# Peer-to-Peer Cloud Computing

*Ozalp Babaoglu*
*Moreno Marzolla*

Department of Computer Science and Engineering
University of Bologna, Italy

Cloud computing is perhaps the most disruptive technological development to have come along in information technology in some time. In little over a decade, cloud technology has revolutionized the path from idea to product launch, especially for Internet startups. In the pre-cloud era, any startup anxious to deploy its "next big thing" on the Internet had to make significant investments for physical real-estate, hardware equipment, network connectivity and technical personnel to keep it all functioning. Even if the startup could sustain the financial burden of building and maintaining a complex in-house computing infrastructure, the inevitable delays in launching its product could nullify any competitive edge it may have had.

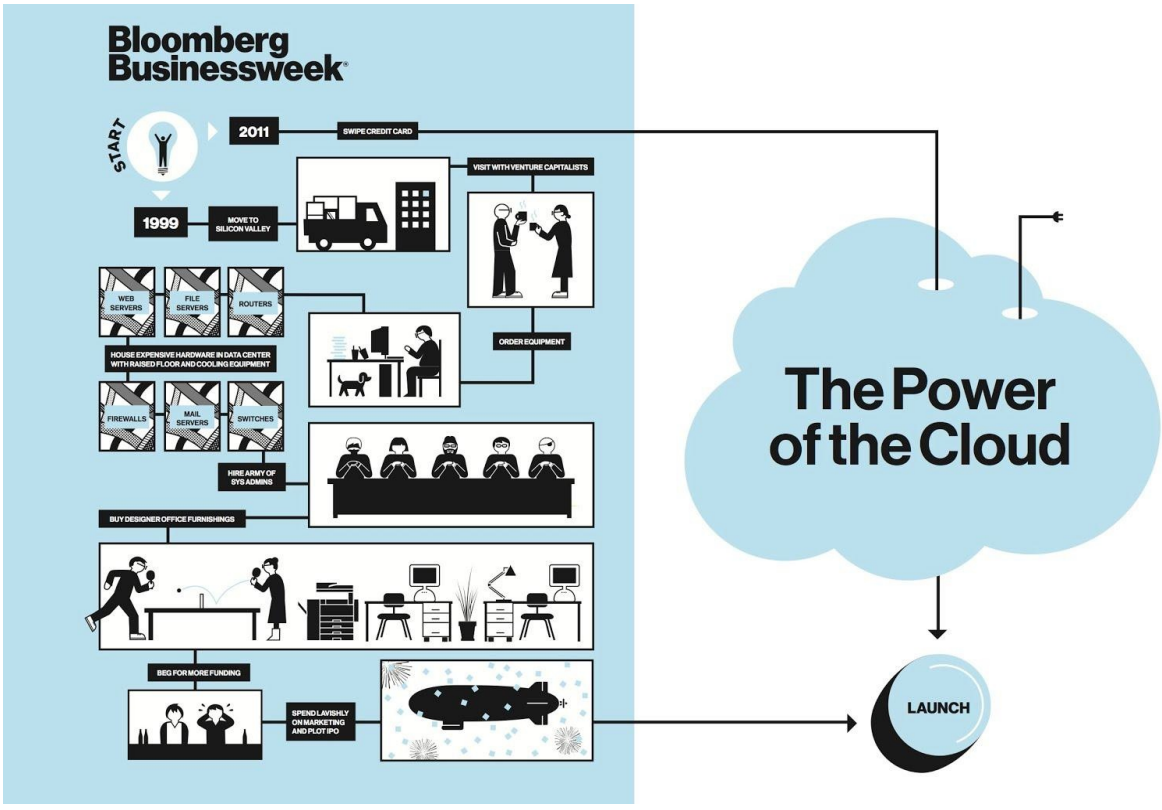


**Figure 1**. Pre-cloud (1999) and post-cloud (2011) paths to product launch for a startup.
(*Bloomberg Businessweek Magazine*, March 3, 2011 )

Today, the same startup could have its product up and running "on the cloud" in a matter of days if not hours with zero up-front investment for infrastructure. All it has to do is purchase the appropriate infrastructure from a commercial "Cloud Service Provider" (this itself could be done online using a credit card) and deploy its product over it. Furthermore, the "pay-per-use" cost model adopted by most cloud service providers means that the purchased infrastructure need not be statically provisioned for

anticipated peak demand. Instead, the "elasticity" feature of the cloud will dimension resources dynamically according to actual demand and the startup will pay only for what it actually uses. With the computing infrastructure out-of-sight and out-of-mind, the startup can concentrate its resources to innovating and improving its product and to maintaining its competitiveness. In short, cloud computing empowers innovative startups by allowing anyone with an Internet connection and a credit card to tap the same world-class computing resources as a major corporation. It comes as no surprise that many of the most popular and successful Internet services today, including Netflix, Instagram, Vine, Foursquare, Farmville and Dropbox, are all deployed over commercial clouds.

**Figure 2**. A server farm.

**Figure 3**. A data-center.

Current cloud computing infrastructures evoke images of huge server farms with racks full of processors, storage and communication hardware (Fig. 2) housed in stadium-size data-centers (Fig. 3) with adjacent or nearby power and cooling stations that can be equally large. For the cloud service provider, these centralized architectures not only require a huge initial investment for real-estate and hardware, they also incur significant operating expenses in the form of power, cooling and technical personnel. As such, this form of cloud computing infrastructure has been a viable option only for giant corporations the likes of Amazon, Google and Microsoft.

Are there alternatives to this centralized version of a cloud computing infrastructure that can lower the startup and operating costs to levels affordable by small businesses or even individuals? In this article we explore the technical challenges that need to be addressed in order to respond affirmatively to this question and show how existing peer-to-peer technologies can be employed to meet the challenges. The resulting architecture, which is totally decentralized and can be built on top of existing commodity computing, storage and communication resources with essentially zero initial investment, can be seen as a "democratization" of cloud computing for the masses.

# Terminology

The National Institute of Standards and Technology defines cloud computing as follows:

> *Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction*.

There are four key attributes implied in this definition:

- *on-demand self-service*: resources such as servers, storage, communication and applications can

be acquired and released dynamically, when required and directly by the customer;

- *network access*: resources can be accessed through the network using standard protocols and well defined interfaces;
- *elasticity*: the customer perceives the cloud as a potentially "unlimited" shared pool of resources that can be purchased in any quantity at any time;
- *measured service*: cloud resource and service usages are billed through a pay-per-use model.

Given this broad definition of cloud computing, various *service models* have been proposed to distinguish the different levels of abstraction at which a customer may interact with a cloud environment. These are the *Software as a Service* (SaaS), *Platform as a Service* (PaaS), and *Infrastructure as a Service* (IaaS) models. The different service models can be seen as forming a stack on top of the hardware base as depicted in Fig. 4.

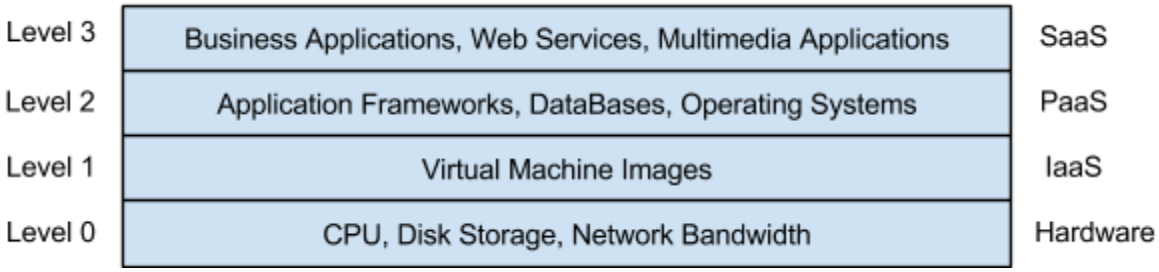


**Figure 4**. Cloud service models.

Level 0 is the collection of physical resources within the data-center implementing the cloud and it is not exposed to customers directly. The lowest level of the service model, IaaS, provides each customer with her private copy of the physical infrastructure through virtualization technologies. In addition to virtualized bare machines, an IaaS cloud service provider may include a choice of host operating systems to run on them. Notable examples of IaaS clouds include Google's Compute Engine and Amazon's Elastic Compute Cloud (EC2). A PaaS cloud includes an application development environment in addition to the underlying infrastructure. Customers develop applications using libraries, APIs and software tools such as databases and middleware provided perhaps by third parties. Examples of PaaS solutions are AppEngine by Google, Force.com from SalesForce, Microsoft's Azure and Amazon's Elastic Beanstalk. Finally, in a SaaS cloud, customers are provided complete application suites for a particular domain and do not interact with the infrastructure itself. As individuals, this is probably the model we are most familiar with when we use cloud-based email or document collaboration services like Gmail and Google Docs. Other examples include SalesForce Customer Relationship Management, SAP Financial Services Network, Microsoft Office 365 and Apple iWorks.

An additional taxonomy is possible by considering the entity who owns and/or operates the cloud infrastructure. In this case, we have the *Private Cloud*, the *Public Cloud*, and the *Hybrid Cloud* models. A private cloud infrastructure is operated exclusively for a specific customer organization, but not necessarily by the organization itself. In the public cloud model, the infrastructure is owned and operated by a cloud service provider and is made available to everyone. The hybrid cloud model refers to cloud infrastructures that have both private and public components, as might be in the case of an organization that runs a private cloud for its critical operations but relies on a public cloud for archiving non-sensitive data.

## The Case for Peer-to-Peer Clouds

Building a cloud infrastructure as a single, massive data-center has certain advantages. Construction, equipment procurement, installation and maintenance are typically simplified since everything is located in one place. Economies of scale allow cost reductions so that a single giant installation is usually cheaper than multiple smaller ones. On the other hand, running a single large data-center creates numerous challenges. The extreme density with which hardware is packed in a confined physical space generates overall power requirements comparable to a small town. And the heat that is generated has to be dissipated somehow. It is no wonder that locations in cold climates and those close to sources of cheap electrical power generation are favored as potential data-center sites.

Perhaps the most serious shortcoming of a centralized cloud (Fig. 5(a)) is that it represents a single-point-of-failure, regardless how cleverly it is designed. Redundant power supplies, backup power generators or replicated network connections cannot provide absolute protection against catastrophic events such as fires, hurricanes, earthquakes or floods. Unlikely as these events may seem, they do happen and numerous incidents have been reported where entire data-centers along with all of the cloud services hosted on them have been knocked out by mild weather storms. Another issue with centralized clouds arises from their geographical location as what is best for the cloud service provider may not be best for the customer. This is the case, for example, when governments place restrictions on sensitive data crossing national borders. A data-center located in a foreign country may shut out numerous domestic customers.

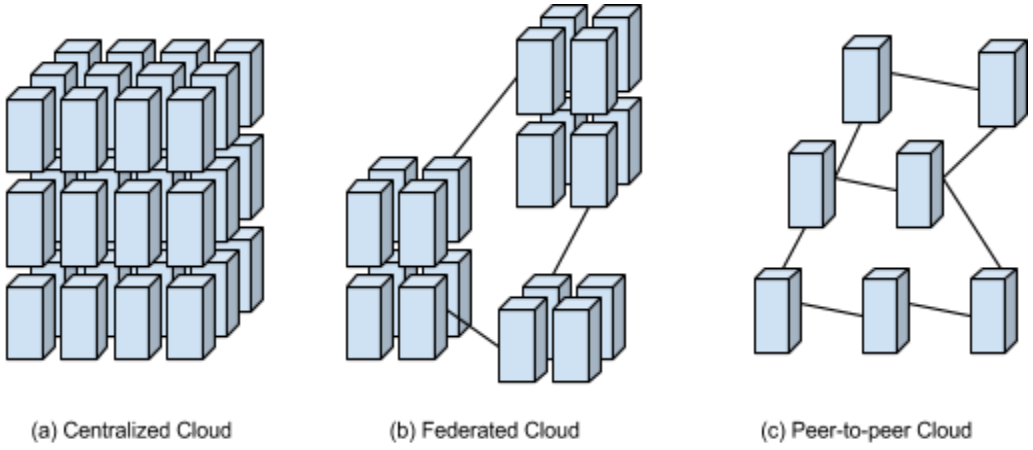


**Figure 5**. Centralized, Federated and Peer-to-Peer Cloud models.

Cloud service providers have addressed the last two concerns by moving their infrastructure towards a federated model (Fig. 5(b)): multiple data-centers are created at distant geographical locations interconnected through fast, private networks. Doing so not only increases the independence of failure modes even in the face of catastrophes, it also provides customers with more options for data location consistent with expected access patterns and legal concerns.

If we continue the geographical distribution initiated by federated clouds to its logical conclusion, we could end up with millions of individual nodes distributed across the globe and interconnected through a public communication fabric like the Internet. We call the resulting architecture a *Peer-to-Peer (P2P)*

*Cloud* (Fig. 5(c)) since it shares many of the characteristic of other P2P systems developed for file sharing, content distribution, and more recently made popular by virtual cryptocurrency schemes such as Bitcoin as their payment networks. In principle, a P2P cloud could be built out of commodity computing, storage and communication resources found in many private homes with essentially zero initial investment. Broadband modems, routers, set-top boxes, game consoles and desktop PCs are all candidates for participation in this architecture since today most of these devices already provide Internet connectivity, reasonable computation and some storage capabilities. The challenge is to turn this collection of loosely connected, unreliable, heterogeneous, resource-constrained hardware into a coherent and usable infrastructure through existing P2P technologies so as to offer an IaaS cloud API to customers. We must also ensure that the salient features of clouds — on-demand resource provisioning, elasticity and measured service — are maintained.

The completely decentralized architecture of a P2P cloud offers many desirable properties. First, there is no single entity that owns or controls it. As with most other P2P applications, a P2P cloud could be created and operated as a grassroots effort without requiring the permission or consent of any authority. Individuals decide unilaterally to participate in a P2P cloud by installing the appropriate client software on their local machines. The "value" of the resulting P2P cloud infrastructure will be commensurate with the number of individuals who have joined it. Just as no single entity is controls the creation of a P2P cloud, no single entity can unilaterally decide to shut it down. A second advantage of a P2P cloud derives from the fact that its components are small, have low power consumption and are geographically distributed. In other words, power to a P2P cloud is supplied through different power grids and network connectivity is provided through different ISPs, reducing drastically the single point-of-failure risk associated with data-centers. Furthermore, the low-power nature of the components and the significant geographic separation between them obliviates completely the heat dissipation concerns of data-centers.

While P2P clouds are unlikely to provide the Quality of Service (QoS) guarantees of centralized or federated clouds, there are nevertheless certain usage scenarios for which a fully distributed cloud architecture can be useful. Applications that can benefit from a P2P cloud include embarrassingly parallel computations, multimedia streaming, online gaming with low-latency and high-interactivity requirements and collaborative tools with shared data. By scaling down the geographic distribution from global to a single organization, we can also imagine P2P clouds constructed out of idle in-house resources. For example, an engineering company could construct a P2P cloud out of its desktop PCs during off-load hours and make it available to the design department for structural simulations, the IT group to analyze network access logs for intrusion detection, and the accounting department to compute cash flow and other financial indicators for evaluation purposes.

## Volunteer Computing

The idea of large-scale computing infrastructures built out of loosely-coupled nodes is not new. *Volunteer Computing* (VC) is a well known computing paradigm where users execute applications on behalf of others on their personal machines. VC systems usually require users to install a specific application on their PC; when the machine becomes idle (i.e., the screensaver would normally kick in) the application fetches and processes input data from a central location and uploads to it the results. This "master-slave" architecture is appropriate for many scientific computations where a central node can "farm out" pieces of computation to workers that can proceed independently and in parallel. If a worker fails to return a result within some reasonable time period, the same work is simply handed out to some other worker. If a worker returns a result, it is assumed to be correct.

The BOINC system [BOINC] from Berkeley is a popular VC container that can load and run different

client programs that participate in different computations. Examples of projects running on the BOINC platform are SETI@home (analysis of radio signals from space to detect potential extra-terrestrial transmissions), Folding@home (protein folding), Einstein@home (gravitational wave detection) and many others. *Desktop Grids* are yet another example of VC systems. A desktop grid is a bridge that allows users to contribute processing power on their PCs to conventional Grid computing infrastructures. BOINC has a specific application for supporting desktop grids; other systems such as EDGeS (Enabling Desktop Grids for e-Science) [EDGES] and XtremWEB exist specifically for implementing desktop grids. We could also characterize the "pooled mining" approach to creating bitcoins as a VC where groups of users pool their computational resources to solve smaller crypto challenges and divide the earned bitcoins fairly among them.

Whereas cloud computing has been conceived as a general platform for arbitrary computing needs, VC is suitable for users to contribute their spare computing resources to specific scientific projects, but not available for general computations. As such, VC systems do not include a full-fledged virtualization environment but a much more limited application container (as in the case of BOINC). Being entirely volunteer based, VC systems have no cost model comparable to cloud computing's pay-per-use.

## Challenges for P2P Clouds

The extreme scale that a P2P cloud could reach, both in the number of components and in terms of geographic distribution, will mean that failures will be common place. To aggravate the situation further, individuals who operate the devices out of their homes could decide to turn on/off or plug/unplug them at will. The resulting dynamic, whereby components are constantly joining and leaving the system, is referred to as *churn* in the P2P literature.

The first challenge for P2P clouds can be stated as: *how can we create and maintain a coherent computing infrastructure from a multitude of unreliable resources and multiplex it among different users even in the presence of sustained rates of churn*? This requires keeping track of all functioning and online devices, dynamically partitioning them among customers and reclaiming them when finished. And all of this has to be done in a completely decentralized manner with no master or controller and despite churn. As it turns out, these challenges have been encountered in other P2P systems and efficient *gossip-based* protocols have been developed to solve them. Gossipping is a very simple interaction paradigm where peers in a large, unstructured network exchange information with a small number of their neighbors, possibly updating their internal state based on the outcome of such interactions. Gossip-based protocols have been extensively studied and have been used to model a diverse set of functions including the spreading of malware in a computer network, diffusion of information in a social network and synchronization of light pulses in a swarm of fireflies. Gossip-based protocols are appealing for P2P clouds because they are extremely simple to describe and implement, yet they can realize complex global computations out of simple local interactions efficiently and quickly despite the possibility of churn.

In a prototype P2P cloud system built at the University of Bologna, we have used completely decentralized gossip-based protocols extensively for implementing different functionality necessary for P2P clouds including membership (figuring out who is up and connected), counting (useful for cloud monitoring), slicing (partitioning a cloud into multiple sub-clouds), slice merging (useful for elastic resource allocation) and for supporting complex queries over the set of peers (for example to identify the top 10% most reliable nodes) (Figs. 6 and 7).

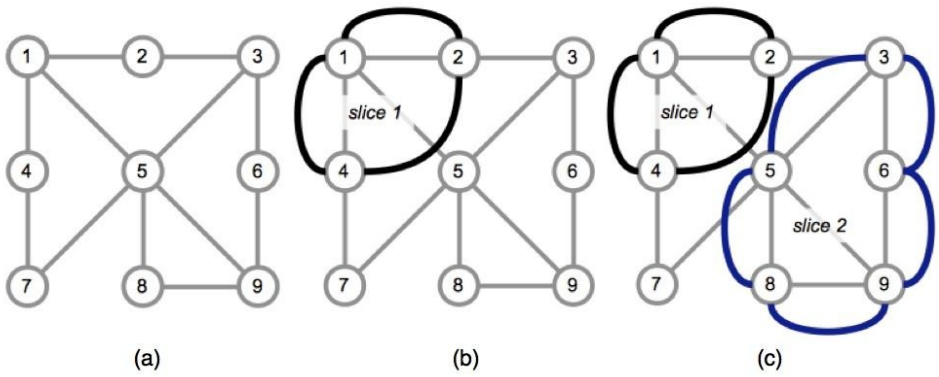


**Figure 6**. (a) Gossip-based membership protocol builds an unstructured connected network containing all available nodes. (b) A ring-structured first slice consisting of nodes 1, 2 and 4 is allocated to a customer. (c) A second slice is allocated to another customer as a disjoint ring consisting of nodes 3, 5, 6, 8 and 9.

For the next set of challenges presented by P2P clouds, we need to distinguish between two different scenarios regarding ownership. In scenario A, all of the devices participating in the system are owned by a single organization and are deployed in individual's homes as might be in the case of broadband modems or routers operated by an Internet Service Provider or set-top boxes operated by a cable television company. In scenario B, on the other hand, all of the devices are owned and operated by individuals in their homes, as might be in the case of game consoles or desktop PCs.

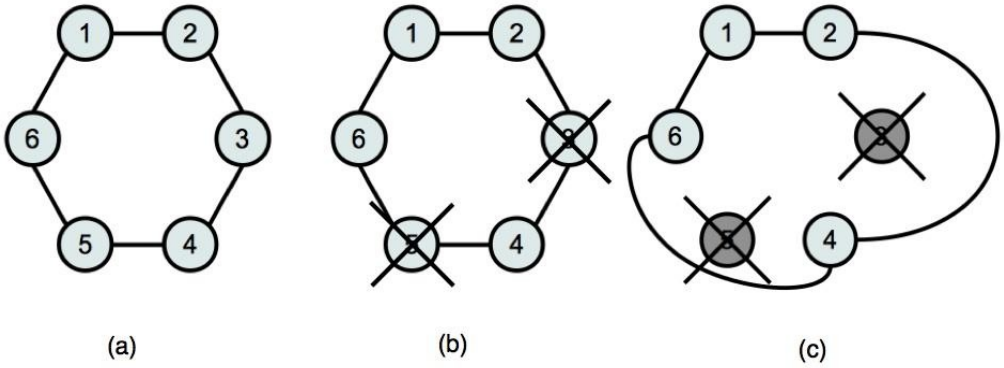


**Figure 7**. (a) Initial slice as a ring of six nodes. (b) Due to churn, nodes 3 and 5 leave the system, breaking the ring structure. (c) The same gossip-based protocol used to construct the ring is able to repair it by passing over the missing nodes and reconstructing a smaller ring.

*Security and Trust:* In general, users trust devices and applications to the extent that they are under their direct control. A private cloud (either conventional or P2P) is built using resources owned and operated by the cloud customer organization itself. This results in a reasonably high level of confidence that data and computations will be handled according to the organization's security policies, provided that resources are protected from external and internal attacks. On the other hand, for public clouds, where customers outsource their data and computations, they have to trust the cloud service provider not to corrupt, leak or misuse data and carry out computations correctly. The problem is greatly aggravated for P2P clouds under scenario B; now the customer has to trust a large number of unknown third parties. Likewise, individuals providing resources have to trust a large number of unknown customers, hoping that they will use resources responsibly. The situation is slightly better for P2P clouds under scenario A but as long as devices are located physically in other people's homes, they could be subject to tampering and abuse. These are formidable challenges that so far do not have simple and general solutions. In case of specific applications, for example storage services, a clever combination of fragmentation, encryption

and replication has allowed the development of distributed, anonymous and censor-resistant solutions. Unfortunately, no similar solutions exist to render computations running on untrusted hardware tamper proof.

*Heterogeneity:* The API presented to a customer in an IaaS cloud is at the level of a host operating system over a virtualized machine. A P2P cloud under scenario B is likely to contain a diverse collection of devices ranging from broadband modems to game consoles to desktop PCs. It is highly unlikely that certain resource-constrained devices in such a collection be capable of running a virtualization layer suitable for supporting host operating systems. Yet, they may be capable of running a JVM (Java Virtual Machine) layer that would become the programming API.

*Incentives:* As with all other P2P systems, this is the "elephant in the room" also for P2P clouds, especially under scenario B. In P2P systems, incentives are necessary to achieve sufficient levels of cooperation and discourage free riding so that the system does not degenerate completely. In a P2P cloud, what would convince an individual to open up her personal resources to complete strangers even though she has nothing to gain other than goodwill? Clearly, some sort of incentive mechanism has to be put in place to encourage individuals to contribute their resources. The situation is a bit simpler for P2P clouds under scenario A where device owners may even have monetary incentives and can impose their will on individuals. The situation is rather different (and even simpler) for VC systems that typically have laudable objectives for scientific progress in fields such as radio astronomy, genomics or cancer research. Who would not want to have contributed to making history when the first extraterrestrial radio transmission is detected by SETI@home?

## Status

Preliminary research efforts and a few commercial applications that are appearing in the market suggest that P2P clouds can indeed be realized in practice, and can be a viable approach at least for some specific use cases.

At the University of Bologna, Italy, we have developed a proof-of-concept implementation of a P2P cloud architecture called P2PCS [P2PCS]. P2PCS demonstrated that it is indeed possible to use gossip-based protocols to assemble an unstructured and dynamic pool of resources, and to support the basic operations of a traditional cloud (elastic resource allocation and basic monitoring operations). Other projects at the University of Messina, Italy (Cloud@Home project [CLOUD@HOME]), at INRIA, France (Clouds@Home [CLOUDS@HOME], note the additional "s"), and in the context of the European Union Nanodatacenters project [NaDa] explore similar concepts. The Nanodatacenters project studies a decentralized cloud architecture where home gateways, controlled by Internet Service Providers (ISPs) are used to provide computing and storage services. Using a managed peer-to-peer model, Nanodatacenters form a distributed data delivery infrastructure suitable for delivering interactive applications to a large number of clients. These applications benefit greatly from the presence of a large number of geographically dispersed "nano" data-centers instead of a few large data-centers, since they are likely to be located closer to end users, and therefore provide lower latency.

A number of commercial distributed storage solutions are based on some of the principles outlined in this article. Wuala (http://www.wuala.com/) allows users to trade space on their hard disks; in order to address the security and trust issues, files are encrypted and split into chunks before they are transferred to other users. The incentive for users sharing disk space is the possibility for them to upload their own data to the Wuala cloud, and access that data from any place and from any device (the client application is available

for multiple platforms). Sher.ly (http://sher.ly/) implements similar functionalities but is oriented at the business sector. Sher.ly allows small and medium-sized businesses to use their own machines and infrastructure to create a secure, always-on private cloud to share files among specific groups of users.

## Conclusions

Peer-to-Peer algorithms are traditionally associated with file-sharing applications; however, as we have shown in this article, P2P techniques are powerful "building blocks" that can be used to create large scale, robust infrastructures out of unreliable components. We have seen how a P2P Cloud could be assembled at virtually zero cost, and would provide computation and storage resources like a conventional infrastructure hosted in a large datacenter. A fully distributed system has the advantage of lacking centralized control, and therefore does not have any single point-of-failure nor any central controlling authority. Of course, there are also disadvantages: security and trust are extremely difficult, if not impossible, to achieve in a fully distributed scenario. Moreover, one should expect a much lower Quality of Service from a P2P cloud compared to that provided by a centralized or federated Cloud. These considerations suggest that the P2P paradigm is not yet appropriate for public clouds serving the general user community. However, a P2P cloud would be good enough for realizing a private infrastructure out of already existing resources already owned by the same organization (think of a private cloud owned and operated by a single company, or by a group of friends who trust each other). From a strictly technical point of view, we already have all the ingredients to build a P2P cloud, but they need to be packaged together appropriately in order to get the overall design right.

## References


[P2PCS] O. Babaoglu, M. Marzolla, M. Tamburini, Design and Implementation of a P2P Cloud System, proc. 27th Annual ACM Symposium on Applied Computing (SAC 2012), march 26—30, Riva del Garda, Italy, pp. 412—417, ACM, ISBN 978-1-4503-0857-1, DOI 10.1145/2245276.2245357

[CLOUD@HOME] V. D. Cunsolo, S. Distefano, A. Puliafito, M. Scarpa, Volunteer Computing and Desktop Cloud: The Cloud@Home Paradigm, Network Computing and Applications, 2009. NCA 2009. Eighth IEEE International Symposium on, vol., no., pp.134,139, 9-11 July 2009
DOI 10.1109/NCA.2009.41

[CLOUDS@HOME] Artur Andrzejak, Derrick Kondo and David P. Anderson. Exploiting Non-Dedicated Resources for Cloud Computing. In 12th IEEE/IFIP Network Operations and Management Symposium (NOMS~2010), Osaka, Japan, April 2010, DOI 10.1109/NOMS.2010.5488488

[NaDa] Vytautas Valancius, Nikolaos Laoutaris, Laurent Massouli, Christophe Diot, and Pablo Rodriguez. 2009. Greening the internet with nano data centers. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies* (CoNEXT '09). ACM, New York, NY, USA, 37-48. DOI 10.1145/1658939.1658944

[BOINC] Anderson, D.P., BOINC: a system for public-resource computing and storage, *Grid Computing, 2004. Proceedings. Fifth IEEE/ACM International Workshop on* , vol., no., pp.4,10, 8 Nov. 2004, DOI 10.1109/GRID.2004.14

[EDGES] Etienne Urbah, Péter Kacsuk, Zoltan Farkas, Gilles Fedak, Gabor Kecskemeti, Oleg Lodygensky, Csaba Attila Marosi, Zoltán Balaton, Gabriel Caillat, Gabor Gombás, Adam Kornafeld, József Kovács, Haiwu He, Róbert Lovas: EDGeS: Bridging EGEE to BOINC and XtremWeb. J. Grid Comput. 7(3): 335-354 (2009), DOI 10.1007/s10723-009-9137-0