messages should taken care of conversion issues in terms of what type of machine is on the other end of a connection. Specifically, how to convert numeric data from the local machine's format (host order) to the industry standard format for sending sockets data (network order) is needed to know.

The byte order of sockets data is the same big-endian as the one in the J2SE and J2ME.

On .NET platform, the byte order may be big-endian or little-endian, depend on the local machine's format (host order). In order to send and receive data through socket correctly, socket data should be transformed between network order and host order on .NET platform. .NET platform supplies the following methods to transform numeric data from host order to network order:

`IPAddress.HostToNetworkOrder(int)`

`IPAddress.HostToNetworkOrder(short)`

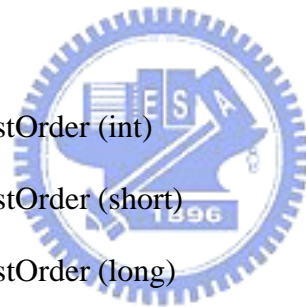
`IPAddress.HostToNetworkOrder(long)`

Similarly the following methods transform numeric data from network order to host order:

`IPAddress.NetworkToHostOrder (int)`

`IPAddress.NetworkToHostOrder (short)`

`IPAddress.NetworkToHostOrder (long)`



3.2.2. String Data Transformation

In order to send and receive ART messages correctly between an ART server and an ART client, string data should be encoded by the character set with the same definition. Both of Java and .NET provide many methods of converting arrays and strings of Unicode characters to and from arrays of bytes encoded for another character set. The character sets of Java and .NET are shown in table 3-1 and 3-2.

character set	Description
US-ASCII	Seven-bit ASCII, a.k.a. ISO646-US, a.k.a. the Basic Latin block of the Unicode character set
ISO-8859-1	ISO Latin Alphabet No. 1, a.k.a. ISO-LATIN-1
UTF-8	Eight-bit UCS Transformation Format
UTF-16BE	Sixteen-bit UCS Transformation Format, big-endian byteorder
UTF-16LE	Sixteen-bit UCS Transformation Format, little-endian byteorder
UTF-16	Sixteen-bit UCS Transformation Format, byteorder identified by an optional byte-order mark

Table 3-1: The encoding types in Java

character set	Description
ASCII	An encoding for the ASCII (7 bit) character set. ASCII characters are limited to the lowest 128 Unicode characters, from U+0000 to U+007f.
Default	An encoding for the system's current ANSI character set
Unicode	An encoding for the Unicode format in little-endian byte order.
UTF7	An encoding for the UTF-7 format.
UTF8	An encoding for the UTF-8 format.

Table 3-2: The encoding types in .NET

```
String=new String(byte[] bytes, "UTF-8")
byte[]=StringName.getBytes("UTF-8")
```

Figure 3-4: US-ASCII Encoding in Java

```
String=Encoding.UTF8.GetString(byte[])
byte[]=Encoding.UTF8.GetBytes(string)
```

Figure 3-5: ASCII Encoding in .NET

US-ASCII and UTF-8 character sets in Java corresponded to each of ASCII and UTF8

character sets in .NET. ART system uses UTF-8 and UTF8 character set in Java and .NET platform respectively to encode string data of ART messages. Figure 3-4 and 3-5 show individual program of encoding in Java and .NET.

3.3. User Interface Display

An ART server sends to an ART client ART messages with UI (user interface) commands. Table 3-3 shows the UI commands in an ART message. An ART client generates the user interface components based on the commands inside ART messages. Based on the ART client of J2ME MIDP, this research establishes the rules to convert user interface components from J2ME MIDP to .NET CF.

Command	Description
FORM_CREATE	Create a form to show user interface.
FORM_ADD_BUTTON	Add a button in a form.
FORM_APPEND_TEXTFIELD	Add an editable text field in a form.
FORM_APPEND_CHOICEGROUP	Add a choice group in a form.
CANVAS_CREATE	Create a canvas to show images.
CANVAS_PAINT	Add an image to a canvas.
CHOICEGROUP_APPEND	Add a choice item in a choice group.
CHOICEGROUP_SETCURRENTSELECTED	Set a choice item as selected in a choice group.
CHOICEGROUP_SETTITLE	Set the title of a choice group.
CHOICEGROUP_DELETE	Delete a choice item in a choice group.
TEXTFIELD_SETSTRING	Set the content of a text field.
INIT_FINISH	Add the ART system menu in a form or canvas.
DISPLAY_CHANGE	Display a form or canvas.

Table 3-3: UI Commands in an ART message

J2ME and .NET CF platform have many user interface components. The javax.microedition.lcdui package provides a set of features for implementation of user

interfaces for MIDP applications. Figure 3-6 shows the relationship of the classes for creating user interfaces in J2ME MIDP used in ART.

The user interface objects shown on the display device are contained within a Displayable object, such as Canvas, List, Alert and Form. At any time an application have at most one Displayable object to show on the display device through which user interaction occurs.

Screen class is a common superclass of all high-level user interface classes. A Form is a Screen that contains an arbitrary mixture of items, such as editable text fields and choice groups. In general, it contains any subclass of the Item class.

Command class is a construct that encapsulates the semantic information of an action. This means that Command contains only information about "command" not the actual action that happens when a command is activated. Besides, Command class may be contained within a Displayable object.

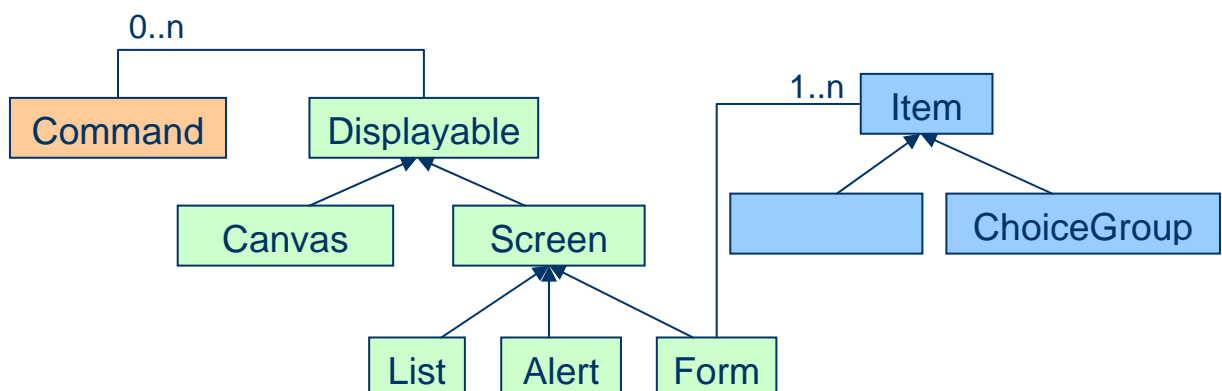


Figure 3-6: User Interface Components in J2ME

In .NET framework, System.Windows.Forms namespace contains classes for creating Microsoft Windows-based applications. Most classes within the System.Windows.Forms namespace derive from the Control class. The Control class provides the base functionality for all controls displayed on a Form which represents a window within an application.

However, some classes in J2ME, such as List and Alert, have not been defined in .NET. This research combines essential objects of .NET to compose such classes in J2ME. Table 3-4 shows the user interface compositions in .NET CF which corresponds to the ones in J2ME.

J2ME MIDP	.NET CF
Form	Form
Canvas	Form
List	ListBox within Form
Alert	Label within Form
TextField	TextBox with Button
ChoiceGroup(single choice mode)	RadioButton and Label within Panel
ChoiceGroup(multiple choice mode)	CheckBox and Label within Panel
Command	MenuItem within MainMenu

Table 3-4: User Interface Composition in .NET CF corresponds to J2ME

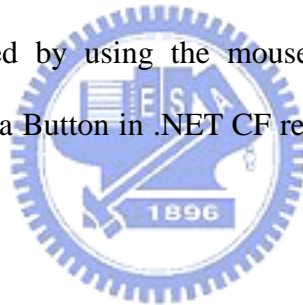
Form class in .NET CF represents a window or dialog box that makes up an application's user interface. Its functions are the same as the Form class in J2ME MIDP,

Canvas class in J2ME MIDP is a base class for writing applications that handle low-level events and issue graphics calls for drawing to the display. In .NET CF, Form class can display graphics in a window within an application. Therefore, it can also be used to replace the J2ME MIDP's Canvas class.

List class in J2ME MIDP contains a list of choices. In .NET CF, ListBox class displays a list of items that users can select by clicking. Therefore, a ListBox within a Form in .NET CF is to replace the List class in J2ME MIDP.

Alert in J2ME MIDP shows data to users and waits for a certain period of time before proceeding to the next screen. .NET Label provides descriptive text to users and is to replace Alert class in J2ME MIDP.

TextField in J2ME MIDP is an editable text component with a button to set text placed into a Form. TextBox in .NET CF allows the user to enter text in an application. A .NET CF focused Button can be clicked by using the mouse, ENTER key, or SPACEBAR. The combination of a TextBox and a Button in .NET CF reaches the same function of a TextField in J2ME MIDP.



ChoiceGroup in J2ME MIDP is a group of selectable elements intended to be placed within a Form. The group can be created with a mode that allows a single choice only or multiple choices. In .NET CF, RadioButton and CheckBox are used for single choice and multiple choices respectively. RadioButton allows users to choose from mutually exclusive options. CheckBox lets users pick a combination of options. A Panel in .NET CF can contain other user interface objects such as RadioButton and CheckBox. Radio buttons in a Panel constitute a group in .NET CF for users to choose at most one choice at a time, like the ChoiceGroup of single choice mode in J2ME MIDP. .NET CF Check boxes in a Panel constitute a group for users to select multiple choices at a time, which is the same as ChoiceGroup of multiple choice mode in J2ME MIDP.

Command in J2ME MIDP can be mapped onto the available physical buttons on a device. Some commands that cannot be mapped onto physical buttons are placed in a menu and the label "Menu" is mapped onto one of the programmable buttons. MainMenu in .NET CF represents a container for the menu structure of a form. It can be added to the menu bar of the form automatically in .NET CF. MenuItem in .NET CF represents an individual item displayed within a MainMenu. J2ME MIDP Command is replaced with MenuItem in .NET CF because these .NET CF components have fixed position on devices. Figure 3-7, 3-8 shows the position of Command and MenuItem for "Start" and "Exit" in an ART client in J2ME MIDP and .NET CF respectively.



Figure 3-7: An ART Client in J2ME



Figure 3-8: An ART Client in .NET CF

In J2ME MIDP, Layout Manager controls the size and position of components in a container. However, .NET CF has no Layout Manager to control the size and position of components in a container. The size and position of components must be set by developers in .NET, which should be taken care of in translating user interface components from J2ME

to .NET CF.

3.4. Event Handling

J2ME and .NET CF has the same event model [9] [10], as shown in figure 3-9. An event source generates events, which is usually a component of user interface. A listener is an object interested in some specific events. The listener subscribes the event with interest to an event source. An event is propagated from an event source object to a listener object by invoking a method on the listener.

During porting an ART client from J2ME to .NET CF, there are two differences between them in event handling, the process of event subscription and the handling of paint event.



3.4.1. Event Subscription

J2ME has defined the interface of listeners for event sources. A class implements this specific interface is considered to subscribe the event generated by the corresponding event source. In .NET CF, the listener is defined by developers, and an event is subscribed by a user-defined method of the listener. For example, figure 3-10, and 3-11 represents how to subscribe an event in J2ME and .NET CF respectively. In figure 3-10, the event source is `okCommand` and `CommandListener` is the listener interface of `Command` class. `SystemMenu` class subscribes to `okCommand` by implementing `CommandListener`. `SystemMenu` overwrites `commandAction`, an event handler interface of `Command` class, to process the event generated by `okCommand`. The `commandAction` method processes events generated by all commands defined in `SystemMenu`. In order to differentiate the event sources with each distinct actions, an event source in the `commandAction` method should be checked during processing an event,

as line 4 of figure 3-10.

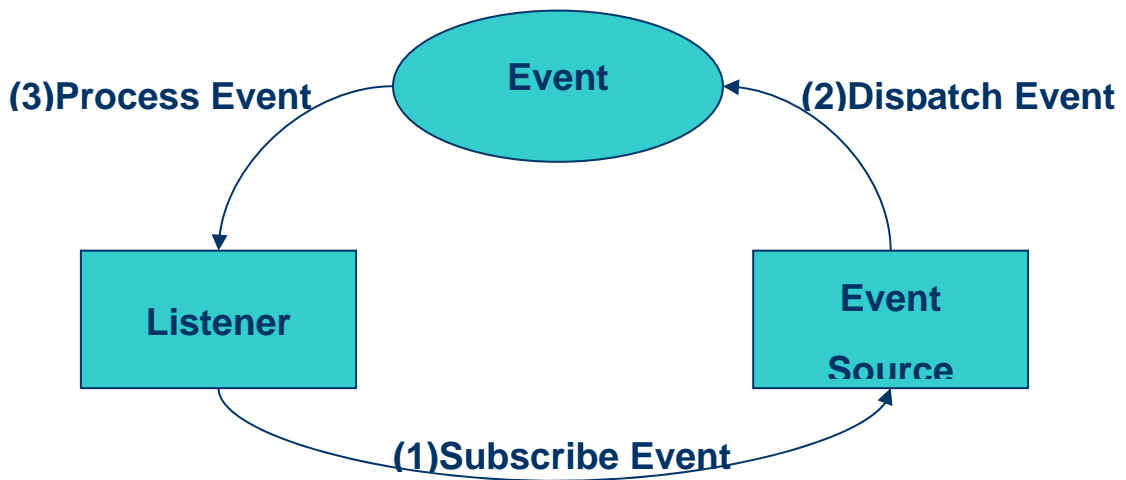


Figure 3-9: Event Model in J2ME and .NET CF

```
1 public class SystemMenu implements CommandListener {
2     private Command okCommand;
3     public void commandAction(Command c, Displayable d) {
4         if (c == okCommand) {
5             .....
6         }
7     }
8 }
```

Figure 3-10: Event Subscription in J2ME

Figure 3-11 shows the code of .NET CF migrated from figure 3-10. The event source is okCommand and the listener is okCommandAction defined by developers. In line 3, okCommandAction subscribes the click event of okCommand. Compared with J2ME, .NET CF event sources with distinct action are differentiated by assigning to different listener methods.

```

1 public class SystemMenu {
2     private MenuItem okCommand;
3     okCommand.Click+=new EventHandler(okCommandAction);
4     public void okCommandAction(object sender, System.EventArgs e){
5         .....
6     }
7 }

```

Figure 3-11: Event Subscription in .NET CF

3.4.2. Paint Event Handling

In J2ME, a paint event handler is executed at J2ME system thread, while in .NET CF, it is executed at the thread who creates the event source of the paint event. The following shows the affection to ART system.



A Canvas object in J2ME shows image data in an ART client. In J2ME, the paint event is generated when a Canvas object is actually visible on the output device. Then, the paint event handler is called to render the Canvas. Figure 3-12 shows the Canvas displaying process of ART client in J2ME. In figure 3-12, “Server” represents the ART server; “MesgHandler” and “System UI thread” are executed in an ART client, the former handles ART messages received from the ART server and the other is J2ME system thread which handles UI events. First, MesgHandler receives an ART message from an ART server to create and show canvas object, then the method of showing canvas, Canvas.Show(), is called and then go back to the state of waiting to get ART messages from the ART server. When Canvas.Show() is called, it triggers the System UI thread to create a paint event of canvas. System UI thread then executes paint event handler. Notice the paint event handler is executed at System UI thread.

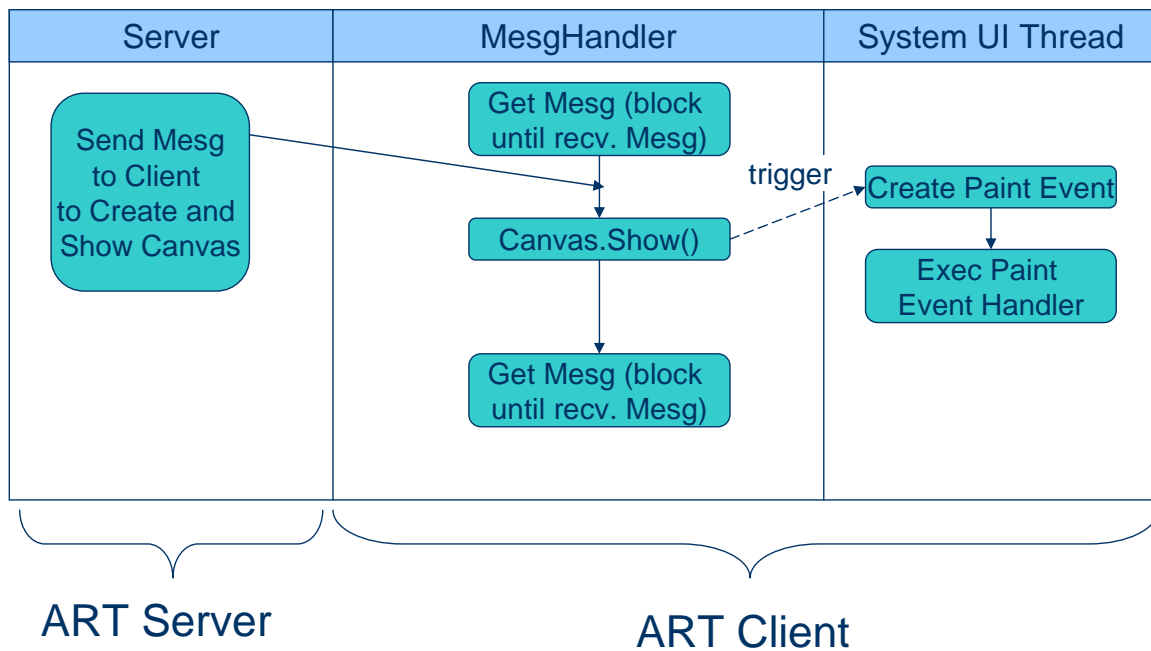


Figure 3-12: Canvas Displaying Process of an ART Client in J2ME

In .NET CF, a paint event occurs when the component of user interface is redrawn. An ART client on .NET CF platform creates a Canvas class inherited from Form class to represent image data and overrides the paint event handler of Form class which triggers .NET system to create a paint event of Form to show image data in Canvas class.

Figure 3-13 shows the Canvas displaying process of an ART client in .NET CF. The difference between J2ME MIDP and .NET CF in paint event handling is that the paint event handler in NET must be executed at MesgHandler thread, because in .NET CF, the paint event handler is executed at the thread who creates the paint event source. In an ART client, Canvas object is the paint event source, which is created by MesgHandler thread. However, MesgHandler is blocked at the state of waiting to get ART messages from the ART server and can not execute the paint event handler. It causes a Canvas object can not be displayed on an ART client.

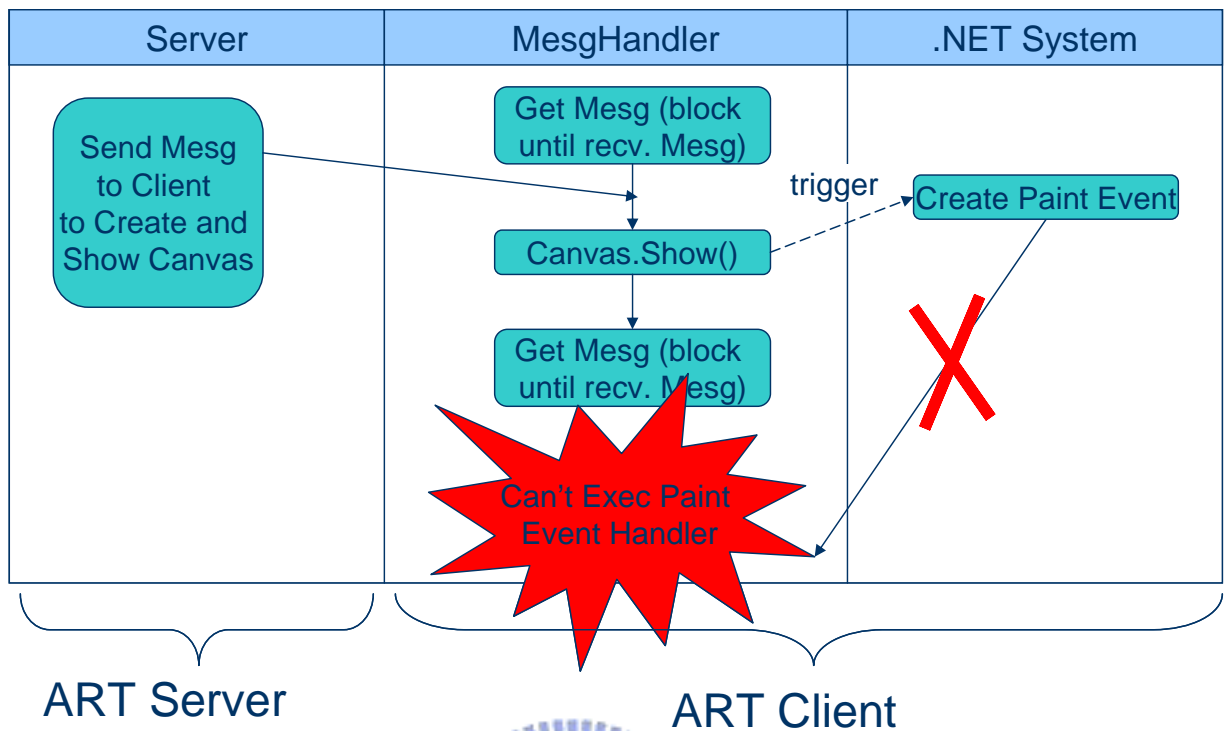


Figure 3-13: Canvas Displaying Process of an ART Client in .NET CF

To resolve the problem mentioned previously, the MesgHandler thread must not be the creator of Canvas object. Therefore, WinMngr thread is generated to create Canvas object and the modified process of Canvas displaying as shown in figure 3-14. The Invoke() method is called in MesgHandler thread to create a Canvas object in WinMngr thread. The usage of the Invoke() method in MesgHandler thread is as follows, which means to execute wm.mesgHandler method in the thread who owns the wm object:

```
wm.Invoke(new EventHandler(wm.mesgHandler));
```

The wm.mesgHandler() method creates a Canvas object. The wm object is created in WinMngr thread, which is an instance of Control class which defines components with visual representation. In .NET CF, the Invoke() method of a Control object executes an event

Chapter 4 GPRS Latency Improving

An ART client must communicate with an ART server through network. GPRS is the most popular facility to access network among many mobile devices. However, GPRS latency is very high. It takes 600ms-3000ms for the downlink, 400ms-1300ms on the uplink [11], and 1000ms or more on round-trip latencies [11]. There are many factors influencing GPRS latency. For ART, data size and the Nagle algorithm are main factors that affect the GPRS latency. To decrease the GPRS latency in ART system, this research executes many experiments to evaluate the effect of data size and the Nagle algorithm. According to these experiments, this research also represents two algorithms to improve the GPRS latency in ART system.

4.1. Test Bed



Figure 4-1 shows the test environment of this research for GPRS latency. The laptop communicates with the PC through the GPRS and Ethernet network. There are a client running on the laptop and a server running on the PC. Besides, the laptop connects to the Sony Ericsson P900 cell phone through infrared rays to act as a GPRS mobile terminal. According to GSM and GPRS architecture, the BSs (base stations) are connected to the BSC (base station controller) linked to the SGSN (Serving GPRS Support Node), and then the SGSN is linked to the GGSN (Gateway GPRS Support Node) to connect to the Ethernet. The measurements presented in the paper were all performed over Chunghwa Telecom GPRS network.

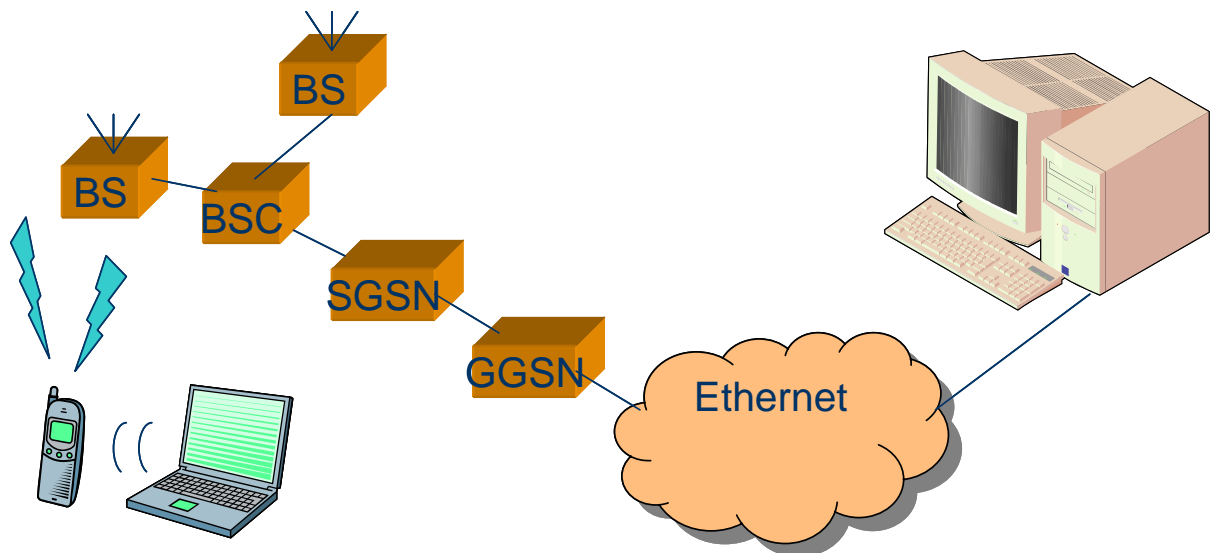


Figure 4-1: Test Bed for GPRS latency

4.2. ART Applications

Many kinds of applications can be executed on ART system. The following three ART applications with different characteristics were chosen for test:

- Pushpuzzle – It is a game that interacts with a user. Initially, it shows a graph on an ART client. When a user presses a button on a device to play the game, the ART client generates an ART message with the user action and sends to the ART server. The ART server executes this application according to the ART message and sends the result as response back to the ART client which shows the result on the device. A user can repeat the action mentioned above to interact with this application.
- Chat – It is a chat room application that has lots of UIs (user interface) to show in the ART client. All UIs are generated during start-up. Therefore, the ART server sends amount of ART messages to the ART client to start up Chat. Each ART message contains one UI generation command. Users can create new group or join an exist group to talk to each other. A user types words at an editable text field

within the “say” window. The ART client sends the ART message with user input content to the ART server that retransmits this ART message to other ART clients in the same chat group.

- RemoteDesk – It is an application for displaying the screen and controlling the mouse position of the ART server from the ART client. ART messages associated with this application usually have larger data size. A user presses an arrow button on device to change the mouse position of the ART server which sends its screen of current mouse position as response to the ART client.

The differences amount these applications are as shown in table 4-1.

	Characteristics	Average Size of ART Message	Number of ART Messages when Initial
Pushpuzzle	Interaction with User	272 bytes	7
Chat	Large amount of ART messages	105 bytes	24
RemoteDesk	ART message with large data	11103 bytes	7

Table 4-1: The Differences amount ART applications

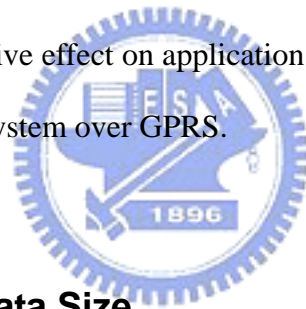
4.3. GPRS Latency Factors in ART

ART messages are transferred through socket connections over TCP. To improve GPRS latency of ART system, this paper evaluates two factors as follows:

- Data Size –ART message size is usually small, and its average size is about 300 bytes. Too small a data may cause inefficiency (i.e. there may be little data compared to lots of header). When small data packets are sent on TCP, TCP congestion may occur because of frequently acknowledgements. Larger data is therefore more

efficient. In some research, the packet size on TCP affects the network latency [12]. This research experiments on data size to get the optimal value.

- The Nagle Algorithm [13] – It was designed to reduce LAN and other network congestion from TCP applications. The Nagle algorithm works by aggregating data on the sending side of TCP applications. It accumulates sequences of small messages into larger TCP packets before data reaches the wire, and thereby prevents the generation of unnecessary large numbers of small packets. When the Nagle algorithm works as designed, TCP applications utilize network resources more efficiently. Applications can enable or disable the Nagle algorithm with the TCP_NODELAY socket option. All of Windows, Linux, and Java systems all normally enable the Nagle algorithm by default. However, in some cases, the Nagle algorithm has a negative effect on application performance. This paper experimented on its effect in ART system over GPRS.



4.3.1. Test Result of Data Size

This paper designed a test plan as follows to evaluate the effect of data size: The client sent a packet to the server, which echoed with this packet to the client. The GPRS latency was measured at the application layer in the client. Its value was the time during a client sent a packet and got the echoed packet. The range of packet size varied from 50 to 2000 bytes at 50-byte increment. Each packet was sent to the server one hundred times. The client sent the next packet until it receives the previous packet echoed from server. The test has been repeated four times. Figure 4-2 shows its result. Each line represents the complete test mentioned above. Each point represents the average round-trip GPRS latency of a particular packet size. The GPRS latency increases with packet size increasing from 50 to 1450 bytes, but suddenly decreases at size 1500 bytes.

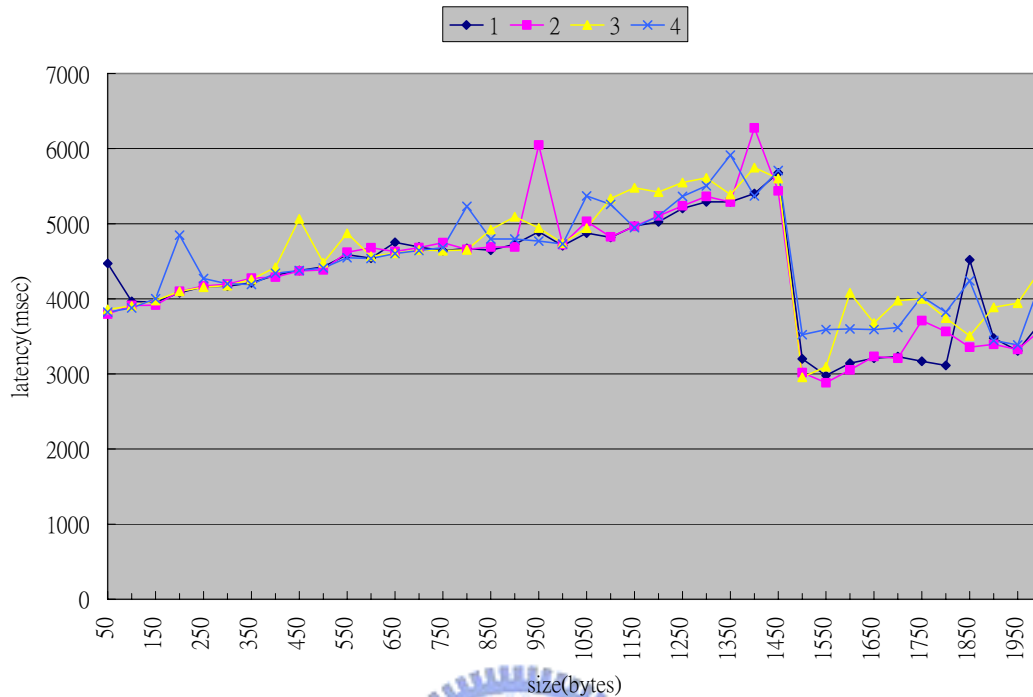
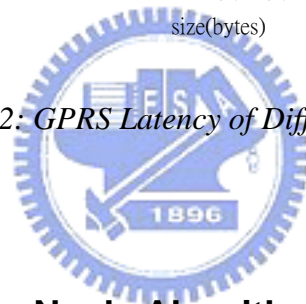


Figure 4-2: GPRS Latency of Different Packet Size



4.3.2. Test Result of the Nagle Algorithm

The Nagle algorithm is enabled by default on Windows, Linux, and Java systems. This test set the TCP_NODELAY socket option to zero at an ART server to disable the Nagle algorithm. The Nagle algorithm was enabled at an ART client as result of the fact that a J2ME MIDP version ART client did not have the TCP_NODELAY socket option to enable or disable the Nagle algorithm. Those ART applications mentioned at section 4.2 were used for this test.

Figure 4-3 shows the effect of the Nagle algorithm for GPRS latency in ART system. The value of GPRS latency was measured at the ART client. The x axis stands for the

operations of ART applications in the ART client. The number behind an operation is the amount of ART messages need in this operation. Line of “Original ART” represents the GPRS latency with the Nagle algorithm being enabled. The other line represents the GPRS latency with the algorithm being disabled. Clearly, disabling the Nagle algorithm makes the lower GPRS latency in most operations, but this is not always the case. The reason is that the ART client usually waits for the response of the ART server to show operation result to users, but the Nagle algorithm does not send the data to network immediately. For example, in Pushpuzzle, when a user presses “down” button on the device, the ART client sends this user action as a request to the ART server and waits for the response, but the response of ART server does not send to the ART client immediately with the Nagle algorithm. In such cases, disabling the Nagle algorithm makes lower GPRS latency.

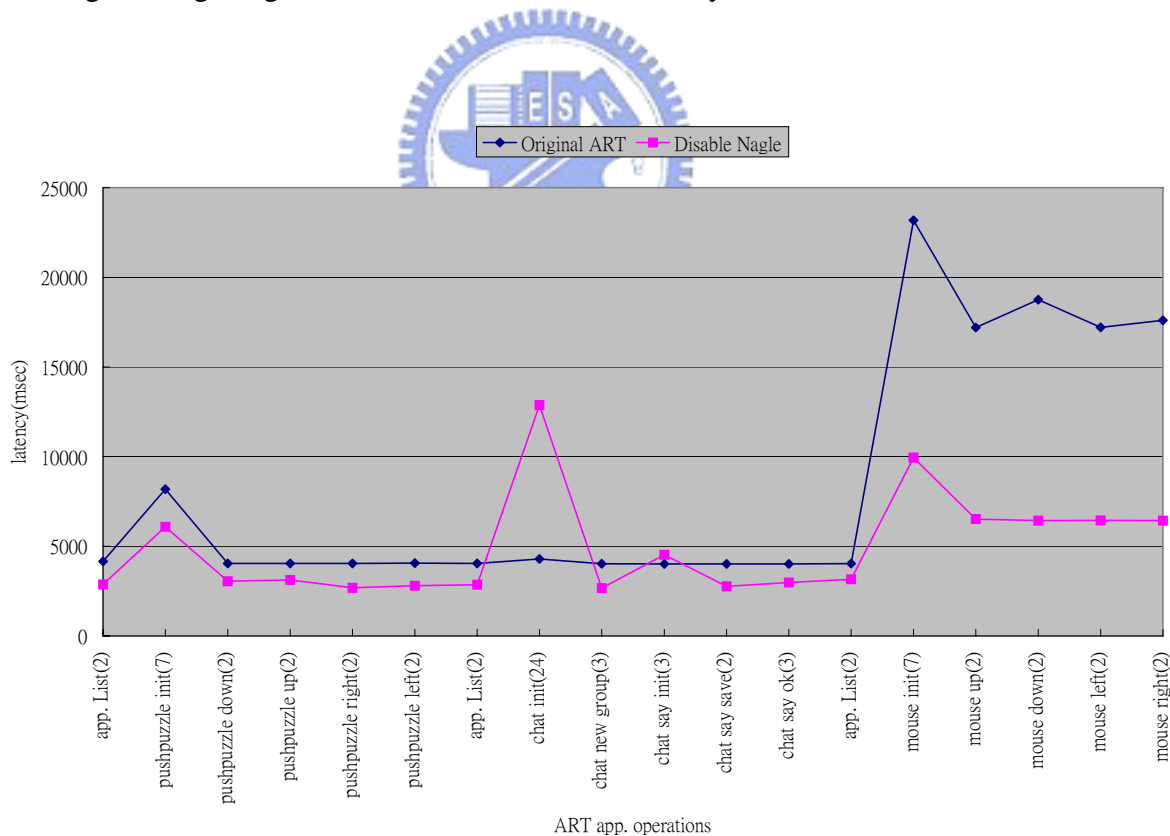


Figure 4-3: The effect of the Nagle algorithm for GPRS latency in ART system

Contrarily, the “chat init” operation has higher GPRS latency with disabling the Nagle

algorithm. The reason is that the ART server sends out a lot of ART messages (the exactly number of ART messages is 24) once for the ART client initializing the Chat. Under this circumstance, the Nagle algorithm helps aggregate ART messages to a larger packet which can be sent more efficiently. The other operations only need to receive one or two ART messages for fulfilling their tasks.

4.4. Algorithms to Improve GPRS Latency in ART system

Consequently, based on the previous test result, two conclusions were made as follows:

- Data size of 1500 bytes has the lower GPRS latency.
- Disabling the Nagle algorithm can reduce the GPRS latency when an operation has little ART messages to deliver. However, it also increases the GPRS latency if an operation has a large amount of ART messages to deliver.

According to these two conclusions, this research comes up with two following algorithms to improve GPRS latency by disabling the Nagle algorithm and aggregate ART messages at a period of time in the ART server.

4.4.1. Algorithm 1

In a period of time T , an ART server aggregates ART messages which will be sent to an ART client. When the ART server receives a request ART message from the ART client, a delay timer of value T begins to count down. When the delay timer expires, the ART messages are concatenated to a larger data and sent to the ART client immediately. If the total accumulated size of ART messages exceeds more than 1500 bytes before the delay timer expires, they are also sent to the ART client immediately.

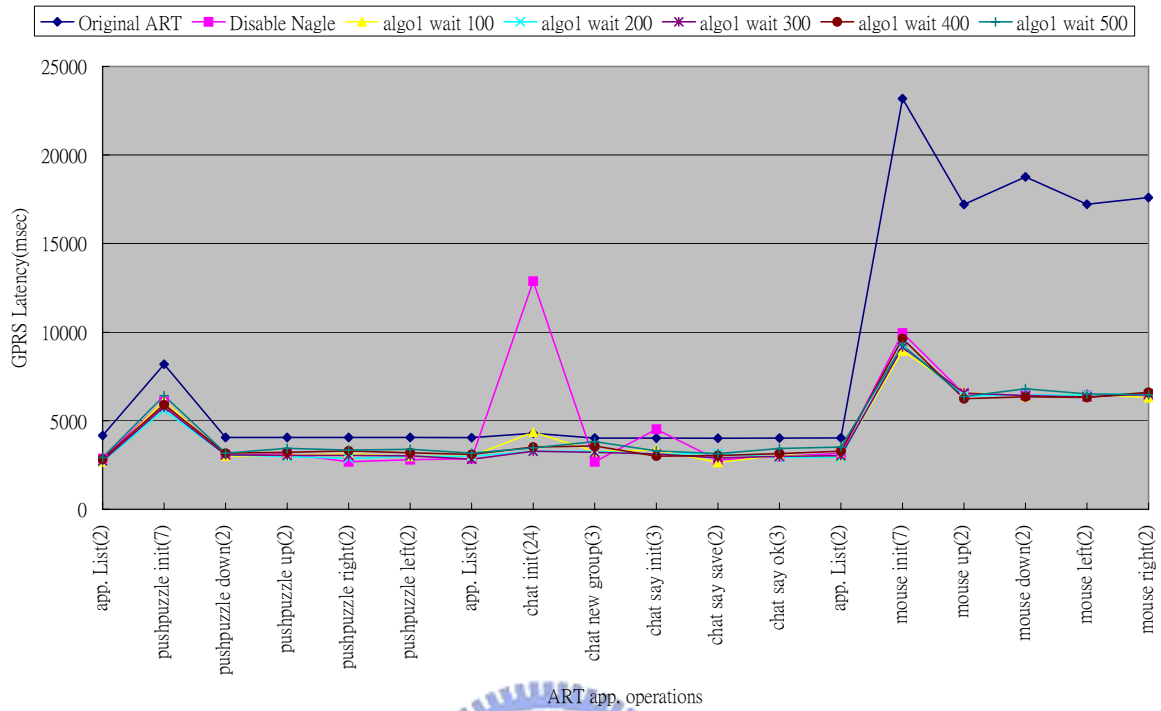


Figure 4-4: GPRS latency of Algorithm 1 with Different Waiting Time in ART system

	Avg. Latency(sec)	Improvement (%)
Original ART	8.388	--
Disable Nagle	4.904	41.53%
Wait 100ms	4.355	48.09%
Wait 200 ms	4.276	49.02%
Wait 300 ms	4.282	48.95%
Wait 400 ms	4.414	47.38%
Wait 500 ms	4.555	45.69%

Table 4-2: Improvement of Algorithm 1 for GPRS Latency

Figure 4-4 shows the GPRS latency evaluation of algorithm 1 with different waiting time T. Table 4-2 represents the average latency and improving percentage from figure 4-4.

Waiting time 200 ms causes the lower GPRS latency in most ART application operations. Compared with the original ART system, algorithm 1 decreases the GPRS latency about 45%.

4.4.2. Algorithm 2

Different operations of ART applications use different fields of ART messages. Some fields are not used for specific ART application operations. Useless fields are taken off from ART messages for each ART application operation. For some ART messages sent to an ART client at the same time, the fields of ART messages with the same values can be merged to minimize the ART messages.

In a period of time T , an ART server merges ART messages by the field with the same value. When the ART server receives a request ART message from the ART client, a delay timer of value T begins to count down. When the delay timer expired, the ART messages are merged to a larger data and sent to the ART client immediately.

Figure 4-5 represents GPRS latency of algorithm 2 with different waiting time in ART system. Table 4-3 shows the average latency and improving percentage from figure 4-5. Waiting time 200 ms causes the lower GPRS latency in most ART application operations. Compared with the original ART system, algorithm 2 decreases the GPRS latency about 45%.

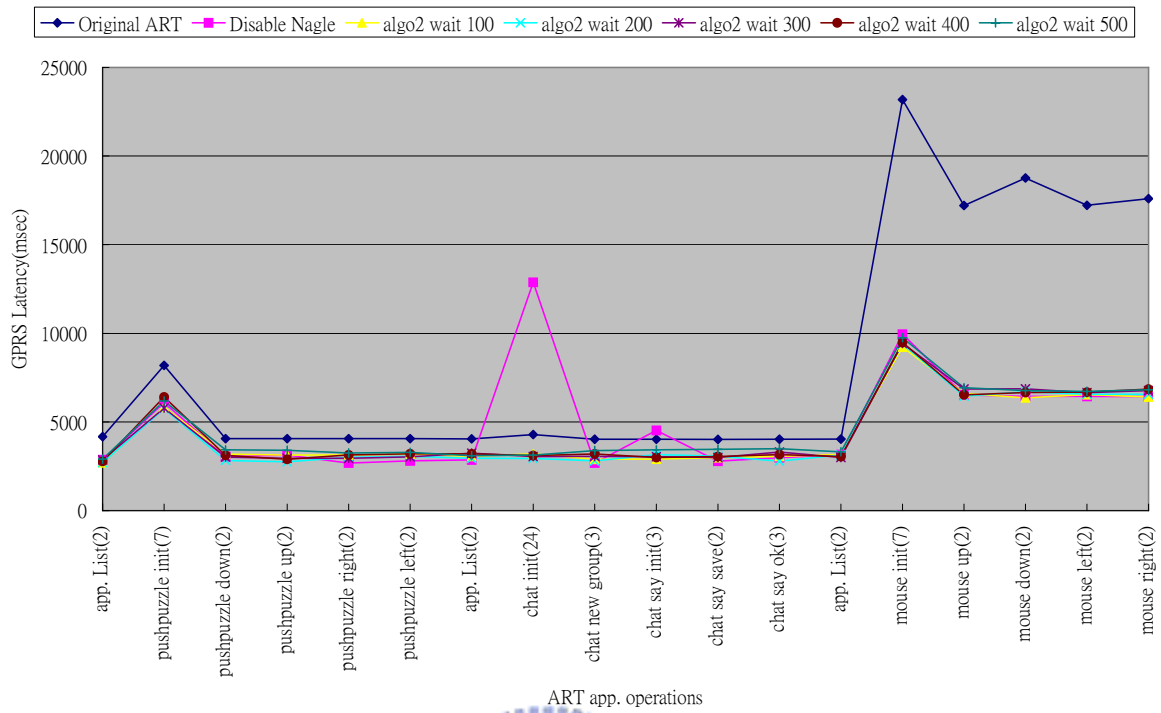


Figure 4-5: GPRS latency of Algorithm 2 with Different Waiting Time in ART system

	Avg. Latency(sec)	Improvement (%)
Original ART	8.388	--
No wait	4.904	41.53%
Wait 100ms	4.308	48.64%
Wait 200 ms	4.256	49.26%
Wait 300 ms	4.380	47.78%
Wait 400 ms	4.414	47.37%
Wait 500 ms	4.523	45.23%

Table 4-3: Algorithm 2 for GPRS Latency Improvement

	Avg. Latency(sec)	Saving bytes
Original ART	97200	--
Wait 100ms	95316	1884(1.94%)
Wait 200 ms	95224.75	1975.25(2.03%)
Wait 300 ms	95225.625	1974.375(2.03%)
Wait 400 ms	95191.75	2008.25(2.07%)
Wait 500 ms	95182	2018(2.08%)

Table 4-4: Algorithm 2 for GPRS Bandwidth Saving



Chapter 5 Conclusions and Future Works

5.1. Conclusions

This paper represents four porting issues from J2ME to .NET CF and two algorithms to improve the GPRS latency. The porting issues include network communication, ART message handling, user interface display and user interaction handling.

In network communication, the methods of establishing a socket connection in J2ME and .NET CF are the same. BinaryWriter and BinaryReader classes in .NET CF are chosen to replace DataOutputStream and DataInputStream in J2ME respectively.

In ART message handling, numeric and string data in ART message are transferred to byte first and then transmitted by a socket connection. The value of numeric data has byte order issue on different executed platforms. The byte order on Java platform is big-endian, and so as in socket. Bytes read from a socket can transfer to numeric data directly on Java platform. However, the byte order on .NET depends on hosts. This paper uses `IPAddress.NetworkToHostOrder()` method to transform byte from network order to host order. For string data in ART messages, both of Java and .NET provide many methods of converting arrays and strings of Unicode characters to and from arrays of bytes encoded for another character set. The string data in an ART message should be encoded by the character set with the same definition.

In user interface display, the user interface composition in .NET CF corresponding to J2ME was discussed. The .NET CF does not have a layout manager which is provided by

J2ME. User interface component in .NET CF should be arranged additionally.

In user interaction handling, J2ME and .NET has the same event model. The paint event in J2ME is handled by system UI thread, however, in .NET CF, the paint event is handled by the owner thread of event sources. For an ART client, the MesgHandler is the owner thread of canvas which brings paint events. In .NET, the MesgHandler thread can not execute the paint event handler. Therefore, the WinMngr thread on ART client .NET platform is added as the owner of canvas object for handling paint events. The MesgHandler thread uses the Invoke() method to create a canvas object in WinMngr thread.

To reduce the GPRS latency, this research discusses about the effect of data size and the Nagle algorithm at first. According to problems found in the evaluation, this paper provides two algorithms to decrease the GPRS latency. These two algorithms suggest an ART server disable the Nagle algorithm and send ART messages to an ART client either when ART messages are aggregated more than 1500 bytes or when the delay timer T is expired. The test result shows the delay timer T with 200 milliseconds causes the lower GPRS latency in most ART application operations.

5.2. Future Works

There are several different directions to explore in future development of ART. ART was aimed at personal use at first even though an ART Server can serve various users simultaneously. If ART is expected to be accepted and deployed widely, it must be more mature. Therefore, a few future works that enhance ART still need to be done.

- *Security*

The checks on users' identifications by CommMngr are very crude. This paper has made only few attempts on the security which is not the issue expected to be solved in this research. Hence ART will be probably exposed to malicious parties if ART Servers are attached to Internet. ART must be able to secure itself in the future.

In order to integrate with available and existent technologies another important work will be adopting AAA (Authentication, Authorization and Accounting) technology that is maintained by the Authentication, Authorization and Accounting Working Group of IETF and applied in mobile computing area popularly.

- *Toward Web Service*

An ART app. may access resources or service of another third party, but it is not constructed the inverse way. A non-ART program is not able to know where or how to get ART services (ART App.).

In terms of scalability and trend, ART should provide an approach compatible with W3C Web service. Web services provide a standard means of interoperating between different software applications which run on a variety of platforms and/or frameworks. Much attention has been paid to introduce mobile computing to Web Service.

Because of the following reasons, supporting Web service will be a good choice.

1. Web service is a very popular and open standard.
2. ART supports HTTP protocol and handles much XML-based actions (e.g. converting functions).
3. Java platform with which ART server is built already supports Web service.
4. With Web service, ART may be close to enterprise use.

- *Hardware implementation*

Besides ART server is installed on desktop computers, system providers are

expected to support ART server on some embedded systems. Embedded systems have several common characteristics, which distinguish such systems from other computing systems such as specific-functioned, tightly constrained, reactive and real time. These characteristics are also suitable for the server side environment, too.

System providers might not want to supply any service but ART, and the performance and the real time user experience are what they only concern. Under this situation, a general purpose computer (i.e. PC) is not a good choice. Therefore, it is suggested that system providers support ART server on embedded systems as well.



References

- [1] 姚立三, 袁賢銘, "An Adaptive Mobile Application Development Framework", 國立交通大學電資學院學程碩士班論文, 民國 92 年 6 月
- [2] Sun Microsystems, "Java 2 Platform, Micro Edition (J2ME)",
<http://java.sun.com/j2me/>
- [3] Microsoft, ".NET Compact Framework (.NET CF)"
<http://msdn.microsoft.com/smartclient/understanding/netcf/>
- [4] Microsoft, "The C# language" <http://msdn.microsoft.com/vcsharp/programming/language/>
- [5] 3GPP, "23.060 GPRS; Service description"
<http://www.3gpp.org/ftp/Specs/html-info/23060.htm>
- [6] Hye-Sun Hur¹ and Youn-Sik Hong, "Performance Analysis of Multimedia Data Transmission with PDA over an Infrastructure Network" ICCSA (3) 2004: 1002-1009
- [7] Sun Microsystems, "CLDC", <http://java.sun.com/products/cldc/>
- [8] Sun Microsystems, "Mobile Information Device Profile (MIDP)",
<http://java.sun.com/products/cldc/>
- [9] Sun Microsystems, "Java AWT: Delegation Event Model",
<http://java.sun.com/j2se/1.3/docs/guide/awt/designspec/events.html>
- [10] Microsoft, "Event Handling in Windows Forms",
<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vbcon/html/vbconeventhandling.asp>
- [11] Rajiv Chakravorty, Joel Cartwright, Ian Pratt, "Practical Experience with TCP over GPRS" Proceedings IEEE GLOBECOM 2002, November 2002, Taipei, Taiwan.

- [12] Calveras, A., Paradells, J.; Gomez, C.; Catalan, M.; Valles, J.C. “Optimizing TCP parameters over GPRS and WLAN real networks”, Communications, Computers and signal Processing, 2003. PACRIM. 2003 IEEE Pacific Rim Conference on, Volume: 2, 28-30 Aug. 2003 Pages: 663 - 666 vol.2
- [13] Nagle, J., “Congestion Control in IP/TCP Internetworks” RFC 896, Ford Aerospace and Communications Corporation, January 1984.

