Table of Contents – Part II

Studies on the Key Technologies of Multi-Platform Mobile Thin Client System: Cross-Layer Isolation and Session Allocation ................................................. 1
  Biao Song, Wei Tang, Tien-Dung Nguyen, Sang-Ho Na, Jun-Hyung Lee, and Eui-Nam Huh

LDPC Equalizer for Compensating the CFO and Phase Noise in OFDM System .................................................................................................................. 11
  Do-Hoon Kim and Heung-Gyoon Ryu

TC-HMIPv6: A Study of HMIPv6 Handover Management for Packet Transmission Analysis ..................................................................................................... 20
  Sung-Gyu Kim, Farkhod Alisherov, and Byungjoo Park

A Multi-hop Communication Scheme for IEEE 802.11p Based V2V Communication Systems ............................................................................................. 26
  Woong Cho and Hyun Seo Oh

A Political Communication Scheme of Citizen Network System on Disembedding and Embedding Principle ................................................................. 34
  Jang-Mook Kang and Bong-Hwa Hong

Web Contents Mining System for Real-Time Monitoring of Opinion Information ............................................................................................................ 43
  Ho-Bin Song, Moon-Taek Cho, Young-choon Kim, and Suck-Joo Hong

An Energy-Efficient Cluster-Based Routing in Wireless Sensor Networks ......................................................................................................................... 57
  Seongsoo Cho, Bhanu Shrestha, Keuk-Hwan La, Bong-Hwa Hong, and Jongsup Lee

A Management of Resource Ontology for Cloud Computing ......................................................... 65
  Hwa-Young Jeong and Bong-Hwa Hong

Development of an Algorithm for Video Quality Measurement for Broadcasting Communications Services .................................................................................. 73
  Sang-Soo Kim, Hae-Jong Joo, and Euy-Soo Lee

An Effective Resource Managements Method Using Cluster-Computing for Cloud Services ..................................................................................................... 83
  Seong-Sik Hong and Jin-Mook Kim
Studies on the Key Technologies of Multi-Platform Mobile Thin Client System: Cross-Layer Isolation and Session Allocation

Biao Song, Wei Tang, Tien-Dung Nguyen, Sang-Ho Na, Jun-Hyung Lee, and Eui-Nam Huh

Department of Computer Engineering
Internet Computing and Network Security Lab
KyunghEE University Global Campus, South Korea
{bsong,wtang,junhyung,johnhuh}@khu.ac.kr,
{ntiendung,shna}@icns.khu.ac.kr

Abstract. Virtualization was considered as the best way to isolate independent thin client sessions on a physical machine. However, the hypervisor, guest OS and guest remote server not only consumes a considerable amount of memory but also degrades the processing power of CPU. In this paper, we propose a novel cross-layer isolation technology to support independent user sessions with only one OS and one remote server. Furthermore, a session allocation/migration algorithm is introduced in this paper. The algorithm solves the multi-machine allocation/migration problem within thin client environment.

Keywords: Thin client, Multi-user, Isolation, Allocation, VM Migration.

1 Introduction

Nowadays, the rapid development of network promotes the investigation of thin client (remote display) technology. Using thin client system, users are able to remotely control other computers (servers) and delegate actual information processing to them. Thus, thin client technology provides a powerful way to break the barrier between diverse applications and insufficient local hardware/software environment. For example, a mobile device with thin client system permits its user to use the applications running on different mobile platforms (Android, iOS, and so on) or PC platforms (Windows, Linux, and so on). In our previous paper [1], we introduced Multi-platform Mobile Thin Client architecture and invented the concept of cross-platform application market running on Cloud server where ubiquitous access to individual applications and PaaS/SaaS are enabled through thin client system. The features of the Cloud server, such as virtualization, flexibility, security, and dynamic management can be fully utilized to support a large number of thin client users. However, the previous architecture cannot achieve mass acceptance until we find practical solutions to the following technical challenges.

First of all, one physical machine should be able to support multiple thin client users. VM, as a completely isolated operating system installation within your normal
operating system, can be viewed as a feasible way to achieve isolation for multi-user sessions. However, attempting to create VM for every single user session is not always efficient as VM implementation itself consumes considerable amount of memory, and degrades the capacity of CPU. Most of existing thin client systems focused on supporting collaborative multi-user sessions. This focus has resulted in the majority of the work done in this area to be centered on the cursor management and remote display protocol towards heterogeneous client devices. While these approaches have resulted in improvements for collaborative scenarios, they could not provide an effective solution for independent multi-user sessions running on one physical machine.

Secondly, the server side should be capable to manage and allocate a multitude of user sessions using its local resources. Although the task placement problem in Cloud environment has been studied in a number of works [4-6], the problem we are facing is distinct from their problem. For one thing, VM is not necessary but alternative carrier for single user session; for another, the metrics of thin client QoS differs from the tradition QoS in Cloud environment.

In this context, we first introduce a novel thin client technology to support independent multi-user sessions running on one physical machine. This work is inspired by both THINC system [2] and VNC system [3] where the former one hooks data from server’s device driver layer and the latter one hooks data from server’s hardware frame buffer layer. Unfortunately, both of them are not directly applicable to our scenario. On the one hand, the device driver hooking requires complex translations among different device drivers as well as the hardware support from client device. Thus, using this model in mobile thin client environment may lose the generality since the hardware (especially the graphic component) of mobile client device are highly heterogeneous and usually insufficient to handle various applications. On the other hand, the hardware frame buffer hooking also has a non-trivial drawback in multi-user session environment, which is the failure of audio isolation. The audio information hooked from audio card frame buffer is a mixed signal including all the audio information from multiple user sessions. To address these shortcomings, we adopt a cross-layer approach, which is an effort to provide light weight isolation of remote display, audio playback and input functionalities for multiple user sessions. Comparing with VM isolation, the cross-layer isolation approach consumes less CPU and memory resources.

However, the cross-layer isolation approach also has drawbacks. The input interference is one of the problems that may degrade the QoS. For example, when one user is dragging the cursor, other users’ operations have to be paused until the cursor is released by that user. Besides, VM isolation also benefits workload balancing as VM can be migrated without losing running state. Thus, we consider both isolation approaches are alternative, and apply a QoS based selection approach considering the differences in application requirements. Meanwhile, a user session allocation algorithm is proposed. The two goals of session allocation are achieving better resource utilization and reducing service interruption time caused by input interference or VM migration.
The rest of this paper is organized as follows. Section 2 discusses some related work in thin client domain. Section 3 gives a brief overview of multi-platform mobile thin client architecture. A description of cross-layer isolation approach is given in section 4. Session allocation algorithm can be found in section 5. We conclude our work in section 6.

2 Related Work

Many collaborative supports have been done in the field of thin client. We found that the main theme among all of these researches was the need of “floor-control management”, which could be used to handle control of a synchronous task. Boyd introduced “fair dragging” where control of a user gains control of the floor once the mouse is dragged [7]. Another well-known study was collaborative VNC [8]. Collaborative VNC was a patch applied to the TightVNC server and client that provides managed collaborative sessions over the RFB protocol. With collaborative VNC, one user has the “floor” (i.e. controls the desktop) at any given time. Other users have the power to take control from or give control to other users at any time. Every user’s cursor is displayed, with each cursor assigned one of several colors. In [9], a multi-user collaborative support named “THINCing Together” was introduced to extend THINC system. The proposal contains a protocol that allows for asynchronous and synchronous multi-user session. It implemented centralized cursor management to optimize bandwidth usage for multiple users. These existing studies provide good solutions for the user input management in an independent multi-user scenario. However, they are not sufficient in terms of remote display and audio support for independent user sessions since they consider same screen and audio output for all users.

The remote display protocols can be categorized into three distinctive groups. At application/OS layer, Remote Desktop Protocol (RDP) is a typical protocol developed by Microsoft, which concerns providing a user with a graphical interface to another computer. RDP clients exist for most versions of Microsoft Windows (including Windows Mobile), Linux, Unix, Mac OS X, Android, and other modern operating systems. At device driver layer, THINC uses its virtual device driver to intercept display output in an application and OS agnostic manner [2]. It efficiently translates high-level application display requests to low-level protocol primitives and achieves efficient network usage. At hardware frame buffer layer, VNC uses the RFB protocol to remotely control another computer [3]. Server only sends the graphical screen updates back to the client. In [10], a hybrid protocol was proposed to handle multimedia streaming and interactive gaming applications. Both VNC RFB-protocol and THINC protocol are alternative in this system. Among these protocols, RDP and THINC require graphical hardware support from client device which is hardly provided by mobile thin client device. VNC RFB-protocol, on the other hand, is more general and flexible since it can fully utilize the GPU of server to support mobile clients. Thus, we choose VNC RFB-protocol with non-overlapping window placement to support remote display for independent multi-user sessions.
3 Multi-platform Mobile Thin Client Architecture

This section introduces the Multi-Platform Mobile Thin Client Architecture. In Fig.1, we present an overview of the architecture. The architecture is composed by several mobile thin clients and a multi-platform thin client server. To receive services, every mobile device must install a thin client viewer which provides all functionalities of remote access. Traditionally, a thin client ran a full operating system for the purposes of connecting to server. A newer trend is called a zero client, which no longer runs a full operating system: the kernel instead merely initializes the network, begins the networking protocol, and handles display of the server's output. Thus, it is not necessary to have an OS on mobile thin client device.

![Multi-Platform Mobile Thin Client Architecture](image)

**Fig. 1. Multi-Platform Mobile Thin Client Architecture**

As Fig.1 shows, the multi-platform mobile thin client server contains a number of modules. At first, the mobile terminal sends a request to authentication module. The request includes the user’s ID, password and terminal information. Then the user selects applications from application store, and provides necessary information to the payment system. The application request is delivered to the task manager. The task manager receives current resource condition and application profiles from the QoS monitor and the internal data repository, respectively. By analyzing the information, the task manager performs task allocation and real-time migration on local resources (only for VM). The QoS monitor continuously monitors the QoS information of running applications as well as the resource condition of local resources. QoS monitor also detects QoS violation, and send reports to task manager for task migration (only for VM).

The remote display module, audio support module and remote input module are the three components of thin client protocol. They provide all functionalities of remote access on the server side. Local resources are composed by physical machines, OSs, hypervisors, VM images, and applications. The actual processing power, platform, and services are provided by local resources. The local resources exchange VM image
and application data with the internal data repository in order to save or load user sessions. Meanwhile, the application running on local resources may interact with the external data source at any time.

The following section will focus on the isolation technology that allows one physical machine to support multiply independent user sessions without using VM.

4 Cross-Layer Isolation Technology

The two issues to be discussed when designing a multi-user thin client system are how to get graphical/audio updates and how to isolate input/output for multiple user sessions. Fig.2 contains a three-layer interception model illustrating the approaches that allow the graphical/audio updates to be retrieved and redirect to the client. This three layer model was first proposed by R.Baratto in his Ph.D. thesis [2]. The original version demonstrates only the display pipeline while we extend the same concept to the audio/input pipeline as well.

There exists three interceptions points in the pipeline: (1) between the applications and the device independent layer, (2) between the device independent layer and the device dependent layer, and (3) between the device dependent layer and the hardware layer. To utilize them, the server side must be able handle the application/OS interfaces, the device drivers and the hardware frame buffer, respectively. The requirement for client side also differs from one interception point to another. The interception between the applications and the device independent layer requires client’s OS support and hardware processing capability. Intercepting between the device independent layer and the device dependent layer needs client’s hardware processing capability. Hardware frame buffer interception merely requires the A/V playback functions on client device.

![Fig. 2. Cross-layer isolation vs. VM based isolation](image)

Since the gap between the capacity of mobile graphic card and the requirement of PC applications always exists, using the first or the second interception point to support remote display cannot be the proper choice in our scenario. However, the hardware frame buffer interception also has two disadvantages. First, display updates consisting of raw pixels along are typically too bandwidth-intensive. Second, the intercepted signals including audio and video are mixed already, which means we
need extra isolation technology to support multiple user sessions. The first problem can be solved by optimizing the remote display protocol and the network condition. A successful example is the latest version of VNC, which is able to provide fluent video display under current network circumstance. To solve the second problem, we invent a new isolation approach named cross-layer isolation.

Fig.2 also shows the detail information of cross-layer isolation and the well-known VM based isolation. Unlike the VM based isolation in which the hypervisor takes the responsibility to divide and manage the hardware resources, the cross-layer technique has three isolation components deployed between the user sessions and the remote server. The motivation of our design is from the fact that the VM based isolation consumes considerable amount of resources to maintain VM, guest OS and remote server for every single user. Since the modern OSs have the capability to manage several processes running concurrently, the QoS of user sessions can be guaranteed by OS as long as the server has sufficient resources to handle the tasks.

The reason we choose the name “cross-layer” is that the audio isolation, the video isolation and the input isolation are implemented in different layers. To isolate the graphical output of one user session from others, the server assigns a non-overlapping rectangle area for each user session. While hooking the whole screen information from the hardware frame buffer, the server can easily extract any user session from the picture by using the coordinates of the corresponding rectangle. Then using RFB protocol, the updates are distributed to the users continuously.

The audio isolation utilizes device driver interception. Before the audio signals from multiple user sessions are mixed, the server hooks them and sends to the corresponding clients. For one thing, it is very hard to extract one session’s audio information from a mixed audio signal retrieved from hardware frame buffer; for the other, the mobile clients are able to provide the hardware processing capability for audio. The RTP (Real-time Transport Protocol) is adopted as the protocol for audio transmission.

Unlike video and audio isolation, the input isolation intercepts on the client side and performs the actual input on the server side. The input isolation on server side needs the APIs of server OS. Taking our implementation on Microsoft Window XP as an example, the input isolation module requires the handle of each user session. The user inputs are managed by a multi-queue system. When the server wants to perform a user input, it first activates the corresponding session using the handle, and then simulates the mouse or keyboard event. We adopt Shortest-Remaining-Size-First (SRSF) preemptive scheduler known to be optimal for minimizing mean response time and improving the interactivity of a system [11].

In terms of resource consumption, the cross-layer isolation approach is more efficient than the VM-based isolation approach; nevertheless it can be only an alternative solution since it has two disadvantages. First, input interference may occur. Second, session migration is not possible. In next section, we will discuss about these problems and give a primal solution to choose the isolation technology and to allocate user sessions.
5 User Session Management

5.1 Isolation Selection Approach

As we introduced in section four, all the user sessions supported by cross layer isolation are running on same OS. While one session gains the control of mouse/keyboard, the inputs of other sessions cannot be performed until the control is released. Since the control shift among these sessions should be made in a quick and implicit manner, we need to figure out which operation may hinder others from getting the control. Any mouse/keyboard operation can be viewed as one of the following two events: instant event and continuous event. Instant events usually do not interfere with each other whereas the total amount of instant events cannot be larger than that can be handled by the input isolation component. Continuous events, such as mouse dragging, greatly interfere with instant events and themselves. Therefore, we need to profile the averaging number of instant event and the cumulative time of continuous event for each application.

Let $NI_j$ be the number of instant events generated by the user of application $j$ within one minute, $TC_j$ be the cumulative time of continuous events generated by the user of application $j$ within one minute. If the value of $NI_j$ or $TC_j$ highly depends on user’s behavior, we need to create profile for each user. Let $MAX(NI)$ be the maximum number of instant events that can be handled by the input isolation component within one minute, $MAX(TC)$ be the maximum time that can be occupied by continuous events within one minute. The value of $MAX(NI)$ depends on the hardware and software environment. The value of $MAX(TC)$ should guarantee that no user feels the existence of other user.

We define the first part of selection approach as follow: given a user session is application $j$, if $NI_j > MAX(NI) \times 50\%$ or $TC_j > MAX(TC) \times 50\%$, the cross layer isolation cannot be applied to this session. Using $MAX(NI) \times 50\%$ and $MAX(TC) \times 50\%$ rather than higher values leaves enough space for allocating other sessions supported by cross-layer isolation.

The CPU consumption is also considered as an important factor. Since the cross-layer isolation technology does not support session migration, it is not suitable for the sessions whose CPU consumption fluctuates all the time. Let $t$ be the duration between the time when the CPU is overloaded and the time when QoS monitor triggers a migration. Given an application $j$, the profiling of its CPU consumption takes time $n \times t$ where $n$ should be large enough to present the averaging CPU consumption of application $j$. By that, we can get $n$ averaging CPU consumption information $\{AC_{j1}, AC_{j2}, \ldots, AC_{jn}\}$ and an overall averaging CPU consumption
information $AC_j$. Let $FC$ be the free CPU capacity to buffer the unexpected workload burst.

We define the second part of selection approach as follow: given a user session is application $j$, if $\max\{AC_{jk} | 1 < k < n\} - AC_j > FC \times 50\%$, the cross layer isolation cannot be applied to this session. The reason of using $FC \times 50\%$ is also to facilitate session allocation.

5.2 Session Allocation Approach

Given an application $i$ and a physical machine $j$, let $c_{ij}$, $m_{ij}$, $g_{ij}$ and $b_{ij}$ be the percentages of resource usage regarding CPU, memory, GPU and network bandwidth, respectively. If the application is supported by VM, the extra resource consumption of VM, OS and remote server should be also included. Let $fc_j$, $fm_j$, $fg_j$ and $fb_j$ be the percentages of idle CPU, memory, GPU and network bandwidth resources, respectively, on machine $j$. For any resource, at least 25% free capacity should be kept to buffer the unexpected workload burst, and should not be counted in the available idle resources. The application can be allocated to that physical machine only if the following condition can be fulfilled:

$$c_{ij} \leq fc_j \& m_{ij} \leq fm_j \& g_{ij} \leq fg_j \& b_{ij} \leq fb_j,$$  \hspace{1cm} (1)

While allocating both cross-layer isolated session and VM isolated session, there are two common objectives: (1) avoiding overuse of any resource, and (2) providing more CPU resource for the session. To explicitly define the objectives, we need new notations.

After application $i$ is allocated on physical machine $j$, the average percentage of free resource $ap_{ij}$ is defined as:

$$ap_{ij} = \frac{(fc_j' + fm_j' + fg_j' + fb_j')}{4}$$

$$fc_j' = fc_j - c_{ij}, \hspace{0.5cm} fm_j' = fm_j - m_{ij}, \hspace{0.5cm} fg_j' = fg_j - g_{ij}, \hspace{0.5cm} fb_j' = fb_j - b_{ij}.$$ \hspace{1cm} (2)

For machine $j$, the resource utilization condition after allocating application $i$ is denoted as $RUC_{ij}$, which is a mean-square value and is given by:

$$RUC_{ij} = (fc_j' - ap_{ij})^2 + (fm_j' - ap_{ij})^2 + (fg_j' - ap_{ij})^2 + (fb_j' - ap_{ij})^2$$ \hspace{1cm} (3)

We assume that the applications running on a single physical machine share all of the CPU capacity in a proportional way. To be more specific, application $i$ to be allocated on physical machine $j$ will get $(c_{ij} / (1 - fc_j')) \times 100\%$ of the CPU capacity.
Regarding application $i$, the superiority of machine $j$ is defined as $S_{ij}$:

$$S_{ij} = c_{ij} / ((c_{ij} / (1 - f_{ej}^i)) = 1 - f_{ej}^i$$ (4)

For cross-layer isolated session, the cumulative number of instant events and the cumulative time of continuous events need to be considered. Let $NI_{ij}$ or $TC_{ij}$ be the cumulative number of instant events and the cumulative time of continuous events on machine $j$ after application $i$ is allocated. The conditions $NI_{ij} < MAX(NI)$ and $TC_{ij} < MAX(TC)$ must be satisfied. Based on that, we define the trade-off metric for allocating cross-layer isolated session as follows:

$$CT_{ij} = \alpha \times RUC_{ij} + \beta \times S_{ij} + \varepsilon \times (NI_{ij} / MAX(NI) + TC_{ij} / MAX(TC))$$ (5)

$\alpha$, $\beta$, and $\varepsilon$ are the coefficients for adjusting the weight of each objective. A smaller the value of $CT_{ij}$ represents a better allocation for cross-layer isolated session.

For VM isolated session, input interference problem does not exist. However, any possible migration should be avoided after the VM is allocated. Although the latest VM technology allows a running VM to be migrated within 20 seconds, it still makes user uncomfortable. To solve that problem, using only $S_{ij}$ is not enough since $S_{ij}$ considers the averaging CPU consumption rather than the burst case. As we have the profile information $\max\{AC_{jk} \mid (1 < k < n)\} = AC_j$ to illustrate the stability of CPU consumption for any application $j$, we can easily figure out whether the CPU consumption on a physical machine is stable or not. The way is to sum up the stability values retrieved from all the applications running on that machine. We use $CS_j$ to denote the summation on machine $j$, and provide the trade-off metric for allocating VM isolated session as follows:

$$VT_{ij} = \alpha \times RUC_{ij} + \beta \times S_{ij} + \varepsilon \times CS_j$$ (6)

$\alpha$, $\beta$, and $\varepsilon$ are the coefficients for adjusting the weight of each objective. A smaller the value of $VT_{ij}$ represents a better allocation for VM isolated session.

Based on Eq. (5) and Eq. (6), we adopt a greedy approach to achieve the goal of efficient session allocation in our Multi-Platform Mobile Thin Client system.

6 Conclusions

We presented cross-layer isolation technology, a novel and efficient approach to support multiple thin client sessions using with one OS and one remote server. We introduced how to implement the video isolation, the audio isolation and the input isolation in
different layer. Meanwhile, we analyzed the advantages and disadvantages of both isolation approaches and provided an effective way to select proper isolation technology for an application using profiling method. A session allocation/migration algorithm was proposed in this paper to allocate multiple sessions on multiple physical machines.

Acknowledgement. This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education, Science and Technology (2011-0020515).

References

The 2011 International Conference on Future Generation Communication and Networking

The 2011 International Conference on Future Generation Communication and Networking will be held on December 8 – 10, 2011 in Jeju Grand Hotel, Jeju Island, Korea.

FGCN 2011 will be the most comprehensive conference focused on the various aspects of advances in Communication and Networking. Our conference provides a chance for academic and industry professionals to discuss recent progress in the area of Advanced Communication and Networking.

The goal of this conference is to bring together the researchers from academia and industry as well as practitioners to share ideas, problems and solutions relating to the multifaceted aspects of Communication and Networking. The complete list of topics will be updated some times later. Please check the Call for Papers section at http://www.sersc.org/FGCN2011/cfp.php

The proceedings of the conference will be published by LNCS/CCIS series (edited by Springer, indexed by EI and SCOPUS). And some papers selected will be included in some journals.
Studies on the Key Technologies of Multi-Platform Mobile Thin Client System: Cross-Layer Isolation and Session Allocation

Biao Song, Wei Tang, Tien-Dung Nguyen, Sang-Ho Na, Jun-Hyung Lee, and Eui-Nam Huh

Department of Computer Engineering
Internet Computing and Network Security Lab
KyungHee University Global Campus, South Korea
{bsong,wtang,junhyung,johnhuh}@khu.ac.kr,
(ntiendung,shna}@icns.khu.ac.kr

Abstract. Virtualization was considered as the best way to isolate independent thin client sessions on a physical machine. However, the hypervisor, guest OS and guest remote server not only consumes a considerable amount of memory but also degrades the processing power of CPU. In this paper, we propose a novel cross-layer isolation technology to support independent user sessions with only one OS and one remote server. Furthermore, a session allocation/migration algorithm is introduced in this paper. The algorithm solves the multi-user to multi-machine allocation/migration problem within thin client environment.

Keywords: Thin client, Multi-user, Isolation, Allocation, VM Migration.

1 Introduction

Nowadays, the rapid development of network promotes the investigation of thin client (remote display) technology. Using thin client system, users are able to remotely control other computers (servers) and delegate actual information processing to them. Thus, thin client technology provides a powerful way to break the barrier between diverse applications and insufficient local hardware/software environment. For example, a mobile device with thin client system permits its user to use the applications running on different mobile platforms (Android, iOS, and so on) or PC platforms (Windows, Linux, and so on). In our previous paper [1], we introduced Multi-platform Mobile Thin Client architecture and invented the concept of cross-platform application market running on Cloud server where ubiquitous access to individual applications and PaaS/SaaS are enabled through thin client system. The features of the Cloud server, such as virtualization, flexibility, security, and dynamic management can be fully utilized to support a large number of thin client users. However, the previous architecture cannot achieve mass acceptance until we find practical solutions to the following technical challenges.

First of all, one physical machine should be able to support multiple thin client users. VM, as a completely isolated operating system installation within your normal