Cloud@Home: Bridging the Gap between Volunteer and Cloud Computing

Vincenzo D. Cunsolo, Salvatore Distefano, Antonio Puliafito, and Marco Scarpa

University of Messina,
Contrada di Dio, S. Agata, 98166 Messina, Italy
{vdcunsolo,sdistefano,apuliafito,mscarpa}@unime.it

Abstract. The ideas of using geographically distributed resources in a secure way (*Network-Internet/Grid computing*), providing self-management capabilities (*Autonomic computing*), quantifying and billing computing costs (*Utility computing*), in order to perform specific modular applications (*web services*), have been grouped altogether into the concept of *Cloud computing*.

Only commercial Cloud solutions have been implemented so far, offering computing resources and (web) services for renting. Some interesting projects, such as Nimbus, OpenNEbula, Reservoir, work on Cloud. One of their aims is to provide a Cloud infrastructure able to provide and share resources and services for scientific purposes. The encouraging results of Volunteer computing projects such as SETI@home and FOLDING@home and the great flexibility and power of the emergent Cloud technology, suggested us to address our research efforts towards a combined new computing paradigm we named *Cloud@Home*, merging the benefits and overcoming the weaknesses of both the original computing paradigms.

In this paper we present the Cloud@Home paradigm, describing its contribution to the actual state of the art on the topic of distributed and Cloud computing. We thus detail the functional architecture and the core structure implementing such a new paradigm, demonstrating how it is really possible to build up a Cloud@Home infrastructure.

1 Introduction and Motivation

Cloud computing is derived from the *service-centric perspective* that is quickly and widely spreading on the IT world. From this perspective, all capabilities and resources of a Cloud (usually geographically distributed) are provided to users *as a service*, to be accessed through the Internet without any specific knowledge of, expertise with, or control over the underlying technology infrastructure that supports them.

Cloud computing is strictly related to *service oriented science* [1], *service computing* [2] and *IT as a service* (ITAAS) [3], a generic term that includes: platform AAS, software AAS, infrastructure AAS, data AAS , security AAS, business process management AAS and so on.

It offers a user-centric interface that acts as a unique, user friendly, point of access for users' needs and requirements. Moreover, Cloud computing provides *on-demand service provision*, *QoS guaranteed offer*, and *autonomous system* for managing hardware, software and data transparently to users [4].

D.-S. Huang et al. (Eds.): ICIC 2009, LNCS 5754, pp. 423–432, 2009.

In order to achieve such goals it is necessary to implement a level of abstraction of physical resources, uniforming their interfaces and providing means for their management, adaptively to user requirements. This is done through *virtualizations*, *service mashups* (Web 2.0) and *service oriented architectures* (SOA). The development and the success of Cloud computing is due to the maturity reached by such technologies. These factors made realistic the L. Kleinrock outlook of computing as the 5th utility [5], like gas, water, electricity and telephone.

Virtualization allows to execute a software version of a hardware machine into a host system, in an isolated way. It "homogenizes" resources: problems of compatibility are overcome by providing heterogeneous hosts of a distributed computing environment (the Cloud) with the same virtual machine.

The Web 2.0 [6] provides an interesting way to interface Cloud services, implementing service mashup. It is mainly based on an evolution of JavaScript with improved language constructs (late binding, clousers, lambda functions, etc) and AJAX interactions.

The *Service Oriented Architecture* (SOA) is a paradigm that defines standard interfaces and protocols that allow developers to encapsulate information tools as services that clients can access without knowledge of, or control over, their internal workings [1].

A great interest on Cloud computing has been manifested from both academic and private research centers, and numerous projects from industry and academia have been proposed. In commercial contexts, among the others we highlight: Amazon Elastic Compute Cloud, IBMs Blue Cloud, Sun Microsystems Network.com, Microsoft Azure Services Platform, Dell Cloud computing solutions. There are also several scientific activities, such as: Reservoir [7], Nimbus-Stratus-Wispy-Kupa [8] and OpenNEbula [9]. All of them support and provide an on-demand computing paradigm, in the sense that a user submits his/her requests to the Cloud that remotely, in a distributed fashion, processes them and gives back the results. This client-server model well fits aims and scopes of commercial Clouds: the business. But, on the other hand, it represents a restriction for scientific Clouds, that have a view closer to *Volunteer computing*.

Volunteer computing (also called *Peer-to-Peer computing*, *Global computing* or *Public computing*) uses computers volunteered by their owners, as a source of computing power and storage to provide distributed scientific computing [10]. It is behind the *"@home"* philosophy of sharing network connected resources for supporting distributed computing.

We believe the Cloud computing paradigm is applicable also at lower scales, from the single contributing user, that shares his/her desktop, to research groups, public administrations, social communities, small and medium enterprises, which make available their distributed computing resources to the Cloud. Both free sharing and pay-per-use models can be easily adopted in such scenarios.

From the utility point of view, the rise of the "techno-utility complex" and the corresponding increase of computing resources demand, in some cases growing dramatically faster than Moore's Law as predicted by the Sun CTO Greg Papadopoulos in the *red shift theory* for IT [11], could bring, in a close future, towards an *oligarchy*, a lobby or a trust of few big companies controlling the whole computing resources market. To avoid such pessimistic but achievable scenario, we suggest to address the problem in a

different way: instead of building costly private *data centers*, that the Google CEO Eric Schmidt likes to compare to the prohibitively expensive cyclotrons [12], we propose a more "democratic" form of Cloud computing, in which the computing resources of single users accessing the Cloud can be shared with the others, in order to contribute to the elaboration of complex problems.

Since this paradigm is very similar to the Volunteer computing one, it can be named *Cloud@Home*. Both hardware and software compatibility limitations and restrictions of Volunteer computing can be solved in Cloud computing environments, allowing to share both hardware and software resources or *services*.

The Cloud@Home paradigm could be also applied to commercial Clouds, establishing an *open computing-utility market* where users can both buy and sell their services. Since the computing power can be described by a "long-tailed" distribution, in which a high-amplitude population (Cloud providers and commercial data centers) is followed by a low-amplitude population (small data centers and private users) which gradually "tails off" asymptotically, Cloud@Home can catch the *Long Tail* effect [13], providing similar or higher computing capabilities than commercial providers' data centers, by grouping small computing resources from many single contributors.

In the following we demonstrate how it is possible to make real all these aims through the Cloud@Home paradigm. Thus, in section 2 we describe the functional architecture of the Cloud@Home infrastructure, and in section 3 we characterize the blocks implementing the functions previously identified into the Cloud@Home core structure. With section 4 we conclude the paper recapitulating our work and discussing about challenges and future work.

2 Cloud@Home Overview

A possible Cloud@Home architecture is shown in Fig. 1, identifying three hierarchical layers: *frontend*, *virtual* and *physical*. A user can interact with the Cloud through the *consumer host* after authenticating him/herself into the system. The main enhancement of Cloud@Home is that a host can be at the same time both contributing and consumer host, establishing a symbiotic mutual interaction with the Cloud.

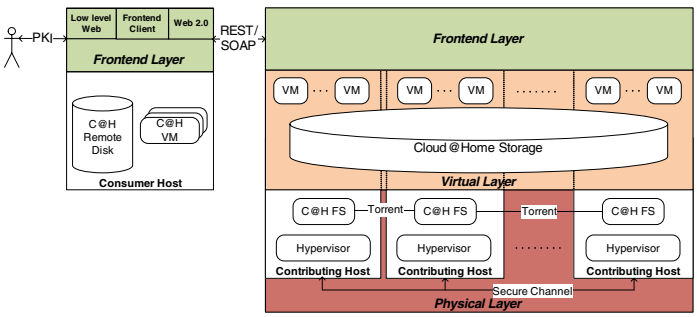

Fig. 1. Basic architecture of Cloud@Home

2.1 Frontend Layer

The Cloud@Home frontend layer is responsible for the resources and services management (enrolling, discovery, allocation, coordination, monitoring, scheduling, etc) from the global Cloud system's perspective. The frontend layer provides tools for translating end-user requirements into physical resources' demand, also considering QoS/SLA constraints, if specified by the user. Moreover, in commercial Clouds, it must be able to negotiate the QoS policy to be applied (SLA), therefore monitoring for its fulfillment and, in case of unsatisfactory results, adapting the computing workflow to such QoS requirements.

If the available Cloud's resources and services can not satisfy the requirements, the frontend layer provides mechanisms for requesting further resources and services to other Clouds, both open and/or commercial. In other words, the Cloud@Home frontend layer implements the interoperability among Clouds, also checking for services' reliability and availability. In order to improve reliability and availability of services and resources, especially if QoS policies and constraints have been specified, it is necessary introduce redundancy.

The frontend layer is split into two parts, as shown in Fig. 1: the server side, implementing the resources management and related problems, and the *light* client side, only providing mechanisms and tools for authenticating, accessing and interacting with the Cloud.

In a widely distributed system, globally spread around the world, the knowledge of resources' accesses and uses assumes great importance. To access and use the Cloud services a user first authenticates him/herself and then specifies whether he/she wants to make available his/her resources and services for sharing, or he/she only uses the Cloud resources for computing. The frontend layer provides means, tools and policies for managing users. The best mechanism to achieve secure authentications is the *Public Keys Infrastructure* (PKI) [14], better if combined with smartcard devices that, through a trusted certification authority, ensure the user identification.

Referring to Fig. 1, three alternative solutions can be offered by the frontend layer for accessing a Cloud: a) Cloud@Home frontend client, b) Web 2.0 user interface and c) low level Web interface (directly specifying REST or SOAP queries). These also provide mechanisms for customizing user applications by composing services (service mashup and SOA) and submitting own services.

2.2 Virtual Layer

The virtualization of physical resources offers end-users a homogeneous view of Cloud's services and resources. Two basic services are provided by the virtual layer to the frontend layer and, consequently, to the end-user: *execution* and *storage* services. The execution service is the tool provided by the virtual layer for creating and managing virtual machines. A user, sharing his/her resources within a Cloud@Home, allows the other users of the Cloud to execute and manage virtual machines locally at his/her node, according to policies and constraints negotiated and monitored at the frontend layer. In this way, a Cloud of virtual machine's executors is established, where virtual machines can migrate or can be replicated in order to achieve reliability, availability and

QoS targets. As shown in Fig. 1, from the end-user point of view an execution Cloud is seen as a set of virtual machines available and ready-to-use. The virtual machines' *isolation* implements protection and therefore security. This security is ensured by the hypervisor that runs the virtual machine's code in an isolated scope, similarly to a sandbox, without affecting the local host environment. The storage service implements a storage system distributed across the storage hardware resources composing the Cloud, highly independent of them since data and files are replicated according to QoS policies and requirements to be satisfied. From the end-user point of view, a storage Cloud appears as a locally mounted remote disk, similarly to a Network File System or a Network Storage. The tools, libraries and API for interfacing end-user and storage Clouds are provided to user by the frontend client, but are implemented at virtual and physical layers.

2.3 Physical Layer

The physical layer is composed of a *"cloud"* of generic nodes and/or devices geographically distributed across the Internet. They provide to the upper virtual layer both physical resources for implementing execution and storage services and mechanisms and tools for locally managing such resources. Cloud@Home negotiates with users that want to join a Cloud about his/her contribution. This mechanism involves the physical layer that provides tools for reserving physical execution and/or storage resources for the Cloud, and monitors these resources, such that constraints, requirements and policies thus specified are not violated. This ensures reliability and availability of physical resources, avoiding to overload the local system and therefore reducing the risk of crashes.

To implement the execution service in a generic device or to enroll it into an execution Cloud, the device must have a hypervisor ready to allocate and run virtual machines, as shown in Fig. 1. If a storage service is installed into the device, a portion of the local storage system must be dedicated for hosting the Cloud data. In such cases, the Cloud@Home file system has to be installed into the devices' shared storage space.

At physical layer it is necessary to implement data security (integrity and confidentiality) also ensuring that stored data cannot be accessed by who physically hosts them. We propose to combine the inviolability of the asymmetric cryptography and the performance of the symmetric one: data are firstly encrypted by the symmetric key, and then stored into the selected host with the symmetric key encrypted by the user private key. Since the data stored in a Cloud@Home storage are encrypted, a higher performance protocol, such as BitTorrent [15] can be used for transferring data. A secure channel such as SSH, TLS, IPSEC and the like is required for sending and receiving non-encrypted messages and data to/from remote hosts.

3 Cloud@Home Core Architecture

Once the functional architecture of Cloud@Home has been introduced, in Fig. 2 we characterize the blocks implementing the functions thus identified, the core structure of the overall system implementing the Cloud@Home server-side, subdivided into *management* and *resource subsystems*.

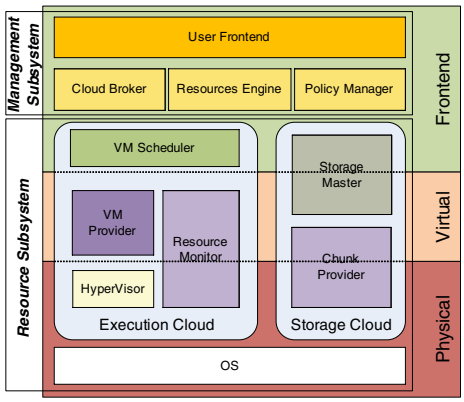

Fig. 2. Cloud@home Core Structure Organization

3.1 Management Subsystem

In order to enroll and manage the distributed resources and services of a Cloud, providing a unique point of access them, it is necessary to adopt a centralized approach that is implemented by the management subsystem. It is composed of four parts: the *user frontend* (UF), the *Cloud broker*, the *resource engine* and the *policy manager*.

The user frontend provides tools for Cloud@Home-User interactions. Incoming requests are transferred to the blocks composing the underlying layer (resource engine, Cloud broker and policy manager) for processing. An important task carried out by the user frontend is the *Clouds interoperability*, implemented by point-to-point connecting the user frontend of the Clouds wishing to interoperate. In case one of the Clouds does not have the Cloud@Home core structure of Fig. 2, it is necessary to translate the requests between Cloud@Home and foreign Clouds formats, task delegated by the user frontend to the Cloud broker. The Cloud broker collects and manages information about the available Clouds and the services they provide (both *functional* and *non-functional* parameters, such as QoS, costs, reliability, *request formats' specifications* for Cloud@Home-foreign Clouds translations, etc).

The policy manager provides and implements the Cloud's access facilities. This task falls into the security scope of identification, authentication, identity and permission management. To achieve this target, the policy manager uses an infrastructure based on PKI, smartcard devices and Certification Authority. The policy manager also manages the information about users' QoS policies and requirements.

The resource engine is the hearth of Cloud@Home. It is responsible for the resources' management, the equivalent of a Grid *resource broker* in a broader Cloud environment. To meet this goal, the resource engine applies a hierarchical policy. It operates at higher level, in a centralized way, indexing all the resources of the Cloud. Incoming requests are delegated to *VM schedulers* or *storage masters* that, in a distributed fashion, manage the computing or storage resources respectively, coordinated by the resource engine.

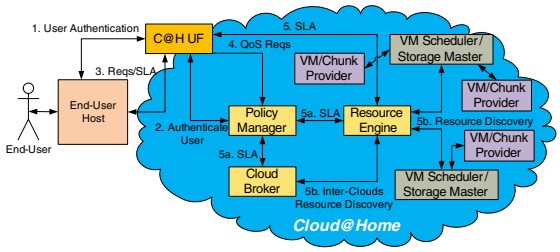

Fig. 3. Cloud@Home End-User Negotiation

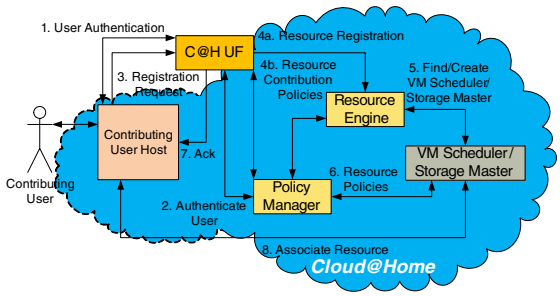

Fig. 4. Cloud@Home Resource Setup

In order to manage QoS policies and to perform the resources discovery, the resource engine collaborates with both Cloud broker and policy manager, as depicted in Fig. 3 showing the step-by-step interactions among such blocks. After authenticating into the system (steps 1 and 2), an end-user specifies his/her requirements (step 3), saved by the policy manager (step 4). Then, a negotiation between the two parties is triggered (step 5), iteratively interacting with the end-user till an agreement is met (SLA). This task is split into two parallel subtasks: the former (step 5a), performed by the policy manager under the supervision of the resource engine, estimates and evaluates the QoS requirements of the request; the latter (step 5b), performed by the resource engine, discovers resources and services to be used. Both subtasks can require the collaboration of the Cloud broker, that looks for other Clouds able to provide resources and services to satisfy SLA/QoS requirements.

Fig. 4 shows the interaction between a contributing user, that wants to provide his/her resources to a Cloud, and the Cloud@Home management system. A user, authenticated by the Cloud's policy manager (steps 1 and 2), sends a request for registering resources and services to the user frontend (step 3), also specifying policies for using them. It sorts the request at the resource engine (step 4a), and constraints and policies at the policy manager (step 4b). After that, the resource engine searches for a VM scheduler or a storage master to which such resources/services have to be assigned (step 5), collaborating with the policy manager. It can also create a new VM scheduler/storage master if the search results obtained do not satisfy the requirements.

Once the scheduler/master is identified the policy manager contacts it for exchanging policies and specifications of the resources. Then the resource engine sends the acknowledgement and the scheduler/master reference to the contributing host (step 7), that signals its availability and actual status (step 8).

3.2 Resource Subsystem

The resource subsystem contains all the blocks implementing the local and distributed management functionalities of Cloud@Home. This subsystem can be logically split into two parts according to the service offered: the *execution Cloud* and the *storage Cloud*. The management subsystem merges them providing a unique Cloud that can offer both execution and/or storage services. The execution Cloud provides tools for managing virtual machines according to users' requests and requirements coming from the management subsystem. It is composed of four blocks: *VM scheduler*, *VM provider*, *resource monitor* and *hypervisor*.

The VM Scheduler is a peripheral resource broker of the Cloud@Home infrastructure, to which the resource engine delegates the management of computing/execution resources and services of the Cloud. It establishes what, where, when and how allocate a VM, moreover it is responsible for moving and managing VM services. From the end-user point of view a VM is allocated somewhere on the Cloud, therefore its migration is transparent for the end-user that is not aware of any VM migration mechanism. The association between resources and scheduler is made locally, as shown in Fig. 4. Since a scheduler can become a bottleneck if the system grows, to avoid the congestion further decentralized and distributed scheduling algorithms (hierarchical, with replication, autonomic, etc) can be implemented.

The VM provider, the resource monitor and the hypervisor are responsible for managing a VM locally to a physical resource. A VM provider exports functions for allocating, managing, migrating and destroying a virtual machine on the corresponding host. The resource monitor allows to take under control the local computing resources, according to requirements and constraints negotiated in the setup phase with the contributing user. If during a virtual machine execution local resources crashes or becomes insufficient to keep running the virtual machine, the resource monitor asks the scheduler to migrate the VM elsewhere.

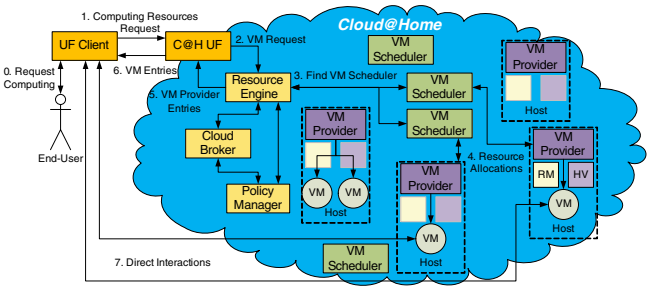

Fig. 5. User Computing Request Processing

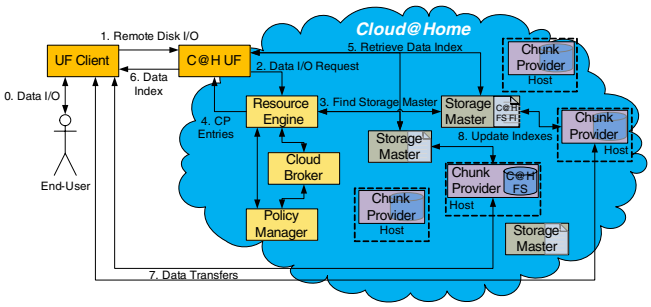

Fig. 6. User Remote Disk I/O Request Processing

Fig. 5 depicts the process of requesting and allocating computing resources in Cloud@Home environments. The overall process is coordinated by the resource engine that estimates requests and requirements submitted by the end-user (steps 0,1 and 2), and therefore evaluates and selects proper schedulers (step 3). Each of such schedulers, in its turn, allocates the physical resources that will host the VM (step 4). The access points of such resources are then fed back to the end-user (steps 5 and 6), and consequently the two parties get connected and can directly interact (step 7).

In order to implement the storage Cloud, we specify the *Cloud@Home file system* (FS) splitting data and files into *chunks* of fixed or variable size, depending on the storage resource available. The architecture of such file system is hierarchical: data chunks are physically stored on *chunk providers* and corresponding *storage masters* index the chunks through specific *file indexes* (FI). The storage master directly interfaces with the resource engine to discover the resources storing data. To improve the storage Cloud reliability, storage masters have to be replicated. Moreover, a chunk provider can be associated to more than one storage master. In order to avoid a storage master becoming a bottleneck, once the chunk providers have been located, data transfers are implemented by directly connecting end-users and chunk providers.

Chunk providers physically store the data, that, as introduced above, are encrypted in order to achieve the confidentiality goal. Data reliability can be improved by replicating data chunks and chunk providers, consequently updating the corresponding storage masters. In this way, a corrupted data chunk can be automatically recovered and restored through the storage masters, without involving the end-user. Similarly to the execution Cloud, the storage Cloud can be implemented as shown in Fig. 6: an end-user data I/O request to the Cloud (steps 0 and 1) is delivered to the resource engine (step 2), that locates the storage masters managing the chunk providers where data are stored or will be stored (step 3), and feeds back the list of chunk providers and data indexes to the end-user (step 4, 5 and 6). In this way the end user can directly interact with the assigned chunk providers storing his/her data (step 7).

4 Conclusions

In this paper we proposed an innovative computing paradigm merging volunteer contributing and Cloud approaches into Cloud@Home. This proposal represents a solution

for building Clouds, starting from heterogeneous and independent nodes, not specifically conceived for this purpose.

In this way Cloud@Home opens the Cloud computing world to scientific and academic research centers, as well as to communities or single users: anyone can voluntarily support projects by sharing his/her resources. On the other hand, it opens the utility computing market to the single user that wants to sell his/her computing resources. To realize this broader vision, several issues must be adequately taken into account: reliability, security, portability of resources and services, interoperability among Clouds, QoS/SLA and business models and policies.

References

1. Foster, I.: Service-oriented science. Science 308(5723) (May 2005)
2. Zhang, L.J.L.: EIC Editorial: Introduction to the Body of Knowledge Areas of Services Computing. IEEE Transactions on Services Computing 1(2), 62–74 (2008)
3. Foster, I., Tuecke, S.: Describing the Elephant: The Different Faces of IT as Service. Queue 3(6), 26–29 (2005)
4. Wang, L., Tao, J., Kunze, M., Castellanos, A.C., Kramer, D., Karl, W.: Scientific Cloud Computing: Early Definition and Experience. In: HPCC 2008, pp. 825–830 (2008)
5. Kleinrock, L.: A vision for the internet. ST Journal of Research 2(1), 4–5 (2005)
6. Tim O'Reilly: What is WEB 2.0 (September 2005), http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html
7. Reservoir Consortium: Reservoir Project (2009), http://www-03.ibm.com/press/us/en/pressrelease/23448.wss/
8. University of Chicago-University of Florida-Purdue University-Masaryk University: Nimbus-Stratus-Wispy-Kupa Projects (January 2009), http://workspace.globus.org/clouds/nimbus.html/, http://www.acis.ufl.edu/vws/, http://www.rcac.purdue.edu/teragrid/resources/#wispy, http://meta.cesnet.cz/cms/opencms/en/docs/clouds
9. Distributed Systems Architecture Research Group: OpenNEbula Project [URL]. Universidad Complutense de Madrid (2009), http://www.opennebula.org/
10. Anderson, D.P., Fedak, G.: The computational and storage potential of volunteer computing. In: CCGRID 2006, pp. 73–80 (2006)
11. Martin, R.: The Red Shift Theory. InformationWeek (August 20, 2007), http://www.informationweek.com/news/hardware/showArticle.jhtml?articleID=201800873
12. Baker, S.: Google and the Wisdom of Clouds. BusinessWeek (December 24, 2008), http://www.businessweek.com/magazine/content/07_52/b4064048925836.htm
13. Anderson, C.: The Long Tail: How Endless Choice Is Creating Unlimited Demand. Random House Business Books (July 2006)
14. Tuecke, S., Welch, V., Engert, D., Pearlman, L., Thompson, M.: Internet X.509 Public Key Infrastructure (PKI) Proxy Certificate Profile. RFC 3820 (Proposed Standard) (June 2004)
15. Cohen, B.: The BitTorrent Protocol Specification (2008), http://www.bittorrent.org/beps/bep_0003.html