# A Hybrid P2P Network-Based Remote Treatment SaaS Cloud Computing

**AbirAchache, AbdelhalimBaaziz, Toufik Sari**

***Abstract: Cloud Computing has emerged as a new model for managing and using IT systems that has become a key topic in the field. The technology consists of deporting to remote servers the treatments, storage and management of data usually performed locally in order to access it as a service. Having so many opportunities also costs a lot in terms of infrastructure. The idea is to create a Cloud without infrastructure for providing to clients services in software as a service "SaaS" form. The solution consists of using a hybrid P2P network as an infrastructure for our Cloud. Cloud services are initially offered by the Cloud but hosted and executed through P2P network peers. The realized prototype and simulation results (with PeerSim) are encouraging and prove that a distributed and dynamic Cloud architecture based on a peer-to-peer network can be a reliable solution to infrastructure issues.***

***Key Words: Cloud Computing, Peer to Peer, PeerSim, SaaS, Services.***

## I. INTRODUCTION

Over the last decade, we have assisted an explosion in the use of the Internet for distributed applications, associated with the development of information technology; a kind of revolution that is becoming more and more important and especially faster. This rapid development is reflected by the emergence of several technologies among Cloud Computing. Cloud Computing is the most important technology of the 21st century that has made the promise of utility computing a reality. This technology is not new, but it is a new model of exploitation resulting from the combination of several technologies that have evolved and matured over the years [1, 2].

Cloud Computing is considered as a computing model in which computing resources include servers, storage, applications and services, are shared among several users. IT resources can be provisioned, released and consumed dynamically by users [1, 3, 4, 5]. Cloud Computing uses a service-oriented business model. These services are offered in three different forms, 1) IaaS: infrastructure as a service, 2) PaaS: platform as a service, 3) SaaS: software as a service, it is the most well-known and the widely used form of Cloud Computing that provides applications to users on demand over the Internet [1]. SaaS providers concede their applications to their clients as a service on demand via a pay-per-use or free payment, which allows users to get rid of the burdens of local installation and executions of applications [2, 3, 4]. Nowadays, SaaS architecture relies on many concepts and technologies such as SOAP (Simple Object Access Protocol) to enable users to use them [6]. In this communication protocol, the implementation of services is built on XML allowing users to use the services according to the syntax declared in the XML file, Hence the need for a careful preparation by expert [7].

Cloud Computing offers many benefits such as computing power, scalability, high availability, resilience and flexibility, which has encouraged most companies to migrate their IT solutions to it by adopting one of the offered models according to their needs [2, 3, 4]. The majority of systems based on this technology rely on a centralized structured infrastructure where they are hosted in large data centers and managed centrally, however, when service requests increase this can lead to system bottlenecks [1] and network congestion causing a huge load on Cloud servers. In addition, the data centers based on Cloud are expensive to build and generally belong to important organizations such as Google and Microsoft. Another disadvantage of centralized Clouds is that their geographic location, which may be the best only for the owners and not for clients. Consequently, recent research has investigated a different Cloud Computing conception strategy without any centralized installation using P2P technologies [8].

Peer-to-peer systems have emerged as an attractive and promising alternative model offering a significant improvement in the conception of large-scale distributed systems and the evolution of Internet architectures. Peer-to-peer systems are distributed systems composed of a set of heterogeneous, autonomous, dynamic, interconnected nodes able to self-organize into network topologies in order to share a part of their own physical resources such as disk space, network processor or bandwidth, as well as logical resources such as services or different forms of knowledge. P2P network participants play a similar role, each node can act as both a server and client and they are directly accessible by other nodes without passing through intermediate entities. Depending on the degree of centralization, P2P networks can be divided into: 1) hybrid P2P network, 2) decentralized (pure) P2P networks, 3) structured P2P network with a distributed hash table (DHT) [9, 10, 11, 12, 13].

In this article, we aim to propose a Cloud solution using a hybrid P2P network as an infrastructure that allows peers to use and execute the offered SaaS services hosted on other peers. The proposed solution has a central entity that is responsible for feeding the system with services. Unlike the

existing service definitions, our services are defined as java

**Revised Manuscript Received on December 30, 2019.**

**AbirAchache**, Laboratoire de GestionElectronique de Documents (LabGED), Computer Science Department, BadjiMokhtar Annaba University, Algeria.E-Mail: abirachache@gmail.com

**AbdelhalimBaaziz**, Laboratoire de GestionElectronique de Documents (LabGED), Computer Science Department, BadjiMokhtar Annaba University, Algeria.E-Mail: baaziz@labged.net

**Toufik Sari**, Laboratoire de GestionElectronique de Documents (LabGED), Computer Science Department, BadjiMokhtar Annaba University, Algeria.E-Mail: sari@labged.net

classes (or functions) using introspection techniques [14] that allow us to develop the tasks of creating, hosting and using services. The hosting of a service by a peer is done directly by downloading its java class without any need for a treatment or pre-condition which allows the system to be used by a larger public.

The rest of this article is organized as follows: Section 2 presents the context of the work. In Section 3, we present a comprehensive and detailed description of our system. We describe the experimental framework and we present the obtained results in Section 4. Section 5 concludes the paper with future directions for the research.

## II. RELATED WORKS

SaaS is a current Cloud Computing trend in the information technology sector. In recent years, it has successfully expanded as a new software application delivery model [15, 16] that will change the way people build, sell, buy and use software by overcoming a range of traditional problems [17, 18]. SaaS has also promised higher values such as performance, quality, facility and low cost, that's why many companies have migrated from the traditional business model to the SaaS model [16] [19]. One of the emerging trends is the use of vertical SaaS applications that focus on all business processes from customer relationship management (CRM) to human resources management (HRM) and supply chain management (SCM) in a specific type of customer [17]. In addition, SaaS is used in Office 365, Google Apps, salesforce.com, Citrix GoToMeeting, Cisco WebEx and Netflix, Vonage VoIP and Skype, etc.

SaaS sometimes referred to as "on-demand software/hosted software", is a software distribution model in which applications are centrally hosted by the supplier and provided as a service in a multi-tenant environment based on virtualization technology, where a large number of users can exploit a server instance hosted on server-side [17, 19]. SaaS provides customers with a range of applications without any personalization, these applications are accessible by users via the Internet from anywhere in the world while eliminating the need for deployment, installation or maintenance [6, 18]. SaaS adopts a pay-per-use pricing model where users pay only for what they use [15, 17, 19]. SaaS is more flexible in terms of scalability where companies can add more users and more features that provide on-demand scalability [17]. The main feature of the SaaS service model is centralized management, which facilitate monitoring, control and updating that could be performed by suppliers in a single location offering a high degree of resilience [6, 18].

Currently, computer services are passing from traditional Internet services to Cloud peer-to-peer [20], which is reflected in a wide range of scientific research and studies that treats the coupling of Cloud Computing and peer-to-peer systems as a way to overcome some of the problems posed by both paradigms. Many decentralized mechanisms for data storage in the Cloud are based on P2P protocols. Shawn Wilkinson et al [21] have implemented a protocol called Stroj that creates a distributed network for the formation and execution of peer-to-peer data storage contracts in P2P Cloud. The network is composed of users wishing to sell their unused storage capacity and users purchasing the storage space that they need. Storj is built on Kademlia, a distributed hash table (DHT) used to store the locations of data fragments that are distributed over many nodes to ensure their availability. Gerardo et al in [22] proposed a P2P Cloud model that considers different approaches such as reputation and collaborative storage. The main feature of this project is its qualified storage method focused on QoS requirements based on reliability indices, which are totally transparent to the user. The reliability level for each peer is measured according to: its storage capacity (confidence), the number of disconnections during a day and its availability. Hanna Kavalionak and Alberto Montresor in [23] proposed a hybrid peer-to-peer network with Cloud to offer a highly available storage service, where a replicated service is provided on a mixed peer-to-peer and Cloud system. The authors designed a protocol that can self-regulate the amount of cloud storage resources (pay-per-use) when peer resources (free) are not sufficient and find a balance between the reliability (durability and availability) of the service and the economic costs. The attractive idea of this article is that the Cloud participates in the communication protocol as a normal peer and Cloud resources are exploited only when there are reliability risks. OzalpBabaoglu et al in [24] have proposed a prototype of P2P Cloud System, a fully distributed IaaS Cloud infrastructure that aims to provide both computing and storage services. The proposed system consists of several software modules organized in a layered structure: the Peer Sampling Service Module (PSS) aims to provide each node with information on network members, the Slicing Service (SS) used to provide a ranking service based on requests and returns slices of the entire Cloud based on certain criteria defined by the user, the aggregation service (AS) allows each counterpart to know the system-wide parameters without having to access to a global register, the Dispatcher is responsible for treating the requests submitted by the user. Sarada.B.S and Nagarajin.S in [25] have proposed a Cloud storage architecture based on P2P that provides a pure distributed data storage environment where all block servers are interconnected to form a ring structure. To minimize overhead costs, a decentralized weighted voting algorithm has been developed, which ensures a consistent sequential access. The proposed system performs the monitoring operation to find the best block servers in the P2P network in order to achieve efficient resource utilization and load balancing of the servers. The block servers are served as data resource nodes and P2P nodes. In [26] the authors proposed an efficient and scalable approach to cloud service provision based on a structured P2P infrastructure with a Distributed Hash Table (DHT) approach used to store, discover and share services.

The above mentioned works have proposed different solutions for storage in a Cloud based on a P2P network. Therefore, the Cloud and P2P overlay is not an innovation in itself, but the idea of using this coupling to create a software as a service delivery network for remote treatment is a new one to exploit it.

Although the P2P approach has been successfully used for about twenty years, the research that addresses the development of SaaS based on it is still rare. On the other hand, even nowadays, most P2P systems deal with simple content sharing, which is essentially different from the service sharing functionality. In this work we present a hybrid P2P network-based remote treatment SaaS Cloud.

## III. PROPOSED SOLUTION

The proposed architecture consists of a Cloud that uses a peer-to-peer network as an infrastructure to provide services in SaaS form to peers who don't have them. These services are executed by invocation on the peers hosting them. We remark that the hybrid peer-to-peer network is similar to the architecture of Cloud computing, which has favored our choice to use it as an infrastructure for our Cloud.

Our Cloud is a dynamic distributed one that has a minimal infrastructure consisting of a central server that will have the role of primary administrator and a set of nodes called super peer that act as partial Cloud servers (PCS) distributed geographically. PCSs perform the tasks of indexing, service request treatment, and are connected to each other by a network layer. The effective and functional infrastructure will be represented by a set of nodes forming the P2P network (including PCS) acting as clients. A client may have two different roles, a participant client when it hosts services and executes them on behalf of other clients who will then have the role of user. When a client joins the network, it connects to the nearest geographically PCS, which acts as a central server for all clients connected to it forming a cluster. The central administrator is responsible for the provisioning of services. These services are represented by methods contained in downloadable java classes that each peer can host or use remotely. PCSs maintain a trace of their connected clients and their hosted services. When a user requests a service, the PCS to which it is connected receives the request, performs a research and returns the information of the participants hosting the requested service. The user client chooses a participant client and invokes the service in this one. Once the request has been treated by the participant, this last one returns the result of the service to the requesting user client. The layered model of the system and its overall architecture are illustrated in Figure 1 and Figure 2 successively.

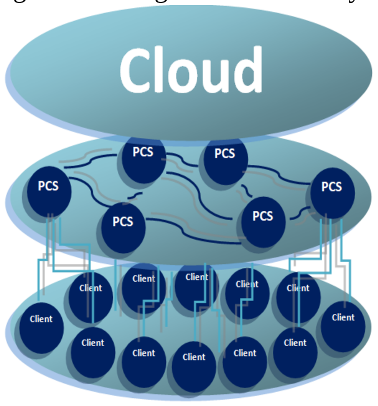


**Figure 1.Layered system model.**

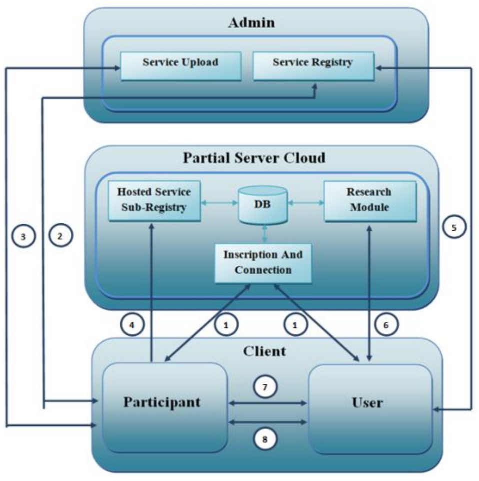


**Figure 2.Global system architecture.**

*3.1 Layer Model :*

The proposed system consists of three interdependent layers as shown in figure (1):

*3.1.1.The Cloud Layer:*

The Cloud layer mainly exposes the list of services offered by the system to clients in order to host or use them. The main actor of this layer is the administrator who is responsible for the feeding of the system by the services as well as their management (the provisioning). The administrator is the only one responsible for adding, deleting and updating the range of services. An interface is provided to display the range of the available services to clients for hosting or research needs. In other words, it allows participants to host a set of services offered that they will execute at the demand of users. These services are kept in a specified directory that determines the list of services offered by the system.

The services offered by our system are programs that provide special treatment. A service is a declaration of a java method where the service inputs are the arguments of the method and the service output (rendering) is the result returned by the method. The Cloud layer does not handle the treatment of requests or the execution of services; these tasks are successively performed by the partial Cloud servers layer and by the participant peers. The update of a service by the administrator requires the modification and recompilation of the class that contains it.

*3.1.2 The Partial Cloud Servers layer:*

It is composed of a set of interconnected PCSs responsible for service request treatment, management and coordination between clients. Each PCS continually keeps track of its connected clients and their hosted services. For this task, we use a SQL database that contains information about each client as well as information about their hosted services. The PCS acts as an index server for the users who are connected to it, it is the only responsible for the treatment of their service requests.

On receiving a service request search from a user, the PCS launches a local search on its own cluster, and returns the list of peer participants who detained the requested service as a result to the requestor user. If the search on its cluster is unsuccessful, it communicates with its PCS neighbors in order to locate the desired service in the other clusters.

*3.1.3 The Clients Layer :*

The client application is designed principally to allow clients to search, host or remotely execute services offered by the Cloud and must be installed on all network peers. After inscription to the system, the client can choose to be a participant or a user. The participant is an effective client in the system acting as a service host and a service provider at the same time. He chooses a set of services he wants to host by selecting them from the central node. However, the user is a consumer client seeking to exploit these services for its own interest. This layer exploits the principle of P2P networks to allow a direct connection between clients during the invoking of services without any intervention from a third party.

*3.2 Detailed description:*

The various tasks performed between the three layers are described below:

*Inscription and Connection*

This step (operation 1, figure 2) allows clients (participants / users) to connect to the system with a username and password to be identified knowing that they must register in advance by creating an account. Client authentication is ensured by the last partial Cloud server to which it identifies itself. If the authentication was performed by a PCS other than the one to which it was connected, the last PCS repatriates the client's information. The main objective is to allow PCSs to know the information of their clients in order to be able to locate its hosted services as well to keep track of the clients connected to the system.

*Service Hosting:*

This task (operation 2, 3 and 4, Figure 2) allows connected participants to select and download a set of services from the services offered by the Cloud. The selection of services to be hosted is made via an interface. This interface allows participants to dynamically download the compiled Java classes that contain services represented by Java method. After hosting, the participant performs an analysis of the hosted services to detect the parameters (input/output) of each service; he creates a manifest file in XML format that will be used later for an automatic generation of the interface associated to the service. The participant informs its PCS of the services that have been recently hosted.

To perform this task, the administrator has a ServiceRegitry module that allows the participant to get a list of the services offered by the Cloud and a ServiceUpload module that permits to upload services to the participant. For the participant, he has a RequestServiceList module that allows it to obtain the list of services offered by the Cloud, a ServiceSelection module that enables it to choose the services to host and to download their corresponding classes, a ServiceAnalysis module to create the service's manifest files and a HostedServiceInformation module used to inform its PCS of its hosted services. The PCS has a HostedServiceSubRegistry module that allows it to keep track of the services hosted by the participants (Figure 3). The scheme presented in Figure 4 explains the hosting procedure.

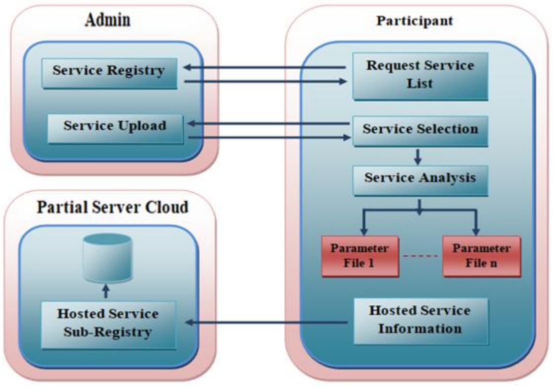


**Figure 3.Hosting Step.**

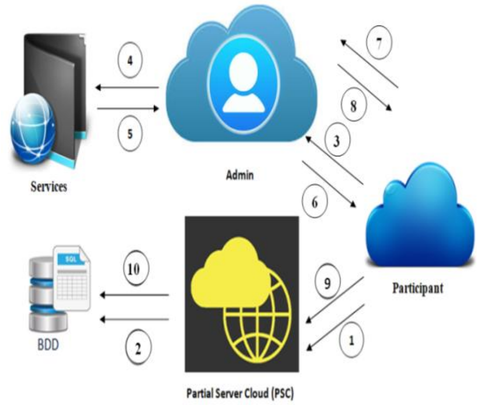


**Figure 4.Hosting Procedure.**

1. The participant identifies himself by connecting to the system.
2. The PCS updates its database of connected clients.
3. The participant requests the administrator for a list of the services offered by the system according to criteria.
4. The administrator consults the service directory.
5. The administrator generates a list of available services according to the criteria.
6. The administrator returns the list of services to the participant.
7. The participant chooses the services that he wants to host and makes a request to the administrator.
8. The participant downloads the classes containing the selected services.
9. The participant informs his PCS of his new services.
10. The PCS updated his database.

*Service Search:*

This step (operation 5 and 6, Figure 2) allows connected users to benefit from the services offered by the Cloud. This step starts with a search in the Cloud layer in order to choose a service from the services offered by the system, and ends with a search in the Partial Servers Cloud layer to locate the service.

In the first phase, the user has a RequestServiceList module that allows him to get a list of the services offered by the system and to choose one; a ServiceResearch module that enables him to send a search request for the selected service to his PCS and obtain a list of participants hosting the service. On its side, the PCS has a ReceptionRequestService module that allows it to receive the search request and a Research module that enables it to launch a search in its local database for the requested service and to establish a list of participants hosting the service (Figure 5). The ReceptionRequestService module returns the result to the user. If the local search is unsuccessful, the Research module allows the PCS to propagate the search request to the neighboring PCSs through the flood mechanism [11]. The search is stopped if the list of participants returned by one of the neighbors is not empty or the TTL (Time de Live) associated with the search request reaches its maximum value. The scheme presented in Figure 6 explains the search procedure.

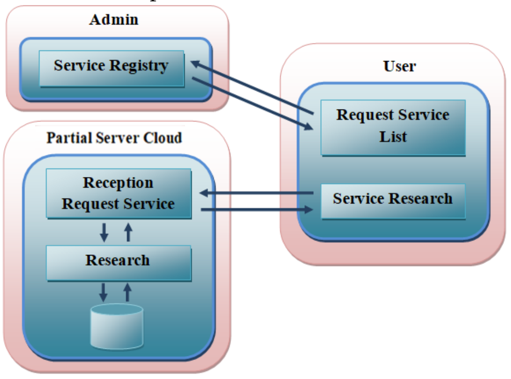


**Figure 5.Research Step.**

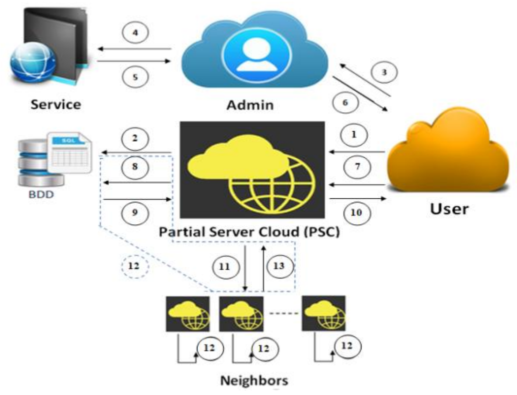


**Figure 6. Search Procedure.**

1. The user identifies him by connecting to the system.
2. The PCS updates its database of connected clients.
3. The client requests a list of available services according to criteria.
4. The administrator consults the service directory.
5. The administrator generates a list of available services according to the criteria.
6. The administrator returns the list of services to the user.
7. The user chooses a service and sends a service search request to his own PCS.
8, 9. The PCS consults the database to get a list of participants hosting this service.

If participant's list is not empty:

10. The PCS returns this list to the user.

Otherwise:

11. Since, no PCSs have no information about the hosted services of the neighboring clusters. Each PCS floods the service search request to its neighbors and increases the TTL value of the received service search request.
12. Each neighbor (PCS) performs steps 8 and 9, if the result set is empty and the maximum value of the TTL has not yet reached, executes step 11 otherwise step 13.
13. The research is stopped and each neighbor returns his list of participants. And then we proceed to step 10.

*Service Invocation:*

This step (operation 7 and 8, figure 2) allows the user and the participant to be directly connected. The user initiates a request to remotely execute a service of his choice by using a ServiceRequest module. The participant on his side has a ReceivingServiceRequest module that allows him to receive the request, to notify the ParameterInformation module that returns the manifest file (definition of the parameters of the requested service) to the user via the AutomaticCreationOfServiceInterface module that allows to automatically create an interface in the user to enable him to introduce the necessary arguments to the service, and to invoke the service remotely using java RMI (Remote Method Invocation) on the participant and receive the result of the treatment on the same interface. The participant has a ServiceInvocation module that permits him to receive the service parameters from the user, to execute the service and to return the result to the user (figure 7). The scheme presented in Figure 8 explains the invocation procedure:

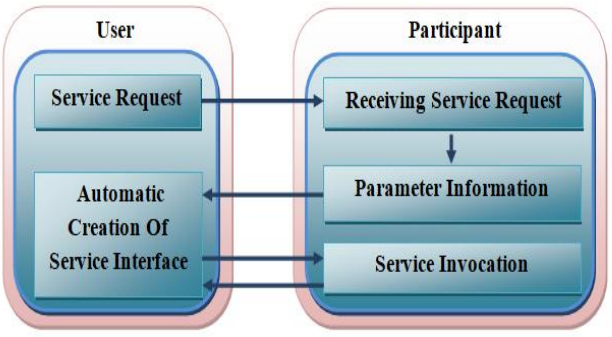


**Figure 7.Invocation Step.**

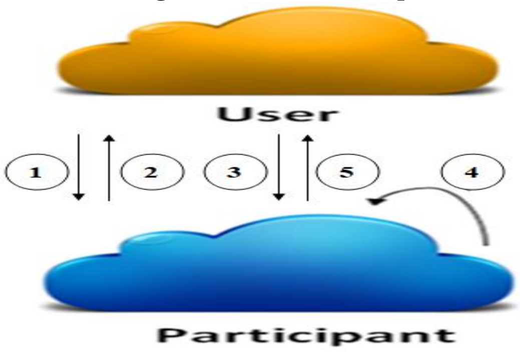


**Figure 8.Invocation Procedure.**

1. The user submits a request for a selected service to the participant.

2. The participant returns the list of requested parameters to the user (the manifest file).

3. The user invokes the service with the necessary parameters on the participant.

4. The participant executes the service with the arguments received from the user.

5. The participant returns the result of the execution to the user.

## IV. EVALUATION AND RESULTS ANALYSIS

In this section, we present the experimental environment and methodology used to evaluate the architecture presented in Section 3.

### *4.1 Experimental setting*

We implemented a Java prototype of the Cloud SaaS based on P2P system described in the previous section, using Java RMI technology for remote communication management and the PeerSim simulator as the experimental framework.

The PeerSim simulator is generally used to simulate a large-scale dynamic P2P network protocol. It can be used to simulate structured and unstructured overlays with a considerable size namely millions of nodes. It is written in Java, developed in part in the BISON project and published under the GPL open source license. PeerSim supports two types of simulations: a cycle-based simulation in which network nodes are randomly selected and each node protocol is called at each cycle and an event-based simulation in which a set of events is scheduled overtime and node protocols are called according to the order in which time messages are delivered. For further information about PeerSim [27, 28, 29, 30].

The proposed prototype was evaluated using a large number of simulations using the PeerSim cycle-based model in which a configuration file is used in the first step of simulation, this file consists of a the simple ASCII file, composed essentially of key-value pairs and containing all simulation parameters for all objects involved in the experiment [27]. For our experiments, we used test configurations by combining the parameters presented in Table 1:

**Table 1.Configuration settings.**

| | |
|---|---|
| Network size (Nodes) | 1000, 5000, 10000, 30000, 50000 |
| Number of Partial Cloud Server | 30 |
| Maximum Number of Neighbors for each Partial Cloud Server | 5 |
| TTL (time to live) | 7 |
| Communication technique between Partial Cloud Servers | Flood |

For adapting our solution to the PeerSim simulator, we have defined a set of classes that describe the behavior of the main actors in the experiment:

Clients: which are represented by a class that contains the different modules (methods) that allow them to perform different tasks (communicating with their PCS and other clients, service hosting, service search and service request). This class implements the CDProtocol interface which inherits the Protocol class and has the nextCycle() method where we have defined the different tasks that a peer must do during its progress. In our solution we have provided four possible scenarios with an equal probability per scenario: 1) hosting services, 2) searching for services, 3) deleting services and 4) the peer does nothing. Each scenario has a probability of 25%. This class uses the following data structure:

Local list of hosted services that contains all hosted services specific to each client.

Partial Cloud Servers: which are represented by a class that has the indexing, search modules and tools for communicating with neighbors. This class implements the Control interface which has "execute()" as a main method to inform each client of the network of the address of his PCS. This Control is called only once in the initialization step. This class uses the following data structure:

Partial list of hosted services that contains information about all services hosted locally by all clients of the PCS as well as their identifiers.

Observers, represented by classes that allow us to extract the measurements (statistics) needed to analyze and understand the behavior of our network. During each cycle, measurements are captured and analyzed at the end of the simulation. The experiments focus on five performance indicators: 1) Success_Search, the search of a requested service takes a positive value if the PCS finds at least one participant that hosts the requested service, 2) Success_Result, reflects the success of the result returned by participants after execution of the requested services, 3) Number_of_Resource_Nodes, it is the number of nodes hosting the requested service, 4) Failure_Result, which means the failure of participant nodes to correctly provide the requested service, 5) Failure_Search, the search for a requested service fails if the PCSs don't find any service provider.

The Central Node: for better concretizing of the Cloud environment and its central node, we have created a class of type Control accessible by all peers which is launched only once during the initialization phase of the simulation and which simulates the central node. This class uses the following data structure:

A global list of services that contains categories, sub-categories, names, inputs and outputs of services. It serves as a reference for clients to choose which services they will host or request.

### *4.2 Result and discussion*

We have realized five simulation scenarios according to the size of the P2P network (size: 1000, 5000, 10000, 30000, 30000, 50000), the results to be discussed are on the metrics previously introduced.

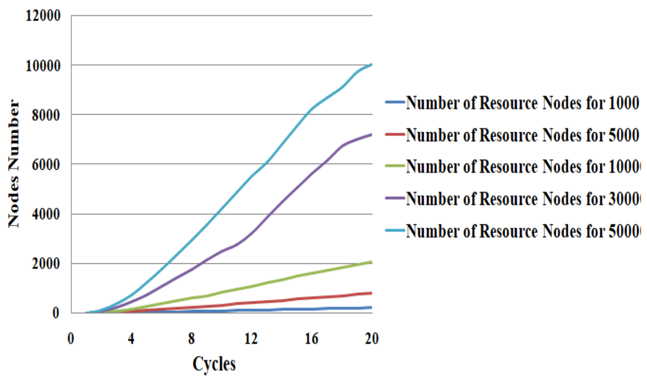
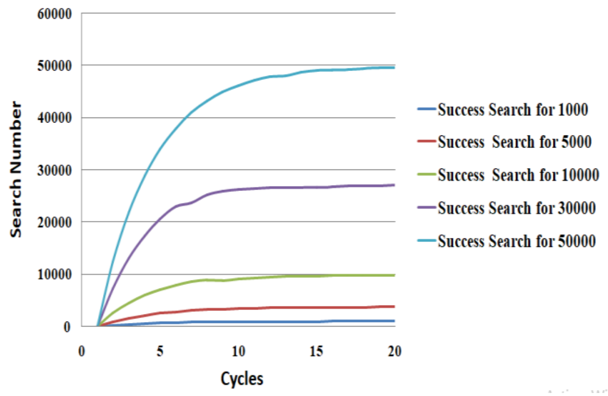


**Figure 9. Number of Resource Nodes.**

Figure 9 shows the variation in the number of nodes having the requested services by cycle for the 5 different simulation scenarios. By observing the graph in figure 9, we can see that the number of nodes that have required services has increased significantly over time. This is due to the fact that clients are actively hosting services and becoming participants in the system. As a consequence, this growth will have a positive impact on the search process.



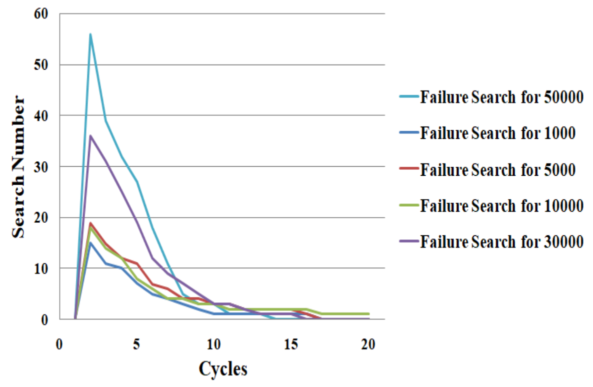
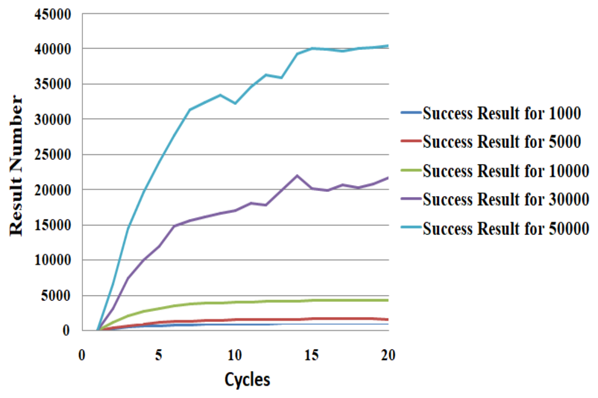


**Figure 10. Success Search Vs. Failure Search.**

Figure 10 shows the variation of the two metrics: Success_Search and Failure_Search in different simulation scenarios. According to the statistics, we first observe that the success search is higher than failure search in all scenarios and during all simulation time. Also, the results show that the success search factor increases constantly over time, which is an obvious consequence since the number of nodes that have the resource increases simultaneously (figure 9). This factor begins to stabilize from the cycle 15. While the Failure_Search factor increases slightly at the beginning of the simulation because at this stage there are fewer participants, then it decreases from the 3rd cycle and tends to zero over time.



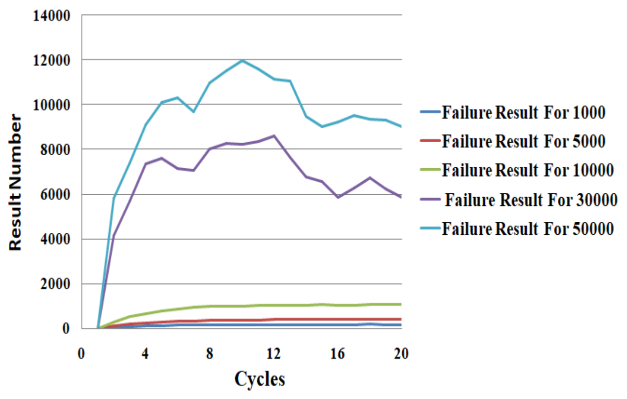
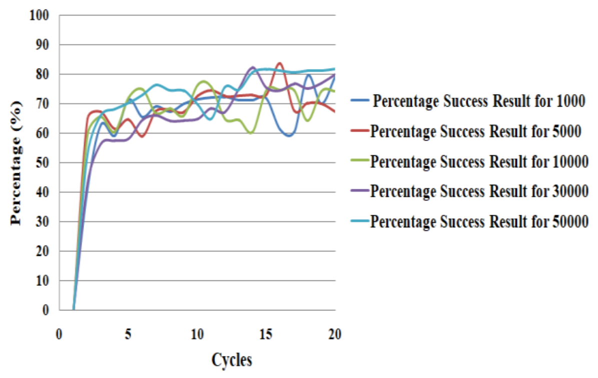


**Figure 11. Success Result Vs. Failure Result.**

Figure 11 shows the variation of the two metrics: Success_Result and Failure_Result in the different simulation scenarios. These are the most important factors that reflect the system's ability to assured the on-demand resource provisioning feature of Cloud Computing. According to these statistics, we can observe that the success result has continuously increased reaching the maximum in the 15th cycle then it stagnates until the end of simulation. This change is associated with a variation of the failure result, this last one increases over time until the 9th cycle then it decreases gradually then it stabilizes until the end of simulation. Moreover, we can clearly see that the results of success result remain superior to those of failure result.



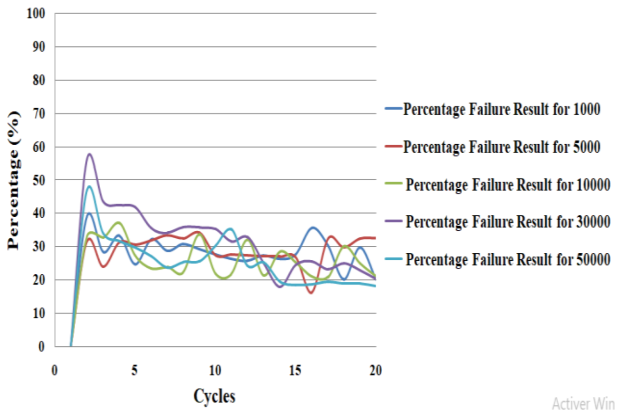


**Figure 12.Percentage Success Result Vs Percentage Failure Result.**

Figure 12 represents the evolution of the percentage of success results and failure results compared to the requests made in the various simulation scenarios. During the first three cycles, we observe a significant increase in the percentage of success results reaching a rate of 60%. This variation is associated with a rapid progression of the percentage of failure results that reaches 40%. From the third cycle the percentage of success results continues to grow slowly reaching a maximum value of 80% then it stabilizes until the end of the simulation. While the percentage of failure results declines over time associated with a stagnation that tends towards a value of 20%.

## V. CONCLUSION AND PERSPECTIVES:

In this work we have proposed a Cloud to provide remotely execution software as a service (SaaS) that uses a hybrid peer-to-peer network as infrastructure. Partial Cloud Servers are available and are responsible for indexing and coordination, allowing user clients to make service requests and connect them with participant clients. Each client in the network can be a participant or a user of the system. Among the key points of our solution is the distribution of the Cloud over a distributed infrastructure which is formed by the clients. Instead that the treatment is done by the Cloud, it will be delegated to these clients. The experimental results obtained have shown that the proposed system can achieve a high success rate up to 80%. These results are encouraging and prove that a distributed and dynamic Cloud architecture based on peer-to-peer network for service delivery could be a reliable solution to infrastructure issues.

For future work, we aim to improve our system to overcome the issue of service availability and some of the challenges of peer-to-peer systems such as peer dynamics, free-riding and load balancing to ensure that the key features of on-demand resource provisioning in Cloud Computing are maintained. In other words, the guarantee of on-demand service availability must be maintained in order to build a more complete and reliable Cloud environment.

## REFERENCES

1. Q.Zhang, L.Cheng and R.Boutaba, Cloud computing: state-of-the-art and research challenges, Journal of Internet Services and Applications, Volume 1, Issue 1, pp 7–18, April 2010.
   H.I.Syed and N.A.Baig, Survey On Cloud Computing, International Journal of Emerging Technology and Advanced Engineering, Volume 3, Issue 4, April 2013.
2. A.Prasanth, Cloud Computing Services: A Survey, International Journal of Computer Applications, Volume 46– No.3, May 2012.
3. S.Bandela, R.Gadde and S.Pabboju, SURVEY ON CLOUD COMPUTING TECHNOLOGIES AND SECURITY THREATS, International journal of engineering research and technology (IJERT), Volume 2, Issue 6, May 2015.
4. https://www.nist.gov/
   G.Kulkarni, J.Gambhir and R.Palwe, Cloud Computing-Software as Service, International Journal of Computer Science & Information Technology Research Excellence, Vol. 2, Issue 1, January-February 2012.
5. D. Box, D. Ehnebuske, G. Kakivaya, A. Layman, N. Mendelsohn, H. F. Nielsen,S. Thatte and D. Winer. Simple Object Access Protocol (SOAP) 1.1, W3C Note 8,W3C, May 2000. http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
6. P.Kisembe and W.Jeberson, FUTURE OF PEER-TO-PEER TECHNOLOGY WITH THE RISE OF CLOUD COMPUTING, International Journal of Peer to Peer Networks (IJP2P), Vol.8, No.2/3, August 2017.
7. R.K.Maurya, S.Pandey and V.Kumar, A Survey of Peer- to-Peer Networks, International Journal of Advanced Research in Computer and Communication Engineering, Vol. 5, Issue 4, April 2016.
8. N.Chaudhary and J.Surolia, A survey on peer to peer system applications, International Journal of Innovative Computer Science & Engineering, Volume: 2, Issue: 1, 2015.
9. B. Pourebrahimi, K. Bertels and S. Vassiliadis, A Survey of Peer-to-Peer Networks, In: 16th Annual workshop on circuits, systems and signal processing, ProRisc 2005, Veldhoven, The Netherlands, 2005.
10. R.Steinmetz and K.Wehrle, Peer-to-Peer Systems and Applications, Springer: Berlin, Germany, 2005; Volume 3485.
11. I.Filali, F.Bongiovanni, F.Huet, and F.Baude , A Survey of Structured P2P Systems for RDF Data Storage and Retrieval, Transactions on large-scale data-and knowledge-centered systems III, pp. 20–55. Springer-Verlag, Berlin, Heidelberg, 2011.
12. J.M.Doudoux. Développons en Java. 2019.
13. P.Guo, A Survey of Software as a Service Delivery Paradigm, In: Seminar on Internetworking, 2009.
14. K.Sharma, Software as a Service and Cloud Security, Journal of Computer, Vol.3 Issue. 2, February- 2018.
15. T.C. Sandanayake and P.G.C.Jayangani, Current Trends in Software as a Service (SaaS), International Journal for Innovation Education and Research, Volume: 6, No-02, pp.221-234, 2018.
16. K.V. K Mahesh Kumar, SOFTWARE AS A SERVICE FOR EFFICIENT CLOUD COMPUTING, International Journal of Research in Engineering and Technology, Volume: 03, Issue: 01, Jan-2014.
17. B.Kaur, Software As A Service: A Brief Study, International Research Journal of Engineering and Technology (IRJET), Volume: 02, Issue: 03, June-2015.
18. S. Jo and J.Han, Convergence P2P cloud computing, International Journal of Peer-to-Peer Networking and Applications, 6, pp: 1153–1155, May 2018.
19. S.Wilkinson, T.Boshevski, J.Brandoff, J. Prestwich , G.Hall, P.Gerbes, P.Hutchins and C. Pollard, Storj A Peer-to-Peer Cloud Storage Network, December 15, 2016.
20. G .García-Rodríguez and F.López-Fuentes, A Storage Service based on P2P Cloud System, Research in Computing Science , pp. 89–96, 2014,
21. H.Kavalionak and A.Montresor, P2P and Cloud:A Marriage of Convenience for Replica Management, the 6th International Federation for Information Processing (IFIP), Workshop on Self Organizing Systems. Lecture Notes in Computer Science. Springer, pp. 60–71, 2012.
22. O.Babaoglu, M.Marzolla and M.Tamburini, Design and Implementation of a P2P Cloud System, in Proceedings of the 27th Annual ACM Symposium on Applied Computing. ACM, pp. 412–417, 2012.
23. B.S.Sarada and S.Nagarajan, Improving P2P in Cloud Computing based on Performance, International Journal of Computer Applications (IJCA), Special Issue on “Computational Science - New Dimensions & Perspectives”, NCCSE, 2011

24. A.Zhygmanovskyi and N.Yoshida, Cloud Service Provisioning Based on Peer-to-Peer Network for Flexible Service Sharing and Discovery, Journal of Computer and Communications, 2, pp.17-31, 2014.
25. G.P.Jesi, PeerSimHOWTO:Build a new protocol for the PeerSim 1.0 simulator, December 24, 2005.
26. S.Srati, D.C. Jinwala and S.Garg, A survey of simulators for P2P overlay networks with a case studyof the P2P tree overlay using an event-driven simulator, Engineering Science and Technology, an International Journal,Volume 20, Issue 2, April 2017, Pages 705-720.
S.Naicken, A.Basu, B.Livingston and S.Rodhetbhai, A Survey of Peer-to-Peer Network Simulators, In The Seventh Annual Postgraduate Symposium, vol. 2, 2006.
27. M.Ebrahim, S.Khan and S.SherazandH.Mohani, Peer-to-Peer Network Simulators: an Analytical Review. arXiv preprint arXiv:1405.0400, 2014.