MOBITHIN Management Framework: Design and Evaluation

Lien Deboosere*, Mario Kind†, Abdeslam Taguengayte‡, Pieter Simoens*, Bert Vankeirsbilck*, Fritz-Joachim Westphal†, Thomas Plantier‡, Filip De Turck*, Bart Dhoedt* and Piet Demeester*

*IBBT - Department of Information Technology, Ghent University
Gaston Crommenlaan 8, bus 201, 9050 Gent, Belgium
Email: lien.deboosere@intec.ugent.be

†T-Systems Enterprise Services GmbH
Goslarer Ufer 35, 10589 Berlin, Germany

‡Prologue
ZA de Courtaboeuf, 12 Avenue des Tropiques,
BP 73 - 91 943 Les Ulis Cedex, France

***Abstract*—In thin client computing, applications are executed on centralized servers. User input (e.g. keystrokes) is sent to a remote server which processes the event and sends the audiovisual output back to the client. This enables execution of complex applications on thin devices. Adopting virtualization technologies on the thin client server brings several advantages, e.g. dedicated environments for each user and interesting facilities such as migration tools. In this paper, a mobile thin client service offered to a large number of mobile users is designed. Pervasive mobile thin client computing requires an intelligent service management to guarantee a high user experience. Due to the dynamic environment, the service management framework has to monitor the environment and intervene when necessary (e.g. adapt thin client protocol settings, move a session from one server to another). A detailed performance analysis of the implemented prototype is presented. It is shown that the prototype can handle up to 700 requests/s to start the mobile thin client service. The prototype can make a decision for up to 700 monitor reports per second.**

I. Introduction

In the thin client computing paradigm, computation and storage are shifted from the client terminal (e.g. desktop PC, laptop, PDA) to the network. User applications are executed on a remote server and the client device only deals with user interaction and the presentation of the screen graphics. Since only basic functionality and processing power is required at the client, redundant hardware can be stripped of the device, resulting in thin devices that are lightweight and could be made energy efficient.

The thin client concept is very promising for mobile users. The mobile user can execute all applications, even without falling back on a restricted mobile version of the application.

Because all data and applications are shifted to the network, it is of the utmost importance that the thin client service is always on, is ubiquitously available and offers a high Quality of Experience (QoE). Current thin client deployments are mostly situated in corporate Local Area Networks (LAN), which are highly controlled environments offering fixed and stable bandwidth availability. The MobiThin project [1] aims to port the thin client concept into the wireless domain, either in a Local or Wide Area Network (LAN/WAN). At least in a WAN-environment, a broad audience of mobile users are connected through unreliable wireless connections. This imposes severe challenges for delivering a good quality of experience for the end user. Ideally, the user should get the same perceived application responsiveness as when running the application locally. These challenges can be met by an intelligent Service Management Framework. In Figure 1, three components of the mobile thin client service can be distinguished: a management server, a thin client server and a client terminal. The Service Management Framework is distributed among the three components. By monitoring the environment, the Service Management Framework enables adapting to variations in the state of the environment in which the mobile thin client service operates.

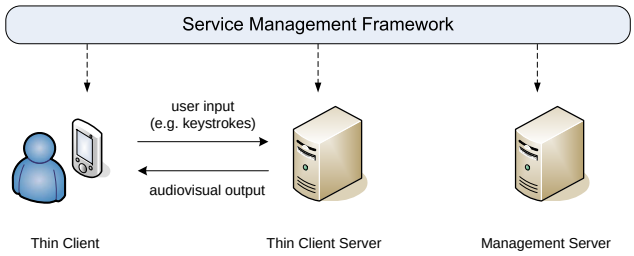


Fig. 1. Mobile thin client service: the service management framework is distributed among management servers, thin client servers and client terminals.

The remainder of this paper is structured as follows. Related work is described in section II. In section III is described how the mobile thin client service can adapt to variations in the state of the environment. The architecture of the Service

Management Framework is discussed in section IV. Technical implementation details of the prototype of the Service Management Framework are covered in section V. Detailed measurements on the response time of the prototype and a performance analysis are discussed in section VI. Finally, conclusions are drawn and future work is described in section VII.

II. Related work

With resource-limited mobile devices, visiting websites containing rich-content and executing complex applications is impossible. To enable rich web-content, a proxy-based approach is proposed in [2]. The active content is replaced with an AJAX-based remote display component. This proxy-based approach is limited to web content and executing complex applications on the mobile device remains impossible. In the current work, a thin client solution is proposed to enable both visiting rich web-content and executing complex applications on a mobile device. With that approach, all applications, even webbrowsers, run on a remote server. In [3], pTHINC is introduced to add specific functionality (e.g. server side screen scaling, portrait and landscape viewing modes) when visiting websites on the mobile device over a thin client protocol. Examples of thin client protocols are Citrix Independent Computing Architecture (ICA) [4], Virtual Network Computing (VNC) [5] and more recently THINC [6] and a hybrid thin client protocol that automatically switches between two modes depending on the amount of motion in the screen [7].

The management of current thin client deployments is straightforward. First of all, the limited number of users is well-known. The administrators know what application is executed by which user. Secondly, current thin client services are deployed in stable environments, usually in a corporate LAN. This means both the infrastructure and the network used by the thin client service are under control of the company itself. Examples of thin client management systems are Prologue's UseIT suite [8] and Citrix' Workflow Studio [4]. These systems target fixed users for accessing IT resources of a single business entity, while the service envisioned in this paper targets a wide range of IT resources accessible to a broad audience of mobile users.

To deliver a high QoE for a large number of mobile users in a WAN environment, an adaptive thin client protocol is required to cope with variations in the environment. However, not all environmental changes can be compensated by tuning the thin client protocol settings. Therefore, a Service Management Framework is presented in this paper, able to guarantee a high user experience for all users during their complete session.

III. Adaptivity

The mobile thin client service is offered in a dynamic environment: the wireless network conditions are unpredictable, but also the load on the thin client servers is unpredictable, since it depends on the amount and the kind of applications executed by the users, which is not known in advance. The mobile thin client service can adapt to changes in the environment in two ways: *(i)* by optimizing the thin client protocol settings and *(ii)* by migrating a user's session from one thin client server to another. Examples of optimizing thin client protocol settings can be found in [7], [9]. The migration of user sessions within the scope of adapting to changes in the environment is discussed below.

By supplying information about the state of the environment, the Service Management Framework can enhance the efficiency of the thin client protocol to increase the user's experience. However, not all situations can be solved by adapting the thin client protocol. For example, when a server gets overloaded, adaptations on the thin client protocol level might not be sufficient to keep guaranteeing a high user experience. The only solution is then to move one or more sessions from this server to another one. Migration can be used as a means for load balancing, but also to anticipate on the varying network delay due to the mobility of the users. When the network delay between the user's device and the thin client server executing his session becomes too high, moving the user's session to a faster reachable thin client server can enhance the user experience. Another application of migration is for maintenance purposes: all sessions from a server are moved to other thin client servers, in order to allow the system administrator to shutdown the server for maintenance.

It is important that a user session can be migrated efficiently, ideally without the user noticing. To avoid problems with process migration tools (e.g. open files, network connections, etc.) [10], virtualization technologies can be adopted. In that case, every user has his own Virtual Machine (VM). Migrating a VM, while the user stays connected, from one thin client server to another relies on the migration tools of the adopted virtualization technology also referred to as *live migration*. Examples of virtualization technologies supporting live migration are Xen [11], Kernel-based Virtual Machine (KVM) [12] and VMWare [13]. Adopting virtualization technologies on the thin client server also brings other advantages: *(i)* the VM's are isolated (e.g. if one VM crashes, it does not cause tearing down other VM's), *(ii)* VM's can be personalized (e.g. desktop background, application settings), *(iii)* different operating systems can be executed on a single server.

IV. Service Management Framework (SMF)

To support the mobile thin client service, able to adapt to variations in the environment, an intelligent Service Management Framework is needed. The architecture of the SMF is shown in Figure 2. The connection management component accepts requests from users and provides authentication and authorization. A component related to user management is required to handle subscription information, personal settings and to manage the user's session. The data management component is required to provide access to specific data on users, sessions and the infrastructure. A business support component is present for accounting purposes. An Application Delivery Service (ADS) is introduced to ensure the scalability of the mobile thin client service: since it is impossible to

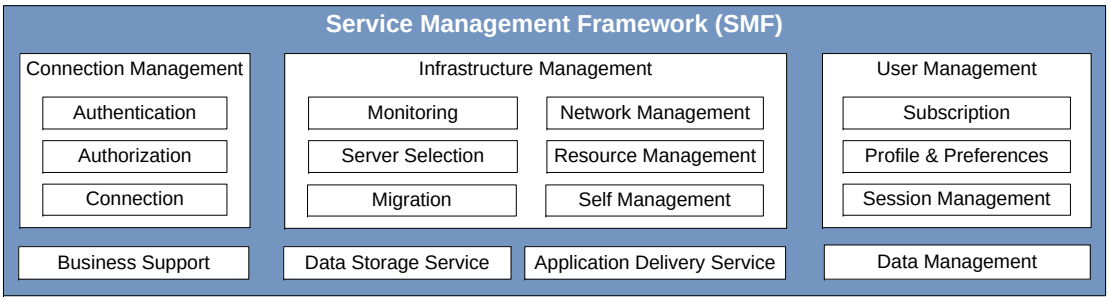


Fig. 2. Service Management Framework

assume all existing applications are installed in every VM, the ADS enables automatic distribution of applications to VM's. A technology that can be used for the ADS is called *application streaming* [14]. The storage of personal data as well as application delivery are services that can be provided by external providers. A component to communicate with these external services is added to the SMF. The most challenging component is the infrastructure management component which is discussed in more detail below.

Efficient management of the infrastructure requires a *monitoring* component to gather information on the current state of the infrastructure. Based on monitoring information, intelligent decisions can be made to optimize resource usage while guaranteeing a high user experience. The infrastructure management is responsible for the proper distribution of sessions among the available servers. The goal of this distribution is a trade-off between load balancing and energy efficiency. Load balancing should pursue an ideal distribution of load between available servers, for example, having nearly equal distribution of load among all servers. From an energy efficiency point of view, it is more efficient to concentrate the load on as less servers as possible. This allows to shutdown non-used servers and conserve energy [15]. Resource usage optimizations always have to take the user experience into account since the SMF is responsible to guarantee the desired quality to the users at all times. An intelligent *server selection* algorithm is required to select an appropriate server for each user, depending on the current location of the user, the current state of the infrastructure and the user's profile. When the conditions of the environment have changed in such way that a high user experience cannot be guaranteed anymore, the *migration* component will migrate the user's session to another, well-chosen server. *Network management* is another part of the infrastructure management. Since the mobile thin client service is offered in a WAN-environment, the network management handles different types of network connections (e.g. WiFi, UMTS). The network management is also responsible to negotiate QoS-classes with the network control component of the network operator. The *Resource Management* component keeps track of the resource usage and state of the whole system. One of the tasks of this component is guaranteeing the high availability of the mobile thin client service by implementing one or more resilience functionalities. For example, when a server has failed, the Resource Management component should automatically trigger a server failover mechanism. Another function of the Resource Management component is reserving resources for the users' sessions. Finally, the *Self Management* component is responsible for the proper working of the mobile thin client service. This component contains the intelligence of the mobile thin client service. It receives information from the Monitoring component about the state of the complete environment in which the mobile thin client service operates. Based on this information, it knows how to make an intelligent decision in order to adapt to variations in the environment and optimize the mobile thin client service.

V. Implementation Details

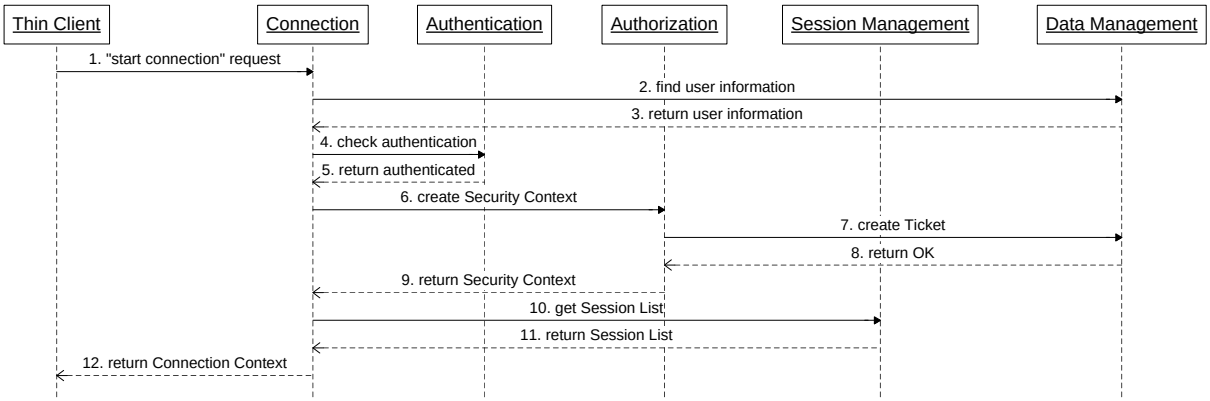
The implemented SMF-components of interest for this paper are shown in Table I. To enable rapid and simplified implementation of the Service Management Framework, Java EJB3 has been used. An EJB is a managed component controlled by a JEE application container. The container controls the lifecycle of the EJBs and the resources they are using. The design of the components takes into account they are running inside a JEE application container.

Two scenarios are described in detail: *(a)* Login and *(b)* Adaptation.

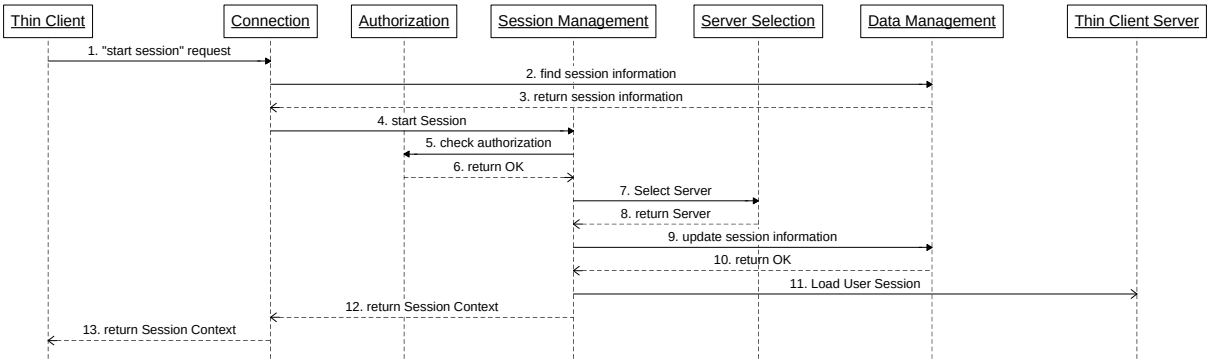
A. Login

The thin client only needs direct access to a single component of the implemented prototype: the Connection component. Since the mobile device has limited resources, we cannot assume a Java Virtual Machine is running on the client terminal. To easily grant the thin client access to the managed Connection component, it is exposed as a webservice. This allows easy accessing the component from different programming languages, since standard SOAP-messages are exchanged between client and server.

Starting the mobile thin client service is split in two parts: (i) starting a connection with a management server and (ii) starting a specific user session. During the first part, the management server checks whether the user is authenticated. When successful, a list of all the user's sessions is returned. During the second part, a request to start a specific user session is handled by the management server and information on how to reach the user session is returned to the user.



(a) Part 1: start connection



(b) Part 2: start session

Fig. 3. Sequence diagrams: starting the mobile thin client service

Component	**Description**
Authentication	Checks whether the user is authenticated or not
Authorization	Creates a Session Ticket for the user
Connection	Accepts incoming requests from user that want to start the service
Server Selection	Round-Robin selection of a server from the database (more complex load balancing algorithms can be plugged in)
Session Management	Coordinates the process to start a session
Data Management	Retrieve information from the database

TABLE I
IMPLEMENTED COMPONENTS ON THE MANAGEMENT SERVER

Figure 3(a) shows the sequence diagram of the first part of starting a mobile thin client session. When the Connection component receives a "start connection" request from a user identified by a username, it contacts the Data Management component to retrieve detailed information about this user (e.g. authentication information, session information and the user's profile). The Authentication component is responsible to check whether the user really is who he claims to be. When this test is passed, a session ticket is created by the Authorization component. The Data Management creates a new session ticket entry in the database and the ticket is returned to the Connection component. Then, a list of all the user's sessions is retrieved from the Session Manager. Finally, a Connection context object, containing the session ticket and the list of the user's session, is created and returned to the Thin Client.

When the first part of starting a mobile thin client session is completed, the user can choose which session he wants to start. He selects a specific session and a "start session" request is sent to the management server. The sequence diagram for the second part of starting a mobile thin client session is shown in Figure 3(b). Upon receipt of a "start session" request, the Connection component asks the Data Management component for detailed information about the selected session (e.g. state information, etc.). Next, the Session Management component takes over. The Authorization component checks whether the

user is allowed to start the desired session. When granted, the algorithm of the Server Selection component selects an appropriate thin client server to execute the user's session. While the user session is loaded on the selected thin client server, the Session Management component creates a Session Context object, containing the required information for the thin client to be able to start the thin client protocol session.

B. Adaptation

As stated before, the Self Management component receives monitoring information about the state of the environment. Monitoring information is exchanged using standard Simple Network Monitoring Protocol (SNMP) messages. When a problem is detected, the Monitoring component sends an SNMP trap message to notify the Self Management component. Based on the trap Object IDentifier (OID) encapsulated in the SNMP trap message, the Self Management component understands the problem and looks for a solution. In this paper, SNMP trap messages notifying the Self Management a thin client server is getting overloaded, are tackled. To lower the load on the thin client server concerned, at least one of the VM's running on that thin client server should be migrated to another thin client server. To avoid making simultaneous, conflicting decisions, the part of the Self Management that takes a decision is implemented as a singleton.

VI. Experimental Results

The performance of the two scenarios described in the previous section is evaluated. To approach a realistic use case, 100k users are subscribed for the mobile thin client server, each having on average 2 personal VM's. Since a thin client server should be able to serve 26 simultaneous users [13] executing complex applications, at least 100k/26 or 3847 thin client servers should be installed. Information on the subscribed users and their sessions is loaded in memory to avoid an I/O bottleneck due to reading and writing from and to a database.

A. Login

The testbed for evaluating the performance of the implemented prototype of the SMF consists of a single management server (2 AMD opteron 2212, 4GB RAM, 1Gbps NIC), which accepts requests from simulated users. The users are simulated with the Spirent Avalanche 2500 [16] hardware. This measurement device is able to generate 45.000 requests per second to a webserver, which certainly fullfills our requirements.

In Figure 4, the average response time users perceive during the first part of starting the mobile thin client service is depicted. It is clear that the response time increases exponentially when the number of requests/s increases. Beyond 800 requests/sec, the CPU of the management server becomes a bottleneck and time-outs occur on the client-side which should of course be avoided as much as possible in a real deployment. Similar conclusions can be drawn for the perceived response time during the second part of starting a mobile thin client session. Figure 5 shows up to 600 requests/s to start a specific session can be handled by the management server.

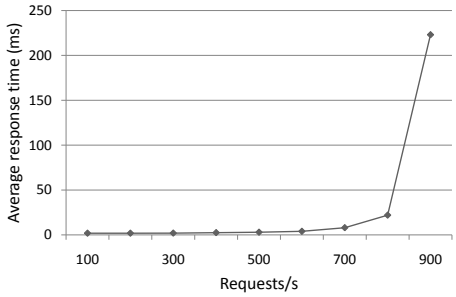


Fig. 4. Average response time when starting a mobile thin client session (part 1: authentication and authorization)

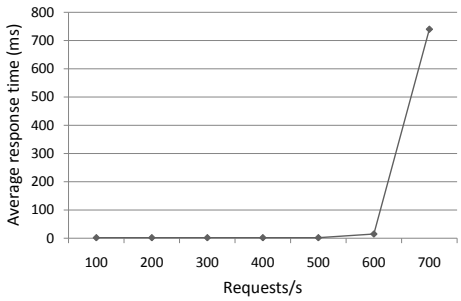


Fig. 5. Average response time when starting a mobile thin client session (part 2: select server and load session)

B. Adaptation

In this experiment, SNMP trap messages are sent to the management server when a thin client server becomes overloaded. The only solution for this problem is to migrate at least one of the VM's running on the thin client server to another one by means of the online migration tool of the adopted virtualization technology.

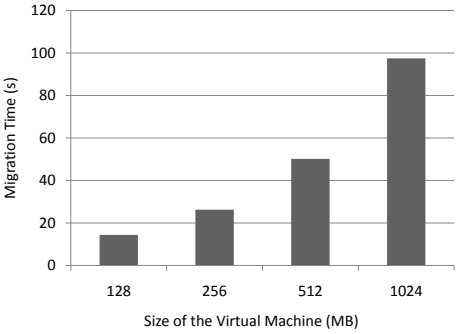


Fig. 6. Total time to live migrate a Xen Virtual Machine.

To evaluate the performance of the adaptation to overloaded thin client servers, an evaluation of migrating a VM from one server to another is described first. The testbed exists of two thin client servers (4 AMD Opteron 2212, 4GB RAM, 1Gbps NIC and Xen 3.1 virtualization software) and an NFS-server containing the filesystem of the VM. In Figure 6, the total time spent during the migration of a VM is shown. Note that only a very small downtime in the order of ms, is noticed by

the user [17]. From user perspective, it is important to avoid this downtime as much as possible, while from infrastructure perspective, the total time spent on VM migrations should be kept as small as possible. Figure 6 shows the time spent during the online migration of a user session depends on the size of the user session in terms of allocated RAM-memory. This means migrating multiple small user sessions (e.g. 128MB) is faster than migrating a single large user session (e.g. 1024MB) at the cost of two users perceiving a short downtime.

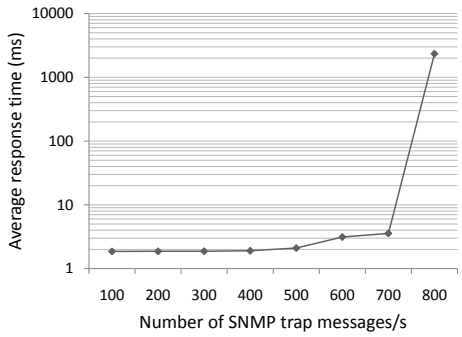


Fig. 7. Average response time of the Self Management component to an SNMP trap message.

In a last experiment, the implementation of the Self Management component is evaluated. The testbed consists of a single management server (AMD Opteron Processor 2350, 8GB RAM, 1Gbps NIC) and a server generating SNMP trap messages. When an SNMP trap message arrives at the management server, a random user session from the thin client server concerned is selected that should be migrated to another thin client server. The new thin client server is selected in a round-robin way from the list of available thin client servers. Figure 7 shows the average response time of the management server to solve a reported problem. Up to 700 messages/s can be handled within on average 3.5 ms. Beyond that boundary, the CPU load of the core executing the server selection component becomes overloaded. The average response time increases a lot and the framework begins dropping SNMP trap messages, which is unacceptable.

VII. Conclusion and future work

The architecture of a mobile thin client service has been presented. A detailed study on the architecture and implementation of the service management framework has been elaborated. Two scenarios have been briefly discussed: *(i)* login and *(ii)* adaptation. Login to the mobile thin client service was split in two parts. It was shown that the prototype of the Service Management Framework can handle up to 700 requests/s for the first part and up to 600 requests/s for the second part of starting a mobile thin client session. The CPU was identified as the bottleneck of the system.

The monitoring component detects changes in the environment and reports problems to the Service Management Framework. The case of an overloaded thin client server was detailed and migrating a VM from a thin client server to another one was indicated as the only solution to adapt to this environmental change. It was shown that the implemented prototype can handle up to 700 reported problems per second.

Future research opportunities include extending the Self Management component with additional algorithms to further optimize the mobile thin client service.

VIII. Acknowledgements

Part of the research leading to these results was done for the MobiThin Project and has received funding from the European Community's Seventh Framework (FP7/2007-2013) under grant agreement nr 216946. Lien Deboosere and Bert Vankeirsbilck are funded by a Ph.D grant of the IWT-Vlaanderen. Pieter Simoens is funded by a Ph.D grant of the FWO-V. Filip De Turck is partially funded by postdoctoral grant of the FWO-V.

References

[1] B. Vankeirsbilck, *et al.*, "Bringing Thin Clients to the Mobile World," in *2008 Conference of Network & Electronic Media Towards Future Media Internet*, Saint-Malo, France, October 2008.

[2] A. Moshchuk, S. D. Gribble, and H. M. Levy, "Flashproxy: transparently enabling rich web content via remote execution," in *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*, New York, USA, 2008, pp. 81–93.

[3] J. Kim, R. A. Baratto, and J. Nieh, "pTHINC: a thin-client architecture for mobile wireless web," in *WWW '06: Proceedings of the 15th international conference on World Wide Web*, New York, NY, USA, 2006, pp. 143–152.

[4] Citrix Systems Inc., "Application delivery infrastructure for a dynamic world," http://www.citrix.com.

[5] T. Richardson, *et al.*, "Virtual network computing," *IEEE Internet Computing*, vol. 02, no. 1, pp. 33–38, 1998.

[6] K. L. Baratto R. and J. Nieh, "THINC: A Virtual Display Architecture for Thin-Client Computing," in *the Twentieth ACM Symposium on Operating Systems Principles (SOSP 2005)*, Brighton, United Kingdom, October 23-26 2005.

[7] P. Simoens, *et al.*, "Design and implementation of a hybrid remote display protocol to optimize multimedia experience on thin client devices," in *2008 Australasian Telecommunications Networks and Applications Conference*, Adelaide, Australia, December 2008.

[8] Prologue, "Useit terminal services," http://www.prologue.fr.

[9] B. Vankeirsbilck, *et al.*, "Bandwidth optimization for mobile thin client computing through graphical update caching," in *2008 Australasian Telecommunications Networks and Applications Conference*, Adelaide, Australia, December 2008.

[10] C. Clark, *et al.*, "Live migration of virtual machines," in *Proceedings of the 2nd ACM/USENIX Symposium on Networked Systems Design and Implementation (NSDI)*. Boston, MA, USA: USENIX, the Advanced Computing Systems Association, 2005, pp. 273–286.

[11] P. B. et al., "Xen and the art of virtualization," *SIGOPS Oper. Syst. Rev.*, vol. 37, no. 5, pp. 164–177, 2003.

[12] Qumranet, "Kvm: Kernel-based virtual machine (white paper)," http://www.qumranet.com/art_images/files/8/KVM_Whitepaper.pdf.

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

[14] D. Song, "Z!stream: An application streaming system by copy-on-reference block of executable files," *Distributed Computing and Networking, Lecture Notes in Computer Science*, vol. 4308/2006, pp. 367–372, 2007.

[15] W. Binder and N. Suri, "Green computing: Energy consumption optimized service hosting," in *SOFSEM '09: Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 117–128.

[16] Spirent Communications, "Avalanche 2500," http://www.spirent.com.

[17] F. Travostino, *et al.*, "Seamless live migration of virtual machines over the man/wan," *Future Gener. Comput. Syst.*, vol. 22, no. 8, pp. 901–907, 2006.