Migration approaches to support nomadic services for communities

Roberto Canonico, Vittorio Manetti and Giorgio Ventre
COMICS Lab, Dipartimento di Informatica e Sistemistica
Università di Napoli Federico II, Via Claudio 21, 80125 Napoli, Italy
Email: {rcanonic, vittorio.manetti, giorgio}@unina.it

Abstract— **Research carried out in the context of the CONTENT Network of Excellence aims at investigating novel paradigms for the provision of AV services to user communities by means of self-organizing overlays and hybrid Content Delivery Networks. Being in the age of mobile telecommunications, the CONTENT architectural framework cannot ignore that an increasing number of users require access to such services by means of mobile devices and in mobility. While a considerable effort is now being made to introduce support for mobility in ordinary communication protocols, e.g. by introducing fast and efficient handoff mechanisms, little work has been done to investigate the possibility and the opportunity to migrate services when the users change their connection to the network, e.g. by switching from a GPRS connection to a public WiFi hotspot. In this paper we present a preliminary investigation aimed at comparing different migration mechanisms that can be considered to support service migration in the context of a CONTENT Service Network.**

I. Introduction

The word community comes from the Latin communis, meaning ”common, public, shared by all or many”. A community usually refers to a group of people who interact and share certain things. In the human communities, in which intent, belief, resources, preferences, needs, risks and a number of other conditions may be present and common, affecting the identity of the participants and their degree of adhesion. Effective communication practices in group and organizational settings are important to the formation and maintenance of communities. How ideas and values are communicated within communities are important to the induction of new members, the formulation of agendas, the selection of leaders and many other aspects. Organizational communication is the study of how people communicate within an organizational context and the influences and interactions within organizational structures. Group members depend on the flow of communication to establish their own identity within these structures and learn to function in the group setting. The process of learning to adopt the behavior patterns of the community is called socialization, and the most fertile time of socialization is usually the early stages of life, during which individuals develop the skills and knowledge and learn the roles necessary to function within their culture and social environment. There is a philosophy that is being used as the basis of how various groups and organizations operate; this philosophy is called ”openness”, and it is related to open source. Openness is a relatively new term to describe this general way of doing things, and it is typified by communal management, and open access to the information or material resources needed for projects.

The rapid diffusion of always-on connections such as ADSL, and of mobile communication environments PDA and high-speed wireless LAN or 3G cellular phones, has increased the expectations and demands for real-time applications such as voice phone, TV phone, and instant messengers. Following up on the ubiquity and heterogeneity of networked resources such as devices and applications, the next-generation networking environment, often referred to as a ubiquitous networking environment, has just started to emerge. In this environment, ubiquity enables a user to enjoy various services on the network anytime and anywhere. On the other hand, owing to heterogeneity, the user must select from among many suitable resources according to the situation. However, these resources that were selected by the user might no longer remain optimal as the surrounding situation changes. It is therefore preferable for both service initiation and migration to be provided in an integrated approach for a future ubiquitous networking environment. In the ubiquitous networking environment, various applications on heterogeneous devices are available to the user. It must be possible to establish communication with a call from an arbitrary device selected by the caller. Two fundamental approaches may be used. In the first approach, the same real-time application is installed into all user devices so that two arbitrary devices can be connected with each other. In the second approach, users register their policy and/or preference with the policy server in the network. The registered information is referred to when an incoming call occurs. The call is directed to the appropriate device according to the calls information after being appropriately translated, if necessary. In this second case, it may be occur an event that involves the need to implement a service migration mechanism.

Many aspects regarding community of users, and regarding the ubiquitous networking, can be find again when we deal community of users for contents and services sharing, as well as CONTENT. The CONTENT Network-of-Excellence aims to implement a Content Delivery Network for home users enabling easy-to-install and easy-to-use Audio-Video services in and between homes. The main technical objective is to amplify the potential of European Community Networking by improving Content Distribution infrastructures for the delivery of live content and interactive stored content, and by integrating, tools and mechanisms that would enable the management of multimedia assets and their subsequent access for the benefit of the communities of users, producing a set of appropriate services for them.

The rest of this paper is organized as follows: in section II we introduce the nomadic user concept regarding the ubiquitous networks. The remaining sections are about different kinds of migration mechanisms: in section III, some details about the service migration techniques; in section IV, an introduction to code migration with mobile agents and active networks; in section V we talk about Xen and the live migration of Virtual Machines; section VI is about virtual workspaces in Grid and approaches for migration of virtual execution environments.

II. Nomadic Users

In the ubiquitous networking, the user could be considered like a nomadic one, because it can exploit technologies for wireless connection, like Wi-Fi, bluetooth, Wi-MAX, and it can use mobile devices like PDA and last generation mobile phone. So, it can move itself in an ubiquitous networking environment while it uses the services available in this scenario. Many distributed service models were been defined with the main goal to enable nomadic users to access remote services in a more efficient and convenient way. They should simplify procedures to lookup and select services in mobile wireless environments, and should also enhance the service usefulness by providing flexible mechanisms to adapt to the characteristics of the mobile devices and communication quality. In order to guarantee the user mobility, they have to be solved some problems joined with the coverage. Let us consider the case in which a mobile user is using a WI-Fi connection; it may be possible that an access-point switch has to be implemented to guarantee the connection availability. This goal has to be reached in a very transparent way in regard to the user, and it has to be guaranteed the availability of the service that the user is exploiting during the switch. Many ways can be employed to solve this problem: it can be implemented a dynamic overlay network which can adapt itself to a similar scenario; it can be used service migration techniques; it can be implemented a mechanism by which a new service can be started and provided to the user. Obviously, in an ideal scenario, each of this solutions are available. As far as migration techniques, after a preliminary analysis, we have identified three kinds of mechanisms: code migration, service migration and Virtual Machine migration. In the following sections we will implement an overview on these mechanisms.

III. Mobile services

Short-range wireless technology is on its way of becoming ubiquitous, and it will soon be possible to program real world ad hoc networks. Deploying services in such networks, however, is constrained by the lack of proper service models and system support. Unlike a statically identified service that is always located on the same node, a mobile service can migrate to different nodes in the network to accomplish its task. Although a service end-point can migrate, it constantly appears to the client application as a unique virtual end-point. The service migration occurs transparently to the client application, which is presented with a unique virtual service end-point. In traditional service models, the client can communicate with such a node for the entire duration of the interaction with the service. Such models can hardly support the deployment of services over highly dynamic ad hoc networks, where ad hoc network scenario consists of nodes supporting services that dynamically join and leave the network, thus continuously modifying the set of available services in the region of interaction. Additionally, due to limited resource availability and unpredictable loss of network connectivity, a node may stop providing a service currently in use, and alternatively new services may appear. An interesting solution could be to make services context-aware such that they can adapt to context changes by migrating their execution to nodes where they can accomplish their task better, for example, in order to implement the required Quality of Service. The service is designed such that it is able to learn about context changes and migrate to other nodes in the networks as soon as the current hosting node becomes unsuitable. Additionally, services carry any relevant state information during migrations. Therefore, the interaction between a user and a service can continue uninterrupted, except for small delays generated by migrations. Obviously, according with this approach, there is a need to design more agile services that are able to adapt to changes in the network, or to changes in both the client context and service context. Context includes parameters such as location, speed, time, device capabilities, or network topology.

IV. Mobile Agents and Active Networking

The term agent is used in many research areas, like Artificial Intelligence, robotics, network management, user interfaces and so on. According to the Object Management Group OMG [12], a software agent is defined as an autonomous software entity that can interact with its environment. Agents consist of program code and the associated internal global and execution states and can perform tasks on behalf of their owners. The agents are classified like stationary or mobile. A stationary agent executes only on the system on which it begins execution. If it needs information not on that system or needs to interact with an agent on another system, it typically uses a communication mechanism, such as remote procedure calling. A mobile agent is not bound to the system on which it begins execution. It is free to travel among the hosts in the network, therefor, in the mobile agent contest, the code migration concept covers a fundamental rule. Created in one execution environment, it can transport its state and code with it to another execution environment in the network, where it resumes execution. The term state typically means the attribute values of the agent that help it determine what to do when it resumes execution at its destination. Code in an object-oriented context means the class code necessary for an agent to execute. A mobile agent has the unique ability to transport itself from one system in a network to another in the same network. This ability allows it to move to a system containing an object with which it wants to interact and then to take advantage of being in the same host or network as the object. While in the classical communication paradigms the focus is on the transfer code between components, in the mobile agent paradigm, a whole computational component is moved to a remote site, along with the code it needs, and some resources required to perform the task.

It can be find again the code migration technologies in the active network, where the code is transferred into what we can call an active packet or a capsule. The active network [18] provides a platform on which network services can be experimented with, developed, and deployed. That platform represents a meta-agreement among the parties concerned with the network regarding the fundamental capabilities built into the network, as well as the mechanisms through which they may be accessed and combined to implement new services. The unit of multiplexing of the network is the packet, and the primary function of the active network is communication and not computation. The network contains some nodes, the active nodes, whose primary reason for existence is to switch packets and thus allow sharing of transmission resources. Computation may occur, and indeed computation services could be built upon the active network platform, but the platform itself is not designed to be a general-purpose distributed computing system. Active nodes are interconnected by a variety of packet-forwarding technologies, and this variety can evolve continually. Each active node is controlled by an administration, and no single administration controls all active nodes. Active nodes provide a common base functionality, which is described in part by the node architecture. The node architecture deals with how packets are processed and how local resources are managed, and, on the other hand, it deals with global matters like addressing, end-to-end allocation of resources, and so on. The node architecture is explicitly designed to allow for more than one network API and network architecture. The functionality of the active network node is divided among the Node Operating System (NodeOS), the Execution Environments (EEs), and the Active Applications (AAs). Each EE exports a programming interface or virtual machine that can be programmed or controlled by directing packets to it. An EE provides an interface through which end-to-end network services can be accessed. The architecture allows for multiple EEs to be present on a single active node. The NodeOS provides the basic functionality from which execution environments build the abstractions presented to the active applications. The NodeOS manages the resources of the active node and mediates the demand for those resources, which include transmission, computing, and storage. The NodeOS thus isolates EEs from the details of resource management and from the effects of the behavior of other EEs. Each node has a distinguished management execution environment, through which certain aspects of the local node configuration and policy may be controlled. Two different approach can be used to implement the packet processing in the active networks: discrete approach and integrated approach [15]. With the first kind of mechanism, the processing of messages may be architecturally separated from the business of injecting programs into the node, with a separate mechanism for each function. When a packet arrives, its header is examined and a program is dispatched to operate on its contents. The program actively processes the packet, possibly changing its contents. A degree of customized computation is possible because the header of the message identifies which program should be run, so it is possible to arrange for different programs to be executed for different users or applications. With the integrated approach, every message that passes between nodes contains a program fragment that may include embedded data. When a packet arrives at an active node, its contents are dispatched to a transient execution environment where they can safely be evaluated. The transient environment is destroyed when packet evaluation terminates.

V. Live Migration of Virtual Machines

Modern computers are sufficiently powerful to use virtualization mechanisms to present the illusion of many smaller virtual machines, each running a separate operating system instance. There are two fundamental approaches regard to the virtualization techniques: full virtualization and paravirtualization. In a traditional Virtual Machine Monitor (VMM) obtained by full virtualization mechanisms, the virtual hardware exposed is functionally identical to the underlying machine; paravirtualization consists in presenting a virtual machine abstraction that is similar but not identical to the underlying hardware. This last technique promises improved performance, although it does require modifications to the guest operating system, but no changes to guest applications. Full virtualization has the obvious benefit of allowing unmodified operating systems to be hosted, it also has a number of drawbacks. Xen [5] is a VMM realized at the University of Cambridge, and it is one of the major product used for the operating systems virtualization. Some fundamental features regard to the Xen paravirtualization system: support for unmodified application binaries, support for full multi-application operating systems, completely hiding the effects of resource virtualization from guest OSes. In Xen a single virtual machine hosts a real operating system which may itself securely multiplex thousands of unmodified user-level processes. In Xen each guest OS performs its own paging using its own guaranteed memory reservation and disk allocation, in order to achieve performance isolation; with this approach, no malicious virtual machines can encourage thrashing behavior, unfairly depriving others of CPU time and disk bandwidth. Xen implements a separation between policy and mechanism: in this architecture the hypervisor itself provides only basic control operations, and these are exported through an interface accessible from authorized domains. The hypervisor has not to be involved in higher level issues; policy decisions are performed by management software running over a guest OS rather than in privileged hypervisor code. In Xen a domain is a running virtual machine within which a guest OS executes, and Xen is itself the hypervisor since it operates at a higher privilege level than the supervisor code of the guest operating systems that it hosts. Two mechanisms exist for control interactions between Xen and an overlying domain: synchronous calls from a domain to Xen may be made using a hypercall, while notifications are delivered to domains from Xen using an asynchronous event mechanism. The hypercall interface allows domains to perform a synchronous software trap into the hypervisor to perform a privileged operation, analogous to the use of system calls in conventional operating systems. Communication from Xen to a domain is provided through an asynchronous event mechanism, which replaces the usual delivery mechanisms for device interrupts and allows lightweight notification of important events such as domain-termination requests.

Operating system virtualization has attracted considerable interest in recent years, particularly from the data center and cluster computing communities. The migration concept holds a great role regard this contest. Migrating an entire OS and all of its applications as one unit, allows to avoid many of the difficulties faced by process-level migration approaches, and introduces some benefits in comparison to these last ones. Firstly, while the migration process has been implemented, the original host machine must remain available and network-accessible in order to service certain system calls or even memory accesses on behalf of migrated processes, and the OS migration approach makes it easy to achieve this goal thanks to the interface properties between a virtualized OS and the VMM. The original host may be deactivate once migration has completed. Secondly, the virtual machine migration means that the memory state can be transferred in a consistent and efficient fashion. This applies to kernel-internal state as well as application-level state, even when this is shared between multiple cooperating processes. Regard to the VM migration mechanism, it is critically important to minimize the downtime during which services are entirely unavailable, and to consider the total migration time during which state on both machines is synchronized and which hence may affect reliability. With the migration approach implemented in Xen, this problem is achieved by using a pre-copy [8] approach in which pages of memory are iteratively copied from the source machine to the destination host, all without ever stopping the execution of the virtual machine being migrated. Pagelevel protection hardware is used to ensure a consistent snapshot is transferred, and a rate-adaptive algorithm is used to control the impact of migration traffic on running services. The final phase pauses the virtual machine, copies any remaining pages to the destination, and resumes execution there. No 'pull' approach is used which faults in missing pages across the network since this adds a residual dependency of arbitrarily long duration, as well as providing in general rather poor performance. It is very important to consider the degradation level the migration process involes, in regard to both the services available on the VM we has to migrate, and the services available on the other systems. The migration process exploits the computational and storage resources on the source and destination machines, and it use the network resources in order to carry the VM from one to another. So it involve an overhead increasing which can make a disadvantage to the available services. Two different methods are considered for initiating and managing state transfer: managed migration and self migration. Managed migration is performed by migration daemons running in the management VMs of the source and destination hosts. These are responsible for creating a new VM on the destination machine, and coordinating transfer of live system state over the network. Self migration [9] places the majority of the implementation within the OS being migrated. In this design no modifications are required either to Xen or to the management software running on the source machine, although a migration stub must run on the destination machine to listen for incoming migration requests, create an appropriate empty VM, and receive the migrated system state.

VI. Virtual Workspaces in Grid Computing

Grid [1] is a coordinated resource sharing system implemented by dynamic and multi-institutional virtual organizations (VO). This sharing is highly controlled by resource providers who define the conditions under which sharing occurs. A virtual organization is a set of consumers and institutions defined by such sharing rules. A protocol definition specifies how distributed system elements interact with one another in order to achieve a specified behavior, and the structure of the information exchanged during this interaction. The Grid architecture is organized by layers: the range of resource types is defined at the Fabric layer, and it can be used to construct a wide range of global services and applications at the Collective layer; the protocols defined at the Resource and Connectivity layers are designed to be implemented on top of a diverse range of resource types. The higher, is the Application layer, which comprises the user applications that operate within a VO environment. The Open Grid Service Architecture (OGSA) [2] supports the creation, maintenance, and application of services maintained by VOs. In OGSA, a Grid service is defined as well as a Web service that provides a set of well-defined interfaces and that follows specific conventions. OGSA is an uniform service-oriented model, and this adoption means that all components of the generic environment are virtual. In other words, in OGSA everything is a Grid service. A Grid service implements one or more interfaces, where each interface defines a set of operations that are invoked by exchanging a defined sequence of messages. The service interface definition and access binding are also distinct from the implementation of the functionality of the service. A service can support multiple implementations on different platforms, facilitating seamless overlay not only to native platform facilities but also, via the nesting of service implementations, to virtual ensembles of resources. Grid services can maintain internal state for the lifetime of the service. The existence of state distinguishes one instance (a particular instantiation of a Grid service) of a service from another that provides the same interface.

Grid offers access to diverse software environments, instead, an application typically requires a very specific, customized environment; in other words, users application may in practice be able to use only a small fraction of the resources potentially available on the Grid. The need to provide reliable isolation and dynamic, fine-grain control of shared resources to ensure enforcement of policies, leads to the Virtual Workspace [3], [4] concept. A Virtual Workspace allows a Grid client to define an execution environment in terms of its hardware requirements and software configuration. The main goal of the workspace definition is to capture the requirements for an execution environment in the Grid and then use automated tools in order to find, configure, and provide an environment best matching those requirements. Thus, jobs can be mapped to workspaces, and workspaces can be mapped to actual resources in the Grid. Workspaces implementation is typically realized by the virtual machine technologies. In addition to outstanding isolation properties obtainable by the virtual machines, the virtualization of the underlying hardware enables instantiation of a new, independently configured guest environment on a host resource. The VMs can be rapidly suspended and their state serialized, and thus easily migrated to remote resources, for example, in order to guarantee the Virtual Workspace availability, independently from the real hardware, and in a way that is transparent for the customer. The Virtual Workspace management requires two fundamental entities: VW Repository which provides a management interface to workspaces, and VW Manager which orchestrates their deployment.

VII. Conclusion

In this paper, we presented an overview of migration techniques implemented in different scenarios and their use to support the provision of Audio-Video service to mobile users. We are currently investigating among the above mentioned mechanisms in order to find the more suitable for content distribution services. Even thought it is too early to adopt a final solution, we may say that maybe each of the mechanisms presented in this article can provide us with useful ideas for the future.

References

[1] I. Foster, C. Kesselman and S. Tuecke, *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, International Journal of High Performance Computing Applications, Vol. 15, No. 3, 200-222 (2001).

[2] I. Foster, C. Kesselman, J. Nick and S. Tuecke, *The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration*, Open Grid Service Infrastructure WG, Global Grid Forum, June, 2002.

[3] K. Keahey, I. Foster, T. Freeman, X. Zhang and D. Galron, *Virtual Workspaces in the Grid*, Lecture Notes in Computer Science, Springer, Vol. 3648, Pag. 421-431, August 2005.

[4] K. Keahey, I. Foster, T. Freeman amd X. Zhang, *Virtual workspaces: Achieving quality of service and quality of life in the Grid*, Scientific Programming, IOS Press, Vol. 13, Pag. 265-275, 2005.

[5] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt and A. Warfield, *Xen and the art of virtualization*, in Proceedings of the nineteenth ACM symposium on Operating systems principles, pag. 164 - 177, 2003.

[6] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C Limpach, I. Pratt, A. Warfield, *Live Migration of Virtual Machines*, NCDI '05.

[7] D. Milojicic, F. Douglis, Y. Paindaveine, R. Wheeler and S. Zhou, *Process migration*, ACM Computing Surveys, 32(3):241.299, 2000.

[8] M.M. Theimer, K.A. Lantz and D.R. Cheriton, *Preemptable remote execution facilities for the V-system*, In Proceedings of the tenth ACM Symposium on Operating System Principles, pages 2.12. ACM Press, 1985.

[9] J.G. Hansen and E. Jul, *Self-migration of operating systems*, In Proceedings of the 11th ACM SIGOPS European Workshop (EW 2004), pages 126. 130, 2004.

[10] Z. Wang, J. Seitz, *An Agent-based Distributed Service Model for Nomadic Users*, Eighth International Conference on Parallel and Distributed Systems (ICPADS'01).

[11] W.J. Hwang, *Design and Implementation of Multimedia Service Management Agent on Home Networks Environment*, IJCSNS International Journal of Computer Science and Network Security, VOL.6 No.7B, July 2006.

[12] Object Management Group, *Agent Technology, Green Paper, version 0.92*, OMG Agent Working Group, 25. April 2000.

[13] H. Harroud, M. Ahmed and A. Karmouch, *Policy-Driven Personalized Multimedia Services for Mobile Users*, IEEE Transaction on Mobile Computiong, vol. 2, No 1, January-March 2003.

[14] A. Sashima, N. Izumi, K. Kurumatani and Y. Kotani, *Toward Role-based Agent Coordination for Mobile and Ubiquitous Services*, Proceedings of the 20th International Conference on Advanced Information Networking and Applications (AINA'06), Vol. 2, 2006.

[15] D.L. Tennenhouse, D.J. Wetherall, *Towards an active network architecture*, DARPA Active NEtworks Conference and Exposition, 2002.

[16] M. Solarski, M. Bossardt, T. Becker, *Component-based Deployment and Management of Services in Active Networks*, in Proceedings of IWAN, 2002.

[17] K.L. Calvert, S. Bhattacharjee, E. Zegura and G, Tech, *Directions in Active Networks*, IEEE Communications Magazine, 1998.

[18] K.L. Calvert, *Architectural Framework for Active Networks Version 1.0*, Active Network Working Group Draft, 1999.

[19] N. Laoutaris, G. Smaragdakis, K. Oikonomou, I. Stavrakakis and A. Bestavros, *Distributed Placement of Service Facilities in Large-SCale Networks*, INFOCOM 2006.

[20] N. Imai, M. Isomura and H. Horiuchi, *Flexible and Seamless Service Migration for Real-time Communication with Ubiquitous and Heterogeneous Networked Resources*, IEEE Communications Society, Globecom 2004.

[21] W.J. Hwang, *Design and Implementation of Multimedia Service Management Agent on Home Networks Environment*, IJCSNS International Journal of Computer Science and Network Security, vol.6 No.7B, July 2006.

[22] R. Grimm, *One.world: Experiences with a Pervasive Computing Architecture*, IEEE Pervasive Computing, 2004.

[23] *802.16 IEEE Standard for Local and metropolitan area networks. Part 16: Air Interface for Fixed Broadband Wireless Access Systems*, IEEE Computer Society and IEEE Microwave Theory and Techniques Society.