

Bug Fix of Mobile Terminal Software using Download OTA

† Masato Takeichi , Atsushi Hosokawa , Kazunori Nasu
‡ Seiji Hoshi , Koichi Moriyama , Tadao Takami , Kazuaki Terunuma

NTT DoCoMo, Inc.
† Multimedia Development Department
‡ Consumer Equipment Development Department
3-5 Hikarinooka, Yokosuka-shi, Kanagawa 239-8536 Japan
E-mail: † {takeichi, hosokawa, nasuk} @nttdocomo.co.jp
E-mail: ‡ {hoshis, moriyamak, takami, terunuma} @nttdocomo.co.jp

Abstract:

With the growing size of software for mobile terminals as a result of the ever growing number of functions, it has become increasingly difficult in recent years to develop bug-free software. It has therefore become more desirable to let users download software for their mobile terminals over-the-air (OTA) namely through wireless communications when bug fixes become necessary. This paper first identifies issues regarding the number of connections to the server and the size of data for fixing bugs that must be addressed in order to make possible a mobile terminal software update system that utilizes wireless communications. Firstly, we discuss proposals to resolve these issues from the viewpoint of the mobile terminals, network, and server. Then, this paper proposes two methods. The one is a transfer control method that a server permits mobile terminals to download the data only when it does not affect the network depending on allowable transfer capacity which is set to the server. The other is a reservation method that a server allows users to reserve the convenient time for the download to occur. Finally, the paper also describes the overall configuration of the system and the interface protocol between mobile terminal and server, and then presents evaluation results of mobile terminal simulator and server trials based on these methods.

Keywords: download, mobile communications, mobile terminal software, bug-fix

Contact Person: Masato Takeichi

Introduction



Background:

Size of software is growing due to increases in mobile terminal functions

⇒ Increase in the possibility of bugs' being inherent.

Present:

To replace mobile terminal or rewrite the software at retail store:

An environment in which users themselves can fix bugs in their mobile terminals rather than taking them to retail stores needs to be established.



Realize to update mobile terminal software OTA.

Improve user convenience

Reduce the costs to replace mobile terminal

Introduction

Recently, with the growing size of software for mobile terminals as a result of the ever growing number of functions, it is becoming increasingly difficult to develop bug-free software.

At present, software bugs are being fixed at retail stores where users need to bring their terminals whenever a bug is found. It is therefore necessary to create an environment in which users themselves can fix the bugs in their terminals.

Against this background, it would be desirable for all mobile terminals to have as part of its basic functions the ability to fix bugs by downloading software over-the-air namely using wireless communications network [1]. This approach would significantly improve user convenience, and mobile terminal manufacturers and communications operators can benefit as they would reduce the cost to recall and replace mobile terminals.

This paper identifies the issues inherent in realizing a mobile terminal software update system that utilizes wireless communications and discusses the issues from the viewpoints of mobile terminals, networks, and servers. It then proposes methods to solve the issues and illustrates a system design. This paper also presents evaluation results from trials based on these methods and describes future prospects.

Prerequisites



1. Enable users to update their software via easy operation and within a reasonable length of time.
2. Ensure that the impact on the mobile communications network does not exceed the allowable range.
3. Minimize the impact on the cost of mobile terminals.
4. Do not build in any reliance on the mobile terminal architecture.

Focusing mainly
on conditions 1 and 2

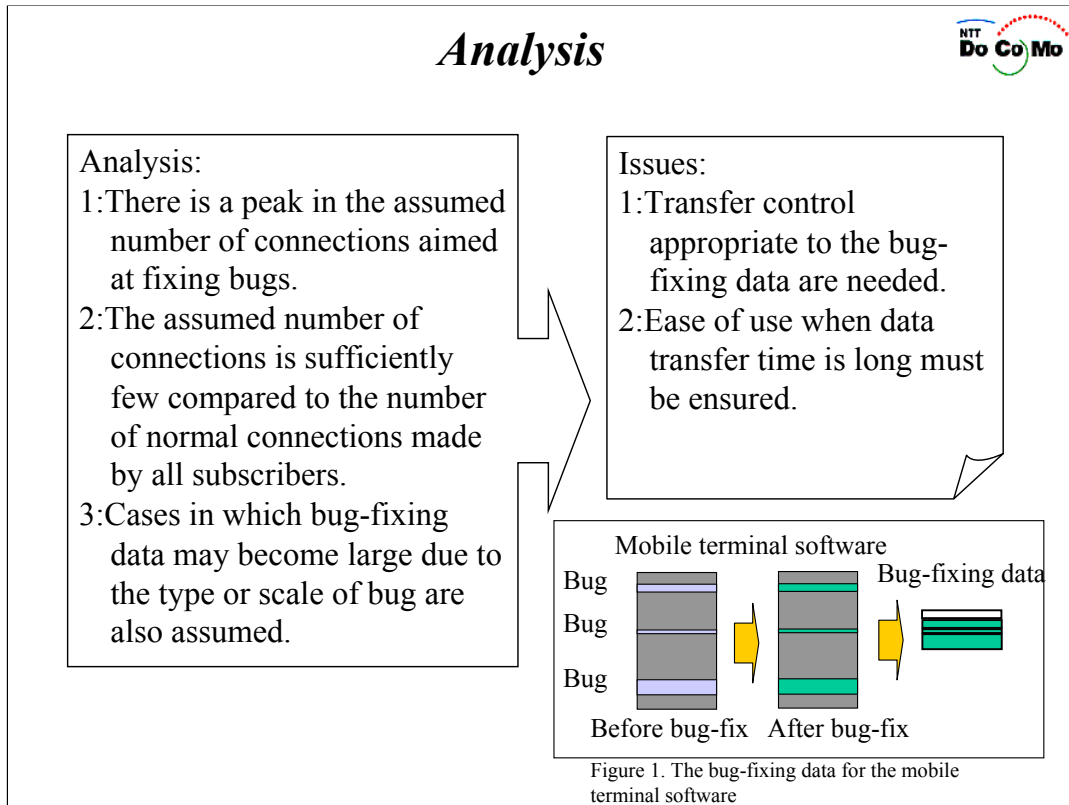
Number of connections to the server and data size to be downloaded must be analyzed.

Prerequisites

In developing a mobile terminal software update system that utilizes wireless communications, a fundamental design principle of actively using existing technology and facility was established. The following prerequisites were also established:

1. Enable users to update their software via easy operation and within a reasonable length of time.
2. Ensure that the impact on the mobile communications network does not exceed the allowable range.
3. Minimize the impact on the cost of mobile terminals.
4. Do not build in any reliance on the mobile terminal architecture.

In order to propose the method to realize which is satisfied with the fundamental design principle and prerequisites that are stated above, this paper has analyzed the number of connections to the server and the size of data focusing on the features of the traffic model.



Number of connections assumed for the purpose of fixing bugs

The number of connections per unit of time from mobile terminals to a server is difficult to predict, since this is dependent on factors such as the number of mobile terminals that need to be fixed and the method of notification. However, software bugs need to be fixed as quickly as possible, and we have assumed the number of wireless connections to the software update system for the purpose of fixing bugs from the actual record of past attempts to replace mobile terminal at retail stores when bugs have been found.

In this assumption, the number of connections peaked during the first days following notification to users that this system could be used to fix a bug. Compared to the normal number of connections from all subscribers, however, the number of connections from mobile terminals requiring bug fixes can be assumed to be sufficiently few.

Maintaining appropriate sizes for bug-fixing data

Since transfer speeds over wireless connections are slower than wired broadband connections, these speeds are insufficient for transferring mobile terminal software as is. It is therefore necessary to keep the size of data used for fixing bugs as small as possible.

Algorithms that creates differences between software before and after it is modified are proposed [2][3]. It is possible to apply these algorithms in creating the data that is to be used to fix bugs in the mobile terminal software. (Figure 1) From the viewpoint of Prerequisites No. 4 , it is not necessary to define a tool or format for creating the data for fixing bugs. But from the viewpoints of Prerequisites No. 1 and No. 2, it is necessary to define the maximum size of the data. On the other hand, while the volume of data transfer (transfer or holding time) per mobile terminal depends on the type of bug and its scale, even if the data for fixing the bug was kept to a reasonable size, it can be assumed that the time will be longer than it would be when performing tasks such as reading content in a browser.

As seen above, transfer control appropriate to the data for fixing bugs and the convenience to users when data transfer time grows longer need to be required.

Access Control Methods



Proposal 1:

A transfer control method where data transfer to mobile terminals is allowed only within a range that does not affect the network.

Proposal 2:

A reservation method of software update to enhance convenience to users.

Access Control Methods

In this section, we propose access control methods that resolves issues that have become clear from the results of our analysis.

The controlling factor to ensure that the impact on the mobile communications network does not exceed the allowable range should not be the number of connections but the volume of data being transferred. As a result of this study, we set the volume of data that can be transferred and proposed a transfer control method that permitted data transfer to occur only within a range that had no impact on the network. We also considered the impact on convenience to users in the event transfer times grew longer in accordance with the type and scale of bug to be fixed, and we propose a reservation method on the updating of software. Hereafter, these two proposed methods shall be generally referred to as the access control methods.

Proposal1: The server's transfer control depending on Allowable Transfer Capacity(1/2)



Transfer control appropriate to the bug-fixing data is needed.

➤ Server control data transfer after the first connection request from the mobile terminal is received.

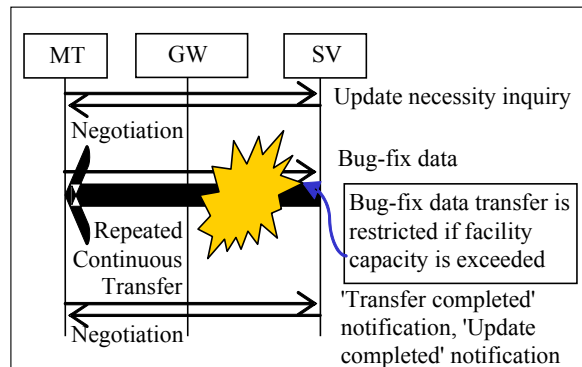


Figure 2: Access control methods

Proposal1: The server's transfer control depending on Allowable Transfer Capacity(1/2)

In this slide, we describe a method of realizing a control protocol used by the server to transfer data to mobile terminals as given in Proposal 1.

As stated above, the controlling factor to ensure that the impact on the mobile communications network does not exceed the allowable range should not be the number of connections but the volume of data being transferred. Accordingly, we are proposing a method of controlling the volume of data transfer at the server only, without controlling congestion related to the bug-fixing function, using network elements (NE) such as the packet gateway (GW) that configure the network. To that end, we are focusing on a sequence that is a feature in software updating.

After the telecommunications operator uses some methods to notify users of the need to update their mobile terminal software, users operate their mobile terminals to update the software. At this time, the mobile terminal sends the name of the terminal's manufacturer, model name, and version of software to the server. Once the server receives this information, it checks whether the terminal's software needs to be updated or not. Bug-fixing data is not sent to terminals if their software does not need to be updated. In this sequence, if there is also a need to suppress the volume of transfer data, when the mobile terminal makes its first connection request (inquiring whether an update is necessary), it is possible for the server to judge whether any further connection is allowed or not, i.e. whether data transfer is allowed, based on the reserved allowable transfer capacity. (Figure 2)

Proposal1: The server's transfer control depending on Allowable Transfer Capacity(2/2)



- Using allowable data transfer capacity on the existing mobile communications network as the criterion for allowing or denying a connection

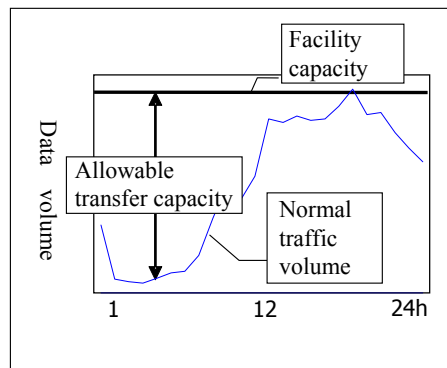


Figure 3: Allowable transfer capacity

Proposal1: The server's transfer control depending on Allowable Transfer Capacity(2/2)

After we describe in detail the method of calculating allowable transfer capacity related to the previous slide, we shall summarize data transfer control that depends on this capacity.

It is possible to use the allowable data transfer capacity (available facility capacity) on the existing mobile communications network as the reference to determine whether a connection is possible or not. Facility capacity represents the maximum value of normal traffic volume obtained from the data on the past traffic. Allowable transfer capacity is the difference between facility capacity and the normal traffic volume that varies over time. (Figure 3)

It is possible for the server to know in advance the size of the data that needs to be transferred for each mobile terminal requiring a bug fix. Therefore, by managing the allowable transfer capacity and the number of mobile terminal connections, the server can also take into consideration the size of the bug-fixing data when judging whether it can transfer data to the terminal or not when it makes its first connection request, and the server can control the volume of data to be transferred.

Proposal 2: Update software at a reserved time

Ensure ease of use when data transfer time is long

- Enable download to occur at the time when a user is not using the mobile terminal to update the software.

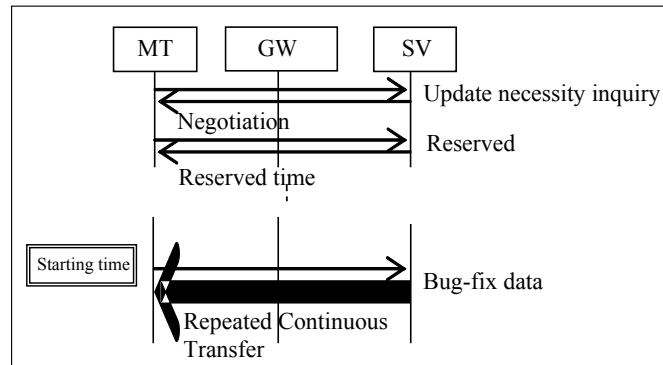


Figure 4: Rewriting software at a preset time

Proposal2: Update software at a reserved time

In this slide, we describe the method given in Proposal 2 in which users are allowed the convenience of reserving the time when their software can be updated.

Much transfer capacity is available at the times---such as the middle of the night---when traffic is quiet and normal traffic volume is small. We propose a software update service at a reserved time, particularly for users who may be inconvenienced by relatively longer transfer times according to the type and scale of bugs.

Users will be able to operate their mobile terminals to let the server know when they wish to have their software update. When the server receives the desired time frame from the user, depending on the allowable transfer capacity and conditions at the desired time, the server will be able to judge whether or not to transfer the data and generate possible times when the transfer could take place. By setting the terminal's internal timer to one of the possible times, the terminal is able to start the download request at the reserved time and update its software. (Figure 4)

In this way, the access control methods that utilize allowable transfer capacity and server transfer control offer good affinity with data transfer judgments during a reserved time frame when users wish to update their software.

System design

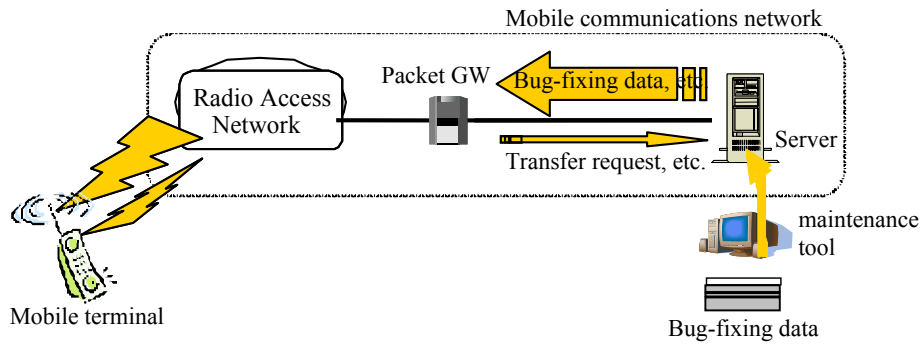


Figure 5: Overall system configuration

System design

This section describes the overall system that incorporates the access control methods to support this function (Figure 5). HTTP is used for the communications between the mobile terminals and the server, and a dedicated maintenance tool is connected to the server in order to operate and maintain the system.

Bug-fixing data is input from the maintenance tool to the server. After sufficient tests for software update are conducted in advance, the user is notified that update service has started. The user operates the mobile terminal to send a connection request to the server (inquiring whether an update is necessary), and the server performs transfer control and manages time reserved based on the access control methods described above. After the mobile terminal downloads the bug-fixing software, the software is rewritten, and the terminal reconnects to the server to notify that the update has been completed.

Based on the fundamental design principle, existing technology and facility is actively used in the design of the system, and the overall system is configured so that it does not impact on the packet GW and other elements.

Implementation of communications protocols and the mobile terminal

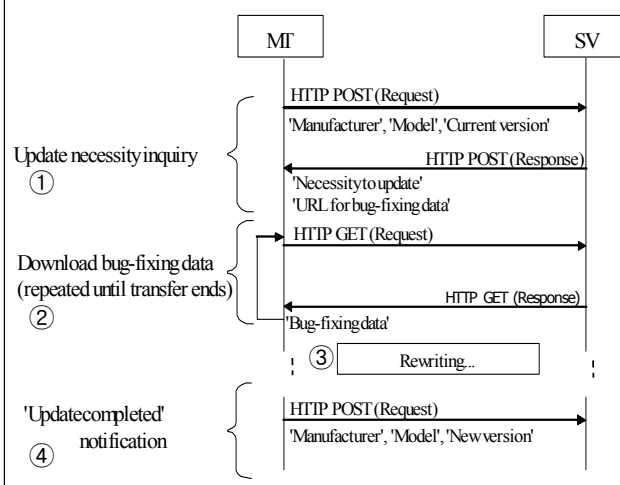


Figure 6: Basic scenario of communications between mobile terminal and server

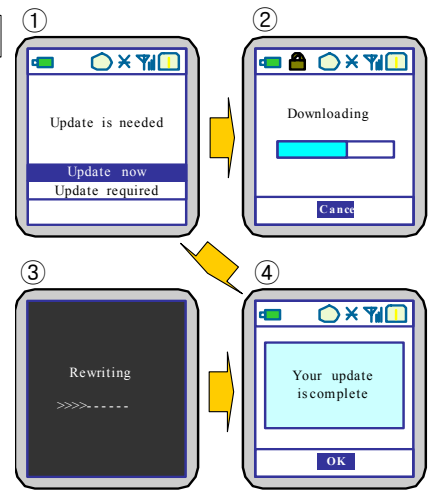


Figure 7: User mobile terminal operations (Example)

Implementation of the interface protocols and the mobile terminal

The HTTP GET method is used for downloading bug-fixing data in the interface protocol between the mobile terminal and the server. This protocol divides the data into practicable sizes which have experience on i-appli etc. The HTTP POST method is used for identifying the bug-fixing data that should be transferred and the reserved transfer time (determining the transfer request start time). Approximately 10 communication commands between mobile terminal and server are designed and implemented as the interface protocol.

Communications with the server complying with the specifications of the mobile terminal (its user interface, capabilities, and functionality) are possible according to the order in which communication commands are issued and how parameters are given. Progress during the downloading and rewriting process is displayed accordingly, offering users ease of use.

When this function is used to download data, the operation of other functions may be restricted to keep temporary storage space for the bug-fixing data. This is a specification designed to minimize impact on the mobile terminal's cost. In addition, reliance on the architecture of various mobile terminals is avoided by not defining the format of the bug-fixing data and using the existing HTTP protocol for communications between the mobile terminals and the server. Further convenience to users is offered by allowing this software update system designed for bug-fixing to be operated together with the other mobile terminal software update system that utilizes wired communication at retail stores / service centers.

In order to identify the bug-fixing data, the mobile terminal sends the name of the terminal's manufacturer, the model name, and version information in the body of an HTTP POST request message to the server. The server responds to the request by checking for necessity to update and identify bug-fixing data from the information on the mobile terminal and the bug-fixing data, finds the URL, and places the results in the body of an HTTP POST response message. (Figure 6)

In accordance with necessity to update indicated in this message, the mobile terminal displays either "Update is needed" or "Update is not needed" and informs the user about what operation to perform next. (Figure 7) If the user selects "Update now," the divided bug-fixing data is transferred via the HTTP protocol from the URL obtained from the server.

Following transfer and rewriting, the mobile terminal sends an "Update completed" message to the server, again in the body of an HTTP POST request message. The server receives the "Update completed" message from the mobile terminal and files the results as a record.

Evaluation of Access Control Methods

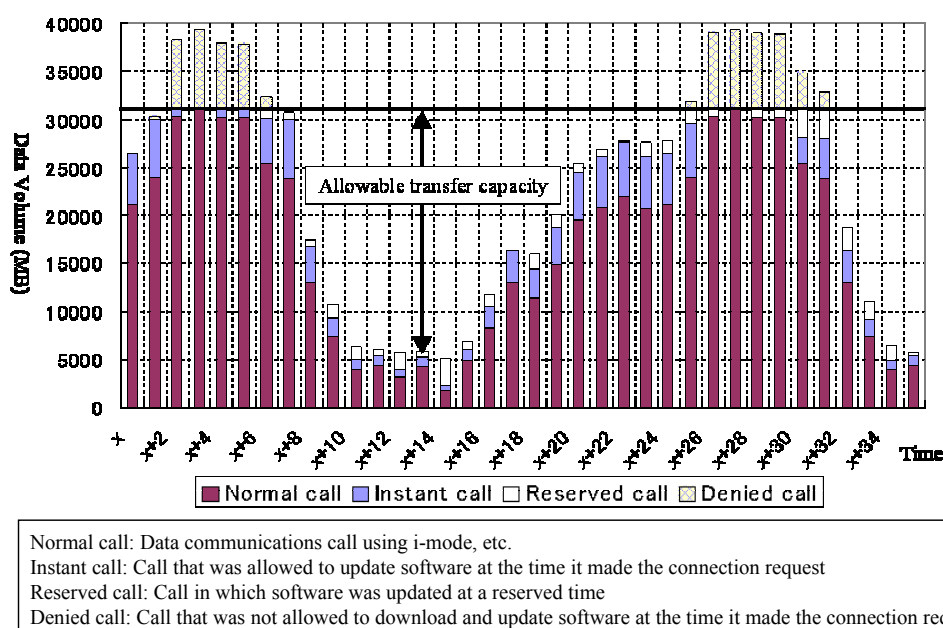


Figure 8: Results of the Simulation of the Access Control Methods

Evaluation of the Access Control Methods

This slide shows the results of evaluative simulation that confirmed the validity of the proposed access control methods.

We established a model in which calls for bug-fixes were added to the volume of normal calls in designated regions based on conditions given below:

- 1) Time distribution for bug-fixing calls was made identical to the time distribution for normal calls.
- 2) When the allowable transfer capacity was exceeded and the call became subject to data transfer control, the time for updating the software was selected at random from candidate times provided by the server and reserved.

The results of simulation on the access control methods conducted according to these conditions are shown in Figure 8. The following observations were confirmed from these simulation:

1. Data volume exceeding the allowable transfer capacity did not enter the network.
2. All denied calls ($x+1 \sim x+6$) which became subject to the data transfer control were completed the software update by reservation using the time frame ($x+5 \sim x+35$) where the traffic was lower.

Conclusion



- ❑ Issues in realizing a mobile terminal software update system that uses wireless communications for the purpose of fixing bugs were identified.
- ❑ The transfer control method based on allowable transfer capacity and transfer permission granted or denied by the server was proposed.
- ❑ A system design proposal that incorporates these methods was illustrated.
- ❑ The validity of the proposed methods based on the simulation results
 1. Data volume exceeding the allowable transfer capacity did not enter the network.
 2. All denied calls which became subject to the data transfer control were completed the software update by reservation using the time frame where the traffic was lower.
- At present, we are receiving feedback on the evaluation results and the principle evaluation results using function-simulation terminals based on general terminals and the server ,then working on development aimed at building the system for practical use.

Conclusion

In this paper, we presented the issues involved in realizing a mobile terminal software update system that utilizes wireless communications as an approach to fixing bugs, and we proposed two methods. The one is a transfer control method that a server permits mobile terminals to download the data only when it does not affect the network depending on allowable transfer capacity which is set to the server. The other is a reservation method that a server allows users to reserve the convenient time for the download to occur. We also illustrated a system design that incorporates the access control methods. The simulation results confirmed that data volume exceeding the allowable transfer capacity did not enter the network. The results also confirmed the validity for all connection attempts subject to data transfer control to make a reservation for a time frame with large allowable transfer capacity and low traffic and to update software at that time.

At present, we are receiving feedback on the evaluation results and the principle evaluation results using function-simulation terminals based on general terminals and the server[4] ,then working on development aimed at building the system for practical use.

Bibliography

- [1] Ministry of Public Management, Home Affairs, and Posts and Telecommunications, Report of the Study Group on Safety and Reliability of 3rd-Generation Mobile Communications Systems, 2001.
- [2] Tridgell, A., and Mackerras, P., "The Rsync Algorithm," Australian National University, TR-CS-96-05, 1996.
- [3] Hunt, J. J., Vo., K., and Tichy, W. F., "Delta Algorithms: An Empirical Analysis," ACM Transactions on Software Engineering and Methodology, 1998.
- [4] S.Hoshi, "Bug Fix of Mobile Terminal Software using Download OTA", IEICE, B-6-208, March, 2003.