UbiNet: A Generic and Ubiquitous Service Provider Framework

Jinsuo (Allan) Zhang
Yahoo! Corp
701 First Avenue
Sunnyvale, CA 94086
azhang@yahoo-inc.com

Sumi Helal
Dept of CISE
University of Florida
Gainesville, FL 32611
Helal@cise.ufl.edu

***Abstract*—In mobile environment, it is very common that mobile devices periodically stay in disconnection mode. In a networked world as today, computer users rely on network services so heavily that even a short duration of disconnection from network will dramatically distract user's normal activity. Past researches explored disconnected operation for file system and database access. In this paper, we introduce and evaluate a ubiquitous network service framework. The proposed framework combines the advantages of both disconnection operation and system virtualization in order to provide 7/24 services to mobile users. We implemented and validated our idea in both Linux and Windows platform. From our initial deployment and daily usage experiment, our architecture is highly feasible and deployable. In this paper, we will illustrate the architecture, implementation, and performance evaluation based on our implemented system.**

Keywords-Disconnected service; virtualization; emulation; mobile computing; Internet computing.

I. INTRODUCTION

Today more and more applications have strong connections with network. Accordingly, end user's pattern is also significantly changed to rely on networked resources. At the other hand, network service is not always available anywhere, anytime, especially for mobile user. For example, in a long air travel, one mobile user will have to spend entire trip on his laptop in network disconnection mode. The user will feel frustrated if he can not:

[1] Upload his modified document to his company/home server or update data in his company/home database immediately after the modification is done.

[2] Use ftp/sftp to transfer files between laptop and company/home machine.

[3] Telnet/rsh to his office/home machine to use latex to work on his proposal, or use his favorite software installed in office/home machine. However, the software can only run in office machine's operating system and can not run in laptop due to various reasons.

[4] Assume there is a web application in office machine that manages student's all course information. When the mobile user (professor) finished homework grading in flight, he want to input them to this system because of both work continuousness and enough available time currently. There are many other similar services in office machines that mobile want to access.

For scenario 1, it is possible to be partially serviced using existed disconnected file operation infrastructure, such as Coda [1], UbiData [12] among others. For Scenario 2, user can memorize and later manually handle it. However, for scenario 3 and 4, user will have to wait until network comes back later. This kind of interruption to work continuousness is very frustrated for every user.

Past researches explored disconnected operation for file system and database access. In this paper we introduce and evaluate a unified network service provider framework. Under this framework, various network service access, not only file/database access, can be seamless streamlined.

The framework we suggested targets for following goals.

○ *Zero Touch Adaptation*

Applications and services should not be extensively modified or adapted to suit for mobile environment. One reason is that volume of application/service is so big that it is not feasible to adapt all applications/services to meet the mobile environment. Another reason is that for many commercial software, it is impossible to acquire licenses and adapt them.

- *Transparency*

Our adaptation methodology is transparent to both user and applications. Both user and application can take it granted that he/it is accessing the network resource just like network is completely on. Our framework takes over the job of disconnected service providing and later the re-integration.

- *Complete Service Support*

Even in disconnection mode, we provide complete, 7/24, networked services (as if) by remote computers.

- *Heterogeneous Data Source*

Data source is heterogeneous: it can be a web application, network file system, distributed database system, or even DNS query etc.

In order to accomplish these goals, we propose to combine the disconnection operation and system virtualization. Before disconnection, mobile user can prepare a cloned system image of remote system. Disconnected service can always be emulated by the cloned virtual system. The framework we proposed aims for seamlessly support to system virtualization and reintegration without user's intervention.

The rest of this paper is organized as follows. In section 2, we survey related work. In section 3, the architecture of UbiNet is presented. Implementation technique is introduced in section 4. We present system evaluation in section 5 and conclude the paper in section 6.

II. RELATED WORK

Our work is related to three fields of past researches.

The first related field is disconnected operation. This field is extensively studied in mobile file systems and recently in database access. Coda [1] and Network File System (NFS) version 4 [5] introduce distributed file system and exploit disconnected operation for data hosted in these file systems. UbiData [12] implements a middleware to automate file replication, disconnected operation and integration for any data in any physical file system. Windows operating system [6] also supports disconnected file operation since Windows 2000. Some DBMSs, like Oracle [8] and Microsoft SQL server [9], also provide disconnected operation. Microsoft ADO.NET of .NET framework [7] supports disconnected relational data access from database application aspect. IBM websphere everyplace [] provides disconnected operation for e-business. The essential technology used in disconnected operation is to hoard needed data to mobile device before network disconnection, and redirect network access to local replicas in disconnection mode. Local update will be propagated to network resource later when network comes back.

However, disconnected operation we encounter so far only limits to a very special domain, either file system or relational database.

The second filed that closely related to our work is system virtualization. System virtualization is a popular concept in recent years. Lots of system virtualization software, both commercial and open source, emerged in the past decade. Some typical examples include VMWare [13], Microsoft Virtual PC (Derived from Connectix) [14], Bochs [11], CoLinux [10], among others. Different from the program running environment such as Java Virtual Machine (JVM), system virtualization provides a mechanism to simulate all aspects of computer system: hardware, operating system and application software. In UbiNet, we adopt concept of virtualization to simulate remote computer from local mobile device in disconnection mode. Combining both traditional disconnection operation and system virtualization to support mobile computing is, as far as we know, the first attempt.

System image will usually occupy large disk space. It's not efficient to generate image and replicate modified local image between mobile computer and remote computer. To minimize overhead, we adopted delta based imaging and re-integration. Delta based synchronization is a traditional and effective method to save network bandwidth. In mobile computing field, it is also widely used [2].

III. UBINET ARCHITECTURE

In UbiNet architecture, any system service in remote machine is aimed to be available 24/7. This is achieved by following mechanism.

First, UbiNet provides tools to image the whole remote system to local machine. Before expected disconnection, mobile user can clone a virtual machine of remote system. This virtual machine includes every bits in remote system, ranging from operating system and utility software to user's data.

Second, in disconnection mode, an interpreter will be executed to host the imaged system. This virtual system play same role as the remote system from user perspective. Requests to the remote system are transparently redirected to the locally running virtual machine. So any service that remote machine supports is active from the local virtual machine.

Third, the local modification on virtual machine is automatically propagated by UbiNet to the "hard" remote system once network is on.

These steps ensured both disconnection operation for all service and seamless connection between virtual machine and remote "hard" system.

In order to make the disconnection operation ubiquitous, we aim for hardware and software independence for both mobile host and emulated remote machines. To achieve this goal, we follow the following guideline:

A. Simulation software is written in ANSI C and standard C++ and library, so it can be easily ported to any platform.

B. The virtual machine is a pure "emulated" environment; no "native" execution is used. However, in future work, we plan to introduce Just-In-Time (JIT) translation to boost emulation performance.

C. No instrumentation to remote system's operating system is needed. This requirement is critical, otherwise availability of source code of remote operating system will become a blocking issue. Even for free source operating system, the version problem will also make instrumentation impossible. The operating system bits in mirrored image are exactly same as that of remote system. This guideline effectively makes our system support any emulated operating system, including future operating system.

UbiNet's hardware and software independent simulation provides highest computing availability. Its instruction-by-instruction interpretation by software interpreter lowers its performance compared with hardware execution based emulation, such as VMWare [13]. This drawback is and will be mitigated by faster and faster computing power of today's and future computers. Our focus in disconnection mode is concentrated on data and service availability instead of performance.

We will discuss detailed operation for every stage in subsequent sections

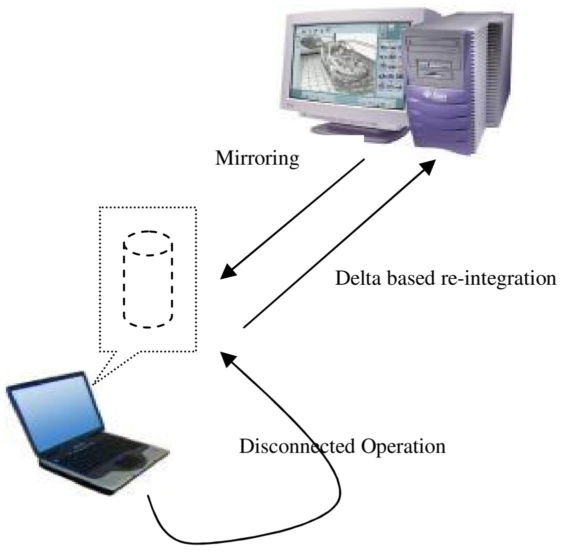

Figure 1. UbiNet Architecture

○ *Mirroring remote system*

UbiNet provides tools to create hard disk image of remote system. This image will be used by the local virtual machine in disconnection mode.

Mirroring the whole system is a non-trivial task. There are both timing and space issues. Currently, mirroring a Linux system takes about 1 GB storage. When storage is becoming cheaper and cheaper, this issue is not severe. We can generate image and store it to removable media (such as CompactFlash card or DVD) and plug it to mobile computer.

For the first time mirroring, it will take relatively longer time. However, for second and later mirroring, we use incremental mirroring to reduce time. User can also use our pre-existing, standard system image then apply incremental mirroring.

A mirrored image logically consists of three parts as illustrated in Figure 2. Physical implementation is very different when considering both performance and flexibility of expansion/shrinking of sections.

Meta-information section contains basic description of image and image modification. The most important information stored here is a map between disk block and image region. Interpreter will rely on these information to convert disk operation in virtual machine to file operation in mobile machine.

Read only image section contains the block-by-block (or sector-by-sector) mapping of mirrored system. However, to save storage space, unused blocks in mirrored system can be optimized out of this image. When disk image section is optimized, the meta data section can also be optimized similarly.

Modification appending section contains all the modification to the image since last image generation. When a system is first mirrored, the third part is empty. Local modifications happened in disconnected operation and delta mirroring are stored in this section. This image structure design also provides possibility for delta-based re-integration. UbiNet also provides tool to merge the modification part into read only image section. Merge procedure is non-reversible.

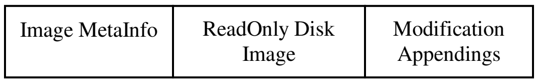

Figure 2: Mirrored Image Structure

System mirroring is only applicable to system that mobile user has access privilege.

o *Disconnected Operation on Virtual Machine*

In disconnection mode, the virtual machine that runs the image mirrored in previous step replace the role of "hard" remote system. All requests to remote system are automatically redirected to the virtual machine. The virtual machine mimics the actions of emulated remote machine instruction by instruction.

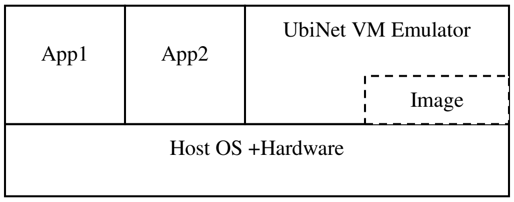

Figure 3: UbiNet Run Time Structure

By faking DNS name and IP address, all network requests from local mobile host to the simulated remote machine are redirected to the emulator. Emulator runs operating system and application in the virtual hardware configured same (at our best) as remote machine. We start our virtual machine simulator by modifying Bochs [11], which is an open source x86 emulator.

Inside the virtual system, if a disk write to a particular disk block is requested for the first time, the emulator employs Copy-On-Write (COW) technique to make the write operation redirected to a cloned block region in modification appending section instead of the read only image section. All subsequent operations (read or write) to the given block will also be redirected to this region. Meta data section contains information to direct access to dirty blocks to modification appending section.

o *Delta-based File Level re-integration*

Since mirrored image of remote system is a block by block mapping of simulated hard disk, the simplest re-integration is to ship modified blocks to remote system and apply them there. However, direct operating from disk block level is not a good way because both our emulator and remote system may relocate blocks. We decide to conduct re-integration from file level.

Past researches have shown that most modification ratios (defined as size of modified portion divided by size of whole file) are very small and delta based synchronization is simply effective in reducing network bandwidth. In UbiNet architecture, we can restore two versions. One is original file whose blocks are completely in read only section; one is up-to-date version whose blocks are partially or completely in modification appending section. After network comes back, these two versions are computed to get delta, which is then shipped to remote system and applied there to the original file.

IV. System Implementation

We implemented UbiNet prototype to validate its architecture and design. In our prototype, we implemented following components: system mirrorer, network redirector, UbiNet VM Emulator, and re-integrator.

System mirrorer can generate a skeleton image file and clone remote system block-by-block (sector by sector) to the read only image section.

Network redirector can redirect network requests to emulated virtual machine in disconnection mode. It is implemented based on netfilter [15] in Linux platform and network filter device driver developed using DDK [16] in Windows platform.

Virtual machine emulator is the core of UbiNet. It is implemented in standard C/C++ so that it can run in multiple architecture. Currently, emulator can only interpret x86 instruction set. Therefore, emulated binary must be running in x86, such as Windows, Linux for x86, DOS, etc. We are planning to extend emulated hardware platform to other popular instruction set, like IA64, Sparc, MIPS, etc. Our emulator is built based on Bochs [11] project.

Re-integrator retrieves two versions of locally modified files from read only image and modification appending section respectively. Then Xdelta [17] algorithm is used to generate patch. This patch will be shipped back to remote machine to be applied. Two daemon programs running in both mobile host and remote machine perform all these actions.

V. SYSTEM EVALUATION

In this section, we validate and evaluate the performance of our implementation. The hardware in our test is Compaq Presario 6000 with AMD althon CPU, 384MB RAM, and IDE 40GB hard disk. The host OS we used is Windows 2003 server with Service Pack 1 and remote OS is Linux Debian 3.0 distribution.

o *System Mirroring*

We create partitions with size 100MB, 500MB and 1GM and measure mirroring speed. The result is shown in table 1 and 2. Direct mirroring is fast but consumes unnecessary space. Compressed mirroring consumes significantly more time. However, due to delta mirroring, such time-consuming process happens only once.

TABLE 1. TIME FOR DIRECT MIRRORING

System Size	100MB	500MB	1GB
Time (min)	0.3	1.6	3.5

TABLE 2. TIME FOR COMPRESSED MIRRORING

System Size	100MB	500MB	1GB
Time (min)	31.6	269.5	765.3

o *Delta Mirroring*

Performance of delta mirroring is highly dependent on the changes since last mirroring. We use timestamp of file/directory to quickly target the modified files. Modifications are converted to changes in disk blocks and are stored in modification appending section. Meta data should also be updated. Table 3 shows two versions of two popular software suite. When these software are updated from first version to second version, the time for delta mirroring is negligible.

TABLE 3. TEST WORKLOAD AND DELTA MIRROR PERFORMANCE

Workload	apache	Bison
Original Version	1.3.0	1.27
Updated Version	1.3.1	1.28
Size	1.1MB	317KB
Time	< 1 sec	< 1 sec

o *Storage Overhead of Delta Support*

Talbe 4 shows overhead for delta mirroring with same workload used in table 3. The overhead includes new allocation in both meta data and modification appending sections.

TABLE 4. DELTA OVERHEAD

Workload	apache	Bison
Overhead	4.5MB	1.7MB

o *Delta re-integration*

Table 5 shows delta re-integration overhead. As expected, delta based file re-integration can significantly reduce replication size.

TABLE 5. DELTA RE-INTEGRATION

Workload	apache	Bison
Overhead	108KB	29KB

VI. SUMMARY

In this paper, we present the UbiNet universal framework that is used to streamline service access in network disconnection mode. To provide 7/24 system service, we propose to clone the remote system and run locally as a virtual machine. The virtual machine will provide all services of that machine. Locally modified data will be re-integrated back to the actual "hard" system when network comes back using delta based algorithm.

In the future work, we are planning to extend emulated binary to a set of popular instruction set, and port and experiment emulator to more platforms. Performance improvement is also a focus for future work.

REFERENCES

[1] M. Satyanarayanan, Coda: A Highly Available File System for a Distributed Workstation Environment. Proceedings of the Second IEEE Workshop on Workstation Operating Systems, Sep. 1989, Pacific Grove, CA

[2] A. Khushraj, A. Helal, and J Zhang, Incremental Hoarding and Reintegration in Mobile Environments. Proceedings of the IEEE/IPSJ

International Symposium on Applications and the Internet (SAINT), Feb 2002 Nara, Japan.

[3] H Lei, and D Duchamp, An Analytical Approach to File Prefetching, USENIX Conference Proceedings, Anaheim, California, Jan. 1997.

[4] http://www-306.ibm.com/software/pervasive/ws_everyplace_access/

[5] NFS version 4 home, http://www.nfsv4.org/

[6] Microsoft Windows, http://www.microsoft.com/windows

[7] Microsoft .NET technology, http://www.microsoft.com/net/

[8] Oracle, http://www.oracle.com

[9] Microsoft SQL server, http://www.microsoft.com/sql

[10] CoLinux, http://www.colinux.org

[11] Bochs project, http://bochs.sourceforge.net/

[12] J. Zhang, S. Helal, UbiData: Ubiquitous Mobile File Service, Proceedings of the ACM Symposium on Applied Computing (SAC), Melbourne, Florida, March 2003.

[13] VMWare, http://www.vmware.com

[14] Microsoft Virtual PC, http://www.microsoft.com/windows/virtualpc

[15] Netfilter, http://www.netfilter.org

[16] Windows DDK, http://www.microsoft.com/whdc/devtools/ddk/

[17] J. P. MacDonald, “File System Support for Delta Compression,” M.S. thesis, Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, 2000.