IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 38, No. 2, April, 2024, pp. 631-641
Journal homepage: http://iieta.org/journals/ria

Hybrid P2P-Based Architecture for Remote Software Utilization

Abdelhalim Baaziz*, Abir Achache

Laboratoire de Gestion Electronique de Documents (LabGED), Computer Science Department, University Badji Mokhtar-Annaba, Annaba PO-Box 12, 23000, Algeria

Corresponding Author Email: baazizhalim@gmail.com

https://doi.org/10.18280/ria.380227

Received: 10 October 2023
Revised: 1 February 2024
Accepted: 3 March 2024
Available online: 24 April 2024

Keywords:
SaaS, SOA, HTML, P2P, service invocation, remote-processing, PeerSim

ABSTRACT

The world presents us with two contradictory situations. Firstly, most people encounter the problem of lacking necessary hardware and/or software resources. Secondly, some individuals or organizations possess these resources but fail to utilize them to their full potential. The proposed solution is to provide underutilized resources to those who require them. However, it is important to ensure that there is no illegal copying or pirating of software. Instead, the owners of these resources can offer their services by processing data remotely. The Peer-to-peer (P2P) paradigm is used primarily for collaborative systems over a network. This article proposes a Hybrid P2P-Based collaborative architecture that employs HTTP transport to provide data remote-treatment services. In our peer-to-peer (P2P) solution, one peer provides a service, such as a software application capable of processing specific data, while another peer actively searches for this software and sends its own data to be processed by it. In this scenario, the second peer can utilize the software without encountering piracy issues, given that the first peer executes the software locally and adheres to the appropriate license. By employing HTTP for communication, it enables collaborative interaction among heterogeneous peer platforms. This system enables individuals without essential hardware and/or software resources to leverage the resources provided by others. The simulation results conducted using PeerSim simulator, are encouraging, indicating that the proposed architecture can serve as a reliable solution for collaboration between peers. The outcomes of the simulation demonstrate a significant level of satisfaction across the essential metrics we defined to assess the effectiveness of our solution. particularly in terms of responsiveness to requests, resulting in approximately 80% satisfaction and 20% dissatisfaction rates for requests.

1. INTRODUCTION

Software as a Service (SaaS) stands out as a modern trend within the cloud computing domain of the information technology sector. This innovative model aims to dissociate software ownership from its usage. By providing software functionality through distributed services that can be configured and bound upon delivery, numerous existing constraints related to software evolution, deployment, and utilization can be effectively addressed. This approach has gained significant momentum in recent years, indicating its success as a new software distribution model [1-5]. Two conflicting scenarios exist in our world. While many individuals experience a shortage of hardware and software resources, others or organizations possess these resources yet they remain underutilized. The SaaS model can be used to address this issue. The aim is to provide these underutilized resources to those who need them the most, without resorting to illegal copying or pirating software. Instead, the owner offers their service by remotely processing data. This approach is feasible due to the SaaS model, where applications are hosted remotely by the provider and delivered as a service upon end-users' requests via the Internet, utilizing a utility pricing model. This results in reduced possession costs, allowing customers to eliminate concerns related to software package licenses, installation, and updates [6, 7].

Both the academic and industrial sectors have devoted considerable attention to Peer-to-Peer (P2P) systems as a promising alternative model. It holds potential to substantially enhance the design of large-scale distributed systems and facilitate the evolution of Internet architectures. These systems collectively accumulate extensive resources, which expand as the demand for them increases. This expansion happens as new nodes, which create additional demand, also contribute fresh resources to the distributed system. This process fosters a mutually beneficial cycle of growth and expansion [8-13]. Currently, the majority of Peer-to-Peer (P2P) systems are primarily used for simple content sharing, such as sharing files. However, this functionality is fundamentally different from service sharing. While numerous projects have aimed to harness the resources within these systems to offer services, the most prevalent P2P solutions are frequently utilized for sharing files that have been restricted due to piracy, including audio and video files, as well as software.

The shift from a Cloud SaaS system to a P2P-based system, which provides resources as SaaS, serves as a strategy to

mitigate the absence of Cloud infrastructure. This transition also allows peers to actively contribute to enhancing the system's functionality. In computer network environments, the majority of programmers are usually highly skilled and involved in intricate software development endeavors. Frequently, numerous programmers collaborate actively to create a single system product in a coordinated manner. This collaborative model involves experienced programmers sharing insights and closely collaborating to develop sophisticated software, which stands in stark contrast to the personal computer model, where a lone beginner typically works independently [14]. However, in a Peer-to-Peer (P2P) network, most users are typically beginners with little to no experience or training in software development.

Service-Oriented Architecture (SOA) is a conceptual framework centered on the definition of services and their interactions [15]. The main idea of SOA is to describe all functions as separate and self-contained services, each having a well-defined and callable interface that can be combined in specific sequences to create business processes [16]. Simply, SOA is an architectural style that facilitates the integration of various applications and resources as services through standardized interfaces, enabling the exchange of structured data and coordination among services to respond to changing business requirements [17, 18]. The collaborative process within SOAs can be depicted through Figure 1, which adheres to the find, bind, and invoke paradigm. Within this process, a service customer initiates a search for a fitting service by querying the service registry with specific criteria. Should a service matching the criteria be available, the registry furnishes the customer with the interface contract and endpoint address for that service. Ultimately, the customer invokes the located service at the provider using the request/reply procedure.

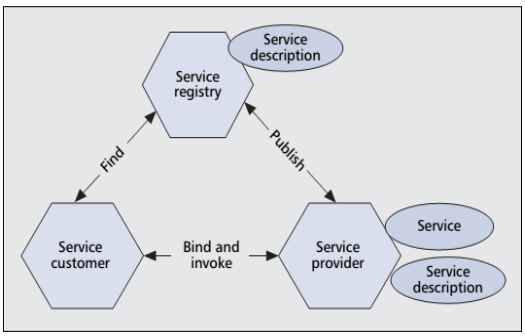


Figure 1. Collaborations SOA

Web services are services that meet certain characteristics, including being self-contained, modular, and loosely coupled, and are accessible over the internet. The standardization of these services is supervised by four organizations: the World Wide Web Consortium (W3C), the Organization for the Advancement of Structured Information Standards (OASIS), the Liberty Alliance, and the Web Service Interoperability Organization (WS-I). Although WS-I does not function as a standardization body, it offers installation-ready packages of Web services known as "profiles," along with tools and guidelines for their implementation.

The initial Web services profile, known as the basic profile, is centered around three primary standards: Simple Object Access Protocol (SOAP), Web Service Description Language (WSDL), and Universal Description Discovery and Integration (UDDI). These standards are respectively defined by W3C and OASIS. SOAP is a decentralized communication protocol that uses XML for exchanging structured information between the requester and the service provider in a distributed environment. WSDL is a description language based on XML grammar that defines network services as a collection of communication endpoints, allowing them to exchange messages. UDDI is a service directory that provides the fundamental infrastructure for discovering and publishing Web services. It adopts a standard approach for service location and invocation, as well as the management of metadata related to these services. By relying on these three standards, services can be defined, discovered, and invoked in terms of their interfaces instead of their implementations [7, 16, 19, 20].

These concepts can be difficult for users who are not experienced in programming. This is especially true in a social P2P network intended for the general public, where users often lack experience in programming. The complexity is further increased when it comes to services, their composition, and invocation.

Certainly, P2P users are typically individuals who may not be familiar with advanced methods for utilizing services. In such instances, a user-friendly system with simple user interfaces can assist these users in easily performing and utilizing the SaaS. Given the assumption that many users of social P2P networks lack programming experience, we propose an HTML hybrid P2P-based solution. A hybrid P2P network incorporates index server (monitor) centrally storing information about resource locations, and it relies on this index server for conducting searches. Based on the SOA paradigm, the service publishing, searching and binding of software offered by peers, are fully automated and managed by the hybrid P2P monitor manager. The solution allows users who are unable to process their data locally due to resources scarceness to search throughout the hybrid-P2P monitor the wanted software and then utilize it remotely by sending its data to be processed by the software's owner.

In this context, invoking the service entails executing the software with the given data, leading to the generation of a link pointing to an HTML page that displays the outcomes. Similar to WSDL, we employ XML-based automated procedures with HTTP support to facilitate various stages of our solution. Due to the users' limited programming experience, we have designed a comprehensive automated solution that assists them in navigating through distinct phases, including publishing, searching, binding, and invoking. While our proposed solution remains specific, we conform to the three mentioned standards, implementing them with our unique approach.

The following paper is structured as follows: Section 2 provides a review of the related literature. In Section 3, we introduce the architecture and components of our proposed system, along with the automated mechanisms for service publishing, finding, invoking, and request/reply. Section 4 describes the experimental framework and presents the results obtained. Security, profile establishment, and incentive mechanisms are discussed in Section 5. Lastly, Section 6 concludes the paper and outlines future research directions.

2. RELATED WORK

In the following section, we will examine a representative sample of pertinent solutions that are related to our proposed

approach. These systems can be categorized into:

2.1 Cycle sharing systems on grid infra-structures

These systems facilitate the execution of parallel applications on remote computers [21]. For instance, the Institutional Grid Globus [22] serves as a technology for grid deployment. It offers mechanisms for communication, authentication, network information, and data access. However, its authentication and authorization models target institutions, posing challenges for regular users in deploying applications atop the Grid. In contrast, Condor [23] enables the integration and utilization of remote workstations. It optimizes workstation utilization, enhances available resources for users, and operates effectively in a distributed ownership setting. In Condor, jobs necessitate executable binary code and compatible machines for execution.

2.2 P2P access to computing cycles available remotely

One example of a solution that combines Grid and Peer-to-Peer models is GridP2P [24], a platform designed for distributed cycle sharing. Its objective is to leverage parallel execution of common applications by allowing regular users to access remote idle cycles, which can significantly speed up the performance of everyday applications. Additionally, users can also contribute their own spare cycles when not in use. Another decentralized P2P network for sharing computing cycles is presented by Mason and Kelly [25], which can be used to develop applications using the Microsoft .NET Remoting infrastructure. Developers can benefit from a familiar programming model by using the Microsoft .NET Remoting infrastructure, as it allows for the potential of porting existing .NET Remoting and Java RMI applications with relative ease. Furthermore, Galatopoullos et al. [26] have devised a middleware architecture enabling the execution of composite services by amalgamating private and public services across P2P overlay networks. This middleware harnesses off-the-shelf P2P technologies to tackle challenges like pervasive service connectivity and distributed group management and trust. It facilitates genuine peer-to-peer execution of composite services by routing SOAP messages end-to-end, obviating the need for service-level intermediaries or centralized service registries. Moreover, this architecture segregates the runtime and connectivity layers, enabling the incorporation of different runtimes and P2P overlays.

The solutions we have discussed indicate that a certain degree of expertise in programming is necessary in order to fully utilize them. However, our target audience consists of individuals with limited control and technical proficiency. Our aim is to provide these individuals with the opportunity to utilize shared resources in the simplest manner possible.

3. THE PROPOSAL ARCHITECTURE

Developing a system that enables users to access the hardware and software tools of other users is a viable solution that addresses various issues such as illegality, heterogeneity, limited computing/memory capacity, and software unavailability. Our goal is to create a fully automated system that simplifies interaction and utilization of services provided by the system, requiring minimal effort from users.

To grasp the system, consider the following scenario: a user wishes to process data but lacks the required resources (hardware and/or software). Another user possesses these resources, and they are currently underutilized. In this context, the idea is to bring these users together to collaborate, allowing the first user to leverage the potential resources of the second user. The initial user sends their data to be processed by the second user, who subsequently forwards the resulting output back to the first user. These users form a set of peers that can collaborate between them.

These peers are overseen by a central monitor, which serves as the central node in a centralized peer-to-peer network.

The proposed solution involves building a centralized peer-to-peer (P2P) network and integrating it with a web application that incorporates key concepts such as service supply and publication (software), service research, discovery and request, and client-to-software supplier data exchange. By deploying this solution on the Internet, the system is designed to function through web interfaces that utilize the HTTP protocol, which is universally supported by various operating systems, thus effectively addressing the problem of heterogeneity.

The system's architecture comprises three fundamental entities, depicted in Figure 2, which operate within a minimal infrastructure. These entities are: (1) Software Suppliers, who provide their software as a service for others to use, (2) Software Requesters, who utilize the software to process their data remotely, and (3) Monitors, who serve as system managers and oversee its operations. The system's overall architecture is depicted in Figure 3.

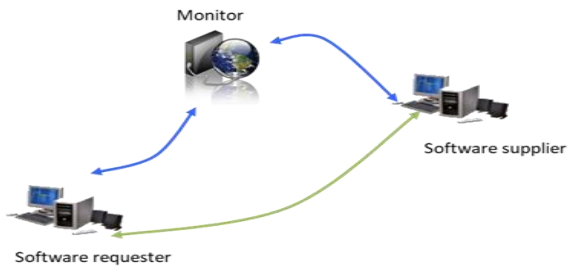


Figure 2. Solution entities

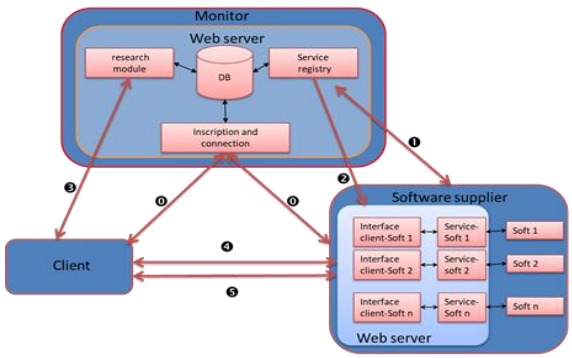


Figure 3. Global P2P architecture

3.1 Exchange process description

The following section outlines the various exchanges that occur between the three entities:

3.1.1 Registration and login

Operation 0, as shown in Figure 3, involves user registration and login to the system. The primary objective of this step is to update the @IPs of software suppliers so that requested

services can be located. Since users can have different @IPs across sessions, it is essential for the Monitor to maintain accurate information on the addresses of listed services at all times.

3.1.2 Publishing shared software

Operation 1 and 2, as depicted in Figure 3, enable software suppliers to edit and publish their software on the monitor to be shared with others. This process is carried out through a web interface provided by the monitor, which software suppliers can access. The web interface (form) allows suppliers to define the software they wish to share and its location path on the local drive. They can also provide a description of the software's capabilities and specify the parameters and arguments required to run the software. This information is used by the indexing module (system registry), which creates two programs (scripts) to be installed in the software provider. One of the scripts is the web interface used by the client to request the service, while the other is the script that runs the shared software and sends the results back to the client, as shown in Figure 4.

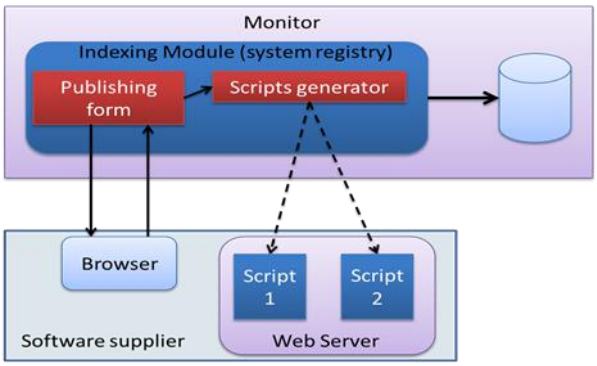


Figure 4. Publishing step

3.1.3 Searching for software

This step (Operation 3, Figure 3) enables clients to search for desired software by entering a description or browsing a pre-determined list through a web page available on the main site of the monitor. The search result is presented as a web page containing direct links to the web pages created during the publishing of services by the software supplier. When the client clicks on the link, he is redirected to the web page associated with the requested service at the software supplier, resulting in a direct interaction between the client and software supplier, as shown in Figure 5.

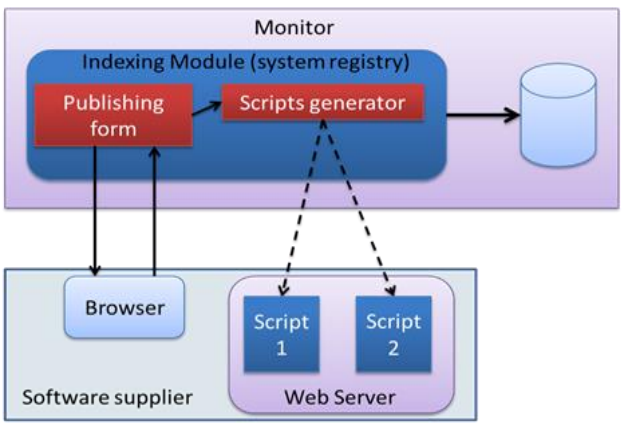


Figure 5. Research step

3.1.4 Service request

During this step (Operation 4 and 5, Figure 3), the client and software supplier establish a direct connection. After the client selects the desired service, he clicks on the link provided by the software supplier, which allows him to download the appropriate web page form automatically created by the monitor during the publishing step. The web page form prompts the client to input the data to be processed and select the required parameters, if any, to run the software. The software supplier defines the possible parameters during the publishing step.

After receiving the data from the client, the second script (which was created by the monitor during the publishing step) is executed. This script launches the software, retrieves the results, and generates a dynamic web page that includes a link to the results. The link is then returned to the client, who can use it to download the results (as illustrated in Figure 6).

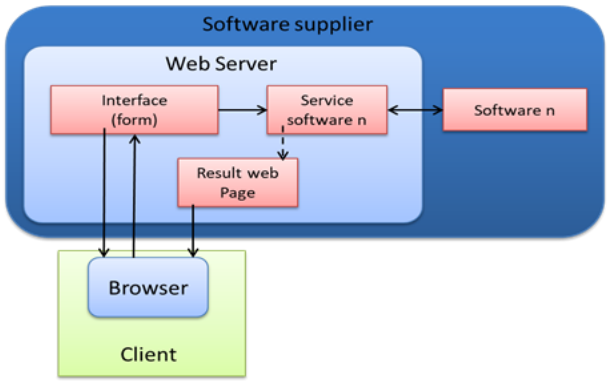


Figure 6. Service request step

3.2 Forms and data

A centralized database at the monitor level consisting of a set of records is used to store information on all shared software. Each record (Figure 7) contains essentially the designation of the software, its category (for easy searching), its description, @ the IP address and port of the software supplier's web server, and the designation of the web page: interface between the client and the software supplier (the link). All this information is collected during the publishing step.

ID	Software designation	Category	@IP	PORT	Link	Description	state

Figure 7. DB record

There are three main forms:
1) publishing form;
2) research form;
3) service request form.

The first two are part of the web application hosted in the monitor while the latter is dynamically created at the software supplier during the publishing step, and is associated with the shared software (for each shared software, a form is created).

During the publishing phase, the software supplier contacts the monitor by requesting the "Publishing Form" web page. This form asks the software supplier to inform some fields. Then, the script generator (Figure 8) generates automatically with the data extracted from the publishing form, two files

(HTML, PHP) which will be saved in the www directory of the software supplier's web server:

1) HTML file (software execution form): interface between the client and the software supplier, the file is saved in the software supplier as a web page.

2) PHP file: which launches the selected software, also registered in the software supplier.

The software supplier must have a web server to run both scripts (PHP and HTML).

The XML file (Figure 9) is an intermediary between the publishing form and the script generator.

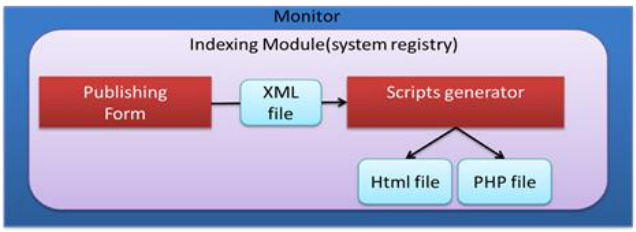


Figure 8. Indexing module details

The XML file's tags are descripted in Table 1.

Table 1. Script generator XML file's tags

Tag	Description
Software	The beginner of the file
Designation	Indicate the line command used to launch the software
Locate	The path of the software in the software supplier drive
Parameters	Indicate the beginning of the parameters used to execute the software
Input	Indicate the input section
Output	Indicate the output section
Argement_nbr	a value that indicates the number of arguments (1...n).
Argement_type	To indicate if the argument is of type "file" or a simple "value"
Attribute_nbr	A value that indicates the number of attributes (1…n)
Attribute	The attribute section
Attribute_name	Attribute designation (name)
Attrib	To indicate the attribute information
Forced	To indicate if the attribute is optional or required
Value	If the input or the output is just a single value
Type	To indicate if the type of output is a file or a simple value

The PHP script's main task:

1) Launch the corresponding software with the settings defined in the XML file. The launch is executed by a line command done by the PHP instruction "string exec (string command [, array & $ output [, int & $ return_var]])."

2) Generate dynamic HTML file that allows the client to retrieve the results of treatment.

In such a scenario of software search and invocation, the client contacts the monitor by requesting the "Software Search Form" web page. This form asks the client to define some keywords that will allow the monitor to find the adequate software. Once this is done, the monitor responds with the following web page which consists of a list of possible proposals. The client has only to choose one of the proposed software by clicking on the corresponding link to be redirected to the HTML web page of the software supplier "Software Execution Form". At this moment, the monitor is no longer in Part, the interaction is solely between the client and the software supplier. Once the treatment is finished, the software supplier answers the client via a web page inviting him to download the result of the treatment.

```
<software>
        <designation>....</designation>
        <locate>....</locate>
        <parameters>
                <input>
                        < argements_nbr>....</ argements_nbr >
                        < argement_type> ....</ argement_type >
                          .
                          .
                          .
                        < attribute_ nbr>.... </ attribute_ nbr>
                        <Attribut>
                            <Attribut_name>....</Attribut_name>
                            <forced>....</forced>
                            <value> ....</value>
                        </Attribut>
                          .
                          .
                          .
                </input>
                <output>
                        <type> ....</type>
                </output>
        </parameters>
</software>
```

Figure 9. XML file used by the script generator

3.3 Service composition

Sometimes the treatment requested by a client cannot be provided by a single software supplier. In this case, we need to use multiple servers that will succeed to achieve the desired result. In this case, the monitor makes research by needs. The client makes a request stating what he has as input data and what he wants as output result.

The monitor will establish as far as possible, a sequence of software suppliers that will carry out the treatment.

Indeed, Software is seen as a black box that has data as input and result as output. A sequence is viewed as a succession of software where the outcome of one is the input of the other (Figure 10).

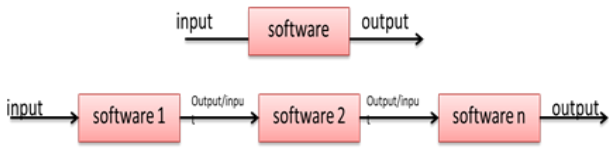


Figure 10. Succeeded treatment

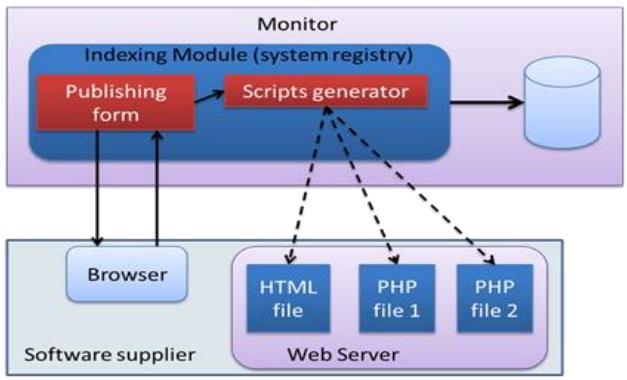


Figure 11. A second PHP script created in the software supplier

To allow a client to beneficiate of this kind of process, a new research interface is defined in the monitor web application that will allow the client to do research by "Need". This solution is only possible if the software suppliers publish their software without any parameters except the input and output data parameters and specifying the nature of each parameter. During the publishing step, the monitor provides the software servers the opportunity to participate in a sequential treatment, if this is the case; another script is created in the software suppliers (Figure 11).

This script will allow an involved server in a sequential treatment, to retrieve an XML file transmitted by the previous server for treatment, and then send a modified version of the XML file towards the next server (Figure 12).

The monitor has a module result by which it stores the results obtained after sequential treatment. In fact, the last server cannot communicate with the client for the simple reason that the client has no way of contact. So, the result is transmitted to the monitor who will save it until the client requests it.

Table 2. Sequential Treatment XML file's tags

Tag	Description
Treatment	Indicates the section treatment for each server involved
Data	Section where to put the input data and the output result
Command	The command to launch the software
Next-server	The next software supplier to invoke
Client	Indicates the client Id that did the request

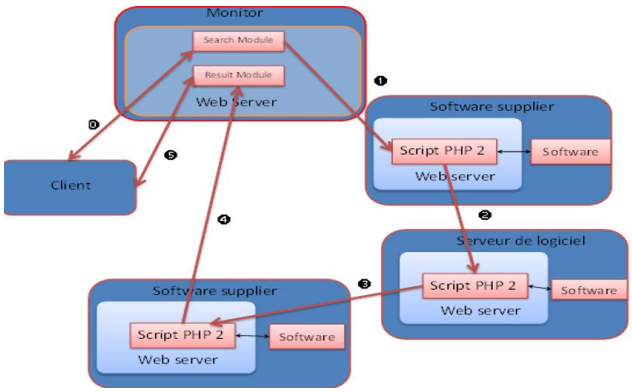


Figure 12. Sequential treatment

Sequential treatment process:

When the client chooses to make a need-based request application, he interacts with the monitor by invoking the web page "Software Search Form (Sequential Processing)". This interface provided by the monitor allows the client to declare what he has as input data and what he wants as result data (output). The monitor looks in its database at the opportunity of making this request and establishes various sequential processing scenarios depending on server's profiles (to be examined later). It creates a new interface (form) to ask the client to choose between several solutions that can be classified according to certain criteria (not covered in this study). Once the client has chosen the solution in the search result interface and has provided the data file, the monitor starts sequential processing by creating a structured XML file, as shown in Figure 13, with a "treatment" entries tag (Table 2) for each server visited.

```
<treatment>
      <data>.....</data>
      <command>.....</command>
      <next-server>.....</next-server>
</treatment>
.<treatment>
      <data>.....</data>
      <command>.....</command>
      <next-server>.....</next-server>
</treatment>
...
<treatment>
      <data>.....</data>
      <command>.....</command>
      <next-server>Monitor</next-server>
</treatment>
<treatment>
      <data>.....</data>
      <command>END</command>
      <client>ID_Client</client>
</treatment>
```

Figure 13. Sequential treatment XML file

The monitor starts integrating the client data into the first "data" tag and the command to launch the software (software name) into the first command tag, this to be executed by the first server and the address of the next server. It did the same thing for the following servers, except that the data tags are empty. In the last server "treatment" tag, the monitor indicates that the next server is the monitor itself.

At the end of the file, it inserts a treatment for himself where it is stated: the END command (end of treatment) and client ID to associate the data result to the client that made the request.

Each software supplier, when it receives this XML file, look for the first "treatment" tag, it completely removes the tag from the file after pulling data to be processed, the software to run and @IP of the next software supplier. After treatment, it incorporates the results at the first "data" tag. The modified XML file is sent to the next software supplier, the Figure 14 illustrates this process.

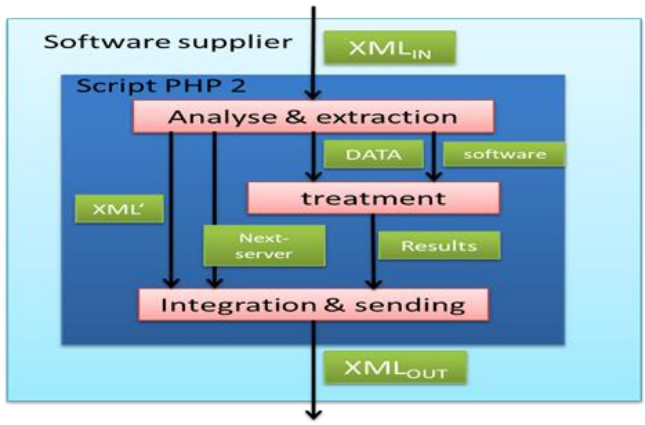


Figure 14. The process of analyze and extraction in a sequential treatment at a software supplier level

The result module (Figure 15) is composed of two units; the first one consists of analyzing and extracting data from the XML file received from the last software supplier involved in the sequential treatment. From this file, the result data and the client ID who did the request are extracted, and then saved in the monitor database until the client reaches them. The second

unit consists of an HTML interface that allows the client to reach his results.

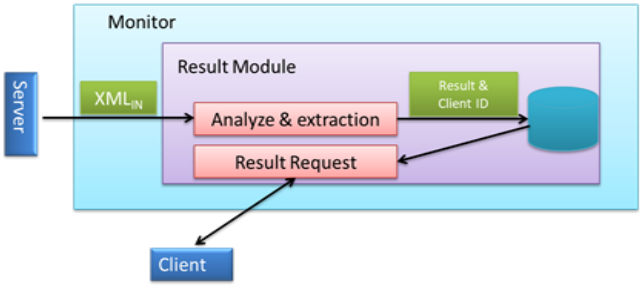


Figure 15. Monitor result module

The client is not notified that the treatment result is ready. He has to do it by himself by invoking the monitor's webpage "Result research (sequential treatment)". This webpage indicates the client while he introduces his ID if the treatment that he requested is completed or not yet. If it is the case, he can download the result by clicking the corresponding link.

4. EVALUATION AND ANALYSIS OF RESULTS

In this section, we outline the experimental setup and methodology employed to assess the architecture introduced in Section 3.

4.1 Experimental setting

We have implemented a fully automated web system where users no longer need to program; all is assisted by HTML coded interfaces. The PeerSim simulator is used as an experimental framework.

The PeerSim simulator is widely utilized for simulating large-scale dynamic P2P networks [27]. It is capable of simulating both structured and unstructured overlays comprising millions of nodes [28]. PeerSim, written in Java, was developed in part as part of the BISON project and is available under the open-source GPL license [29]. PeerSim supports two simulation paradigms: a cycle-based simulation, where network nodes are randomly selected, and each node protocol is invoked at each cycle [27, 30], and an event-based simulation, where a set of events is scheduled over time, and node protocols are invoked according to the order of message delivery time [28]. In PeerSim, the network is conceptualized as a collection of nodes, where each node possesses a fixed identifier and a set of protocols accessible through the node interface. The simulation incorporates initializers and controls [27, 28]. Initializers are executed prior to the simulation, while controls are executed during the simulation. These components implement the Control interface and can modify or supervise the different nodes during the simulation [28], the collected statistics can be formatted and sent to a standard output [27].

The proposed prototype underwent extensive evaluation through numerous simulations utilizing the PeerSim cycle-based model. The simulation lifecycle follows a defined sequence: Initially, a simple ASCII configuration file is read, comprising key-value pairs that encompass all simulation parameters for the experiment's objects [27]. Subsequently, the simulator configures the network by initializing the network nodes and protocols. Network nodes and protocols are instantiated by cloning using the "clone()" method of the "Node" class. Essentially, a single instance is constructed using the object's constructor, serving as a prototype from which all network nodes are cloned. The initialization phase is carried out by control objects, whose execution is scheduled only at the outset of each experiment. Following initialization, the cycle-driven engine invokes all components (protocols and controls) once per cycle until the simulation concludes [27].
In our experiments, we employed a test configuration with a fluctuating network size: 10,000, 20,000, 30,000, 40,000, and 50,000 nodes. The experimental framework does not account for the composed treatment scenario. To adapt our solution to the PeerSim simulator, we defined a set of classes outlining the behavior of the primary actors in the experiment:

The central node (Monitor): is considered as the main node of a hybrid p2p network and acts as an intermediary between the different peer clients. It is embodied by a class encompassing the global publishing, search, and result sending modules. This class implements the Control interface, with "execute()" serving as the primary method to disseminate the central node's address to every peer in the network. The Control is initiated solely once during the initialization phase. This class makes use of the following data structure:

Global list of services: This static class comprises the names of services along with their input and output parameters. It acts as a reference point for peers to select a (random) number of services they will either share or request. Accessible by all peers, it ensures consistency and accessibility throughout the system.

Global list of published services: This list contains all services published by all peers and their identifiers. It is located in the central node.

Peers (clients): Represented by a class, peers encapsulate various modules (methods) enabling them to execute different tasks such as communicating with the central peer, local and central publishing, service search, service request, and result sending. This class implements the "CDProtocol" interface inherited from the Protocol class, incorporating the "nextCycle" method() where we define the diverse tasks a peer must undertake during its execution. Our solution encompasses four potential scenarios, each with an equal probability: 1) publishing of services, 2) searching for services, 3) deletion of services, and 4) peer inactivity. Each scenario holds a probability of 0.25. This class relies on the following data structure:

Local list of services: Generated from the global list of services, it is unique to each peer. It is from this list that the peer will choose the services he wants to share.

Local list of published services: this list contains the entire shared services specific to each peer.

Observers: We represent performance indicators through classes designed to extract the necessary measurements (statistics) for analyzing and comprehending the behavior of our network. Throughout each cycle, measurements are captured and analyzed at the simulation's conclusion. We have concentrated on five key performance indicators:

1) Success search: Indicates a positive value when the central node discovers at least one peer publishing the requested service during the search process.

2) Success result: Reflects the successful return of **results** by suppliers following the execution of requested services.

3) Number of resource nodes: Represents the count of **nodes** publishing the requested service.

4) Failure result: Denotes the inability of supplier nodes to

correctly deliver the requested service.

5) Failure search: Occurs when the central node fails to find any service provider during the search for a requested service.

Indicators Meaning:

1) Success search: This implies that as peers contribute more resources, the collaboration within the network becomes more accessible, and peers can better fulfill their needs. In such a scenario, selfish behavior is absent and does not impact the system's behavior.

2) Success result: This implies that the peers have finished the service processing correctly. This means that more this indicator is significant, more the system work without failures. Also, it means that the services offered by the suppliers are functioning correctly.

3) Number of resource nodes: More this indicator is significant more there are peers who want to contribute to the system with the same resource. This means that this resource have a large probability to be available. In other terms, we can utilize this indicator to study the issue of resources availability and profile the supplier peers.

4) Failure search: This indicator is the opposite of the first one (Success search). This implies that more this indicator is bigger, less the collaboration within the network is accessible, and consequently, peers cannot fulfill their needs. In such a scenario, selfish behavior is present, resulting in a negative impact on the system's behavior.

5) Failure result: This means that the service does not delivery the result after invoking it, due to implementation issues, data type or service declaration. More it is significant, more the system is useless.

4.2 Result and discussion

Utilizing the PeerSim cycle-based model, we conducted numerous simulations of the proposed prototype. The forthcoming discussion will revolve around the metrics introduced earlier.

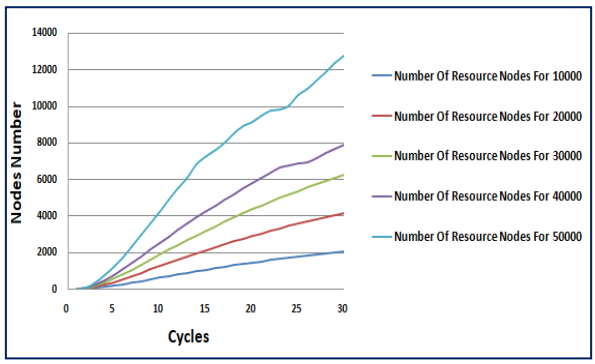
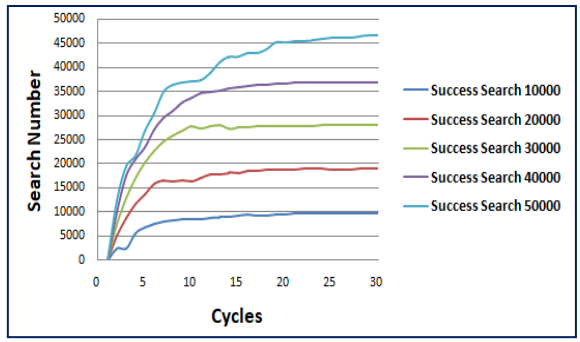


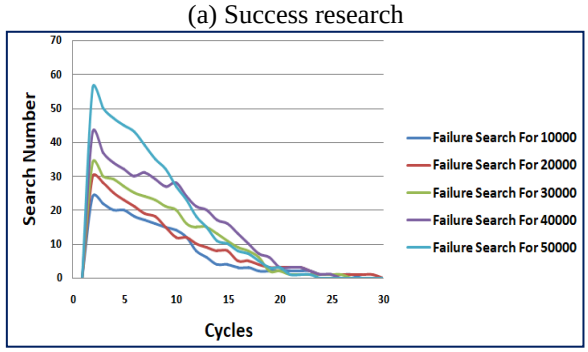
Figure 16. Number of resource nodes

Figure 16 illustrates the fluctuation in the quantity of nodes possessing the requested services per cycle throughout the simulation duration. By examining the graph depicted in Figure 16, it becomes evident that the number of nodes offering demanded services has notably surged over time. This trend is attributable to clients actively participating in service publication and transitioning into service providers within the system. Consequently, this expansion is poised to positively influence the research process.

Figure 17 depicts the variation of two metrics, namely Success_Search and Failure_Search, over the simulation duration. Analysis of the statistics reveals that Success_Search consistently exceeds Failure_Search throughout the entire simulation period. Moreover, the results indicate a continuous increase in the Success_Search metric over time, which is a direct outcome of the concurrent increase in the number of nodes possessing the resource. Notably, this metric begins to stabilize from Cycle 17. Conversely, the Failure_Search metric experiences a slight rise at the initial stages of the simulation due to fewer service suppliers. Subsequently, it declines from Cycle 3 and gradually approaches zero as the simulation progresses.

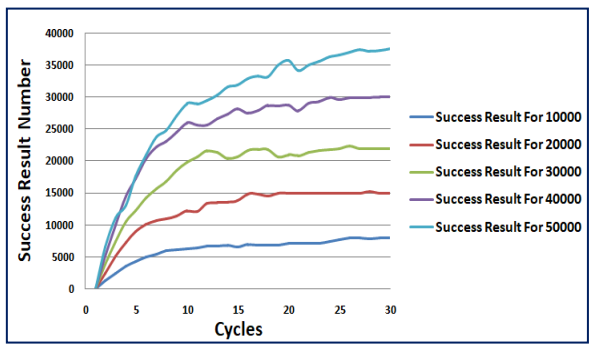


(a) Success research

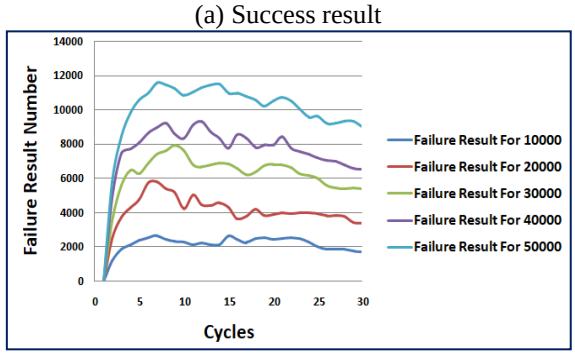


(b) Failure research

Figure 17. Success research vs. failure research



(a) Success result

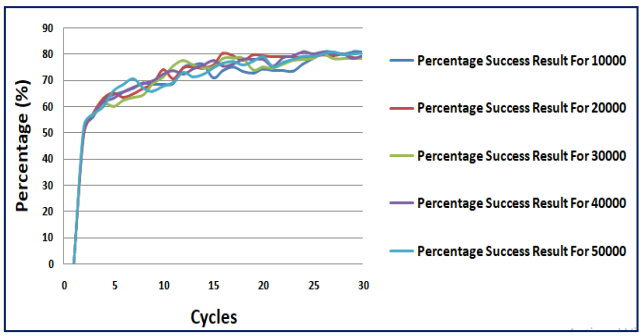


(b) Failure result

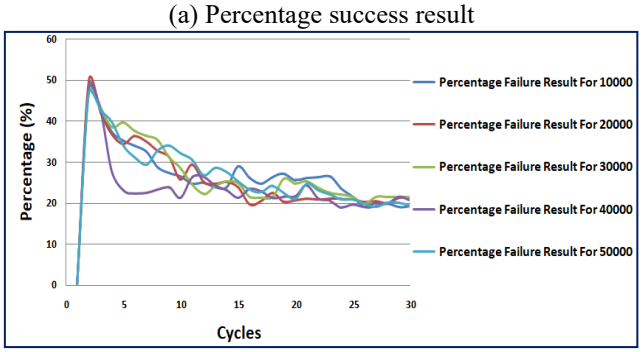
Figure 18. Success result vs. failure result

Figure 18 illustrates the variation of two pivotal metrics, Success_Result and Failure_Result, throughout the entire simulation period. These metrics serve as crucial indicators reflecting the system's ability to ensure resource availability. Analyzing these figures, we observe that Success_Result experiences continuous growth, peaking at the 25th cycle before plateauing until the simulation's conclusion. This trend is juxtaposed with the variation of Failure_Result, which initially rises until the 8th cycle before exhibiting minor fluctuations between increases and decreases, eventually stabilizing towards the simulation's end with occasional decreases. Furthermore, it is evident that the results of Success_Result consistently surpass those of Failure_Result by a significant margin.

Figure 19 illustrates the evolution of the percentage of Success_Result and Failure_Result compared to the total requests made throughout the entire simulation period. In the initial 3 cycles, a notable surge in the percentage of Success_Result is observed, peaking at 50%. Concurrently, there is a rapid escalation in the percentage of Failure_Result, also reaching 50%. Beyond the 3rd cycle, the percentage of Success_Result continues to increment gradually, reaching a maximum value of 80% before stabilizing until the simulation's conclusion. Meanwhile, the percentage of Failure_Result diminishes over time, displaying a trend toward stabilization at around 20%.



(a) Percentage success result



(b) Percentage failure result

Figure 19. Percentage success result vs percentage failure result

5. RELEVANT POINTS

5.1 Collaborations incentives

In collaborative systems, efforts are often directed towards implementing mechanisms to enhance collaboration. Several researchers have delved into the study of incentive mechanisms for collaboration. For instance, to establish Grid computing as a feasible business model, investigations have focused on formulating and implementing economic models and algorithms essential for fostering widespread adoption in commerce and industry [31]. Some researchers advocate the incorporation of market mechanisms to determine resource prices [32].

When peer-to-peer systems, are more like social systems, the researchers define simple rules of contribution [31]. Those who are contributing to a common fund may have access to this fund.

In the realm of incentive research, two distinct techniques have emerged: soft incentives and hard incentives. Soft incentives rely on a reputation system where peers receive positive ratings based on their reliability, contribution, effort, etc., resulting in better quality of service and higher priority access to resources. Conversely, newcomers or those with limited interaction receive lower ratings and develop poor reputations [33, 34]. On the other hand, the hard incentive system advocates for the implementation of external billing, individual and collective invoicing, and micro-currency approaches to incentivize resource sharing [35-41].

As we are currently implementing a collaborative P2P system, we recommend the unitization of soft incentives techniques by adopting the reputation system; we consider that it is most suitable for a P2P network.

We can establish at the monitor a reputation's computation module that consists to evaluate the degrees of contribution of the peer. Whenever a software supplier peer agrees to process data of another, his reputation is increased. If it ever happens to him to seek treatment at another peer, the monitor would recommend it to another peer and he will have the priority of being served before others. a peer that does not contribute to the system may face rejection of service or receive lower priority. The recommendation made by the monitor to a peer may be a code that the monitor sends with the link that the client must follow in step search and invocation. The software supplier, upon receipt of this code, may accept or decline the request of the client. A method of evaluation can be integrated into the monitor system to know if the software supplier has indeed done its job.

In this case, according to the peer’s reputation’s evaluation the monitor can block

5.2 Security

In P2P networks, file sharing stands out as the most common operation. However, a significant security concern arises when downloading files from other peers: the uncertainty regarding the authenticity and integrity of the file. While you may believe you are downloading a valuable utility, there's a risk that the downloaded file contains malicious content, such as a Trojan or backdoor, which could grant unauthorized access to your computer. This issue is particularly prevalent when downloading executable files. In our system, this problem is mitigated since the data transferred are non-executable and will never be executed by the software supplier.

Given that the system is entirely automated and peer exchanges involve only data, there is no risk of transmitting malicious data. Even if peers attempt to send malicious data, we can incorporate a detection mechanism, such as an antivirus, to prevent such situations.

The primary security concern revolves around data confidentiality. The key question is whether a peer can access the data he has received for processing. For an inexperienced

peer, accessing the received data can be challenging since it is transmitted automatically through the HTTP protocol. However, for an experienced peer, it remains possible to retrieve the data. In such a scenario, we can empower the requesting peer to assess the confidentiality level of their data and decide whether to send it or not.

5.3 Profile use

By profile, we intend to classify peers into categories. Classify clients into a set of categories and software suppliers into another set of categories, and then try to make a correspondence between these two sets. When a client manifests, the monitor is able to recognize his profile (category), which means recognizing his intentions and even deduce the category of software suppliers to whom he corresponds better. This will allow the monitor to restrict his research field and propose solutions that best will satisfy the client's needs. Profiling the peers aims to enhance the system's performance in terms of publishing and retrieving resources, as well as satisfying peer requests. Through the classification of requester peers and supplier peers, we can leverage recommendation systems to suggest specific services to peers and ensure service availability.

6. CONCLUSION

This article presents a SOA-P2P system based on HTML that allows to persons who suffer for any reason, of lack of hardware and/or software resources, achieve their needs by transmitting their data to be treated remotely and return back the results. Our goal is to bring to non-experienced user the means to exploit others' resources with zero programming, therefore bypassing software pirating, heterogeneity problem and lack of resources. We developed a completely automated system where the users no longer need to program. The all is assisted by HTML coded interfaces. The experimental findings have demonstrated that the proposed system can attain a commendable success rate of up to 80%. These results are promising and affirm that the proposed architecture has the potential to serve as a dependable solution to challenges associated with software illegality, heterogeneity, and unavailability. We assert that our proposal introduces novel contributions to the field. Nevertheless, it is evident that a crucial concern lies in understanding the behavior of the proposed system in large-scale scenarios, such as the current Internet. Scalability, for instance, emerges as a pivotal aspect warranting emphasis; a thorough analysis of potential bottlenecks in the system is imperative.

Employing profile notions for sequential treatment can enhance the composition of software suppliers. The sequential treatment can be improved by applying the notion of profile and strategies selection of software suppliers, based on statistics taken during their interaction in the system. These statistics allow the monitor to select peers that are less involved and avoid those most solicited, thus there will be charge equilibrium between peers in the system.

REFERENCES

[1] Turner, M., Budgen, D., Brereton, P. (2003). Turning software into a service. Computer, 36(10): 38-44. https://doi.org/10.1109/MC.2003.1236470

[2] Guo, P. (2009). A survey of software as a service delivery paradigm. In TKK T-110.5190 Seminar on Internetworking.

[3] Sandanayake, T.C., Jayangani, P.G.C. (2018). Current trends in software as a service (SaaS). International Journal for Innovation Education and Research, 6(2): 221-234.

[4] Kumar, K.K.M. (2014). Software as a service for efficient cloud computing. International Journal of Research in Engineering and Technology, 3(1).

[5] Sharma, K. (2018). Software as a service and cloud security. Journal of Computer, 3(2).

[6] Kaur, B. (2015). Software as a service: A brief study. International Research Journal of Engineering and Technology (IRJET), 2(3).

[7] Kulkarni, G., Gambhir, J., Palwe, R. (2012). Cloud computing-software as service. International Journal of Cloud Computing and Services Science, 1(1): 11-16.

[8] Chang, T., Aharnad, M. (2004). GT-P2PRMI: Improving middleware performance using peer-to-peer service replication. In Proceedings. 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. FTDCS 2004, Suzhou, China, pp. 172-177. https://doi.org/10.1109/FTDCS.2004.1316610

[9] Maurya, R.K., Pandey, S., Kumar, V. (2016). A survey of peer-to-peer networks. International Journal of Advanced Research in Computer and Communication Engineering, 5(4).

[10] Pourebrahimi, B., Bertels, K., Vassiliadis, S. (2005). A survey of peer-to-peer networks. In Proceedings of the 16th annual workshop on Circuits, Systems and Signal Processing, pp. 570-577.

[11] Chaudhary, M.N., Surolia, J. (2015). A survey on peer to peer system applications. International Journal of Innovative Computer Science & Engineering, 2(1): 16-20.

[12] Steinmetz, R., Wehrle, K. (2005). Peer-to-peer systems and applications. Springer.

[13] Kisembe, P., Jeberson, W. (2017). Future of peer-to-peer technology with the rise of cloud computing. International Journal of Peer to Peer Networks (IJP2P), 8(2/3): 45-54.

[14] Varma, P.C.V. (2005). Resource sharing. IEEE Potentials, 5.

[15] Schulte, R.W., Natis, Y.V. (1996). Service oriented architectures. Part 1 &2. http://www.gartner.com.

[16] Griffin, D., Pesch, D. (2007). A survey on web services in telecommunications. IEEE Communications Magazine, 45(7): 28-35. https://doi.org/10.1109/MCOM.2007.382657

[17] Hack, S., Lindemann, M. (2007). Enterprise SOA Einführen. Galileo Press. https://opus.bibliothek.fh-aachen.de/opus4/frontdoor/index/index/docId/5287.

[18] Juric, M.B., Loganathan, R., Sarang, P., Jennings, F. (2007). SOA Approach to Integration: XML, Web Services, ESB, and BPEL in Real-World SOA Projects. Edition Packt Publishing Ltd.

[19] Rathod, D. (2017). Performance evaluation of restful web services and soap/wsdl web services. International Journal of Advanced Research in Computer Science, 8(7): 415-420.

[20] Al-Shargabi, B., Sabri, A., El Sheikh, A. (2010). Web service composition survey: State of the art review. Recent Patents on Computer Science, 3(2): 91-107.

[21] Silva, J.N., Ferreira, P., Veiga, L. (2010). Service and resource discovery in cycle-sharing environments with a utility algebra. In 2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS), Atlanta, GA, USA, pp. 1-11. https://doi.org/10.1109/IPDPS.2010.5470410

[22] Foster, I., Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. The International Journal of Supercomputer Applications and High Performance Computing, 11(2): 115-128. https://doi.org/10.1177/109434209701100205

[23] Litzkow, M.J., Livny, M., Mutka, M.W. (1987). Condor-a hunter of idle workstations. University of Wisconsin-Madison Department of Computer Sciences.

[24] Esteves, S., Veiga, L., Ferreira, P. (2010). Gridp2p: Resource usage in grids and peer-to-peer systems. In 2010 IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW), Atlanta, GA, USA, pp. 1-8. https://doi.org/10.1109/IPDPSW.2010.5470917

[25] Mason, R., Kelly, W. (2003). Peer-to-peer cycle sharing via .net remoting. In AusWeb 2003. The Ninth Australian World Wide Web Conference.

[26] Galatopoullos, D.G., Kalofonos, D.N., Manolakos, E.S. (2008). A P2P SOA enabling group collaboration through service composition. In ICPS '08 Proceedings of the 5th International Conference on Pervasive Services, pp. 111-120.

[27] Jesi, G.P. (2005). HOWTO: Build a new protocol for the PeerSim 1.0 simulator. Universität Trento, Italien.

[28] Surati, S., Jinwala, D.C., Garg, S. (2017). A survey of simulators for P2P overlay networks with a case study of the P2P tree overlay using an event-driven simulator. Engineering Science and Technology, an International Journal, 20(2): 705-720. https://doi.org/10.1016/j.jestch.2016.12.010

[29] Naicken, S., Basu, A., Livingston, B., Rodhetbhai, S. (2006). A survey of peer-to-peer network simulators. In Proceedings of The Seventh Annual Postgraduate Symposium, Liverpool, UK, p. 13.

[30] Ebrahim, M., Khan, S., Mohani, S.S.U.H. (2014). Peer-to-peer network simulators: An analytical review. arXiv preprint arXiv:1405.0400. https://arxiv.org/abs/1405.0400

[31] Iyilade, J., Aderounmu, A., Adigun, M. (2007). Incentives for resource sharing and cooperation in grid computing system. In The 2007 International Conference on Next Generation Mobile Applications, Services and Technologies (NGMAST 2007), Cardiff, UK, pp. 191-198. https://doi.org/10.1109/NGMAST.2007.4343420

[32] Buyya, R., Abramson, D., Giddy, J., Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing. Concurrency and Computation: Practice and Experience, 14(13-15): 1507-1542. https://doi.org/10.1002/cpe.690

[33] Vega, D., Meseguer, R., Freitag, F., Ochoa, S.F. (2013). Effort-based incentives for resource sharing in collaborative volunteer applications. In Proceedings of the 2013 IEEE 17th International Conference on Computer Supported Cooperative Work in Design (CSCWD), Whistler, BC, Canada, pp. 37-42. https://doi.org/10.1109/CSCWD.2013.6580936

[34] Vega d'Aurelio, D., Meseguer Pallarès, R., Freitag, F., Ochoa, S. (2014). Understanding collaboration in volunteer computing systems. Journal of Universal Computer Science, 20(13): 1738-1765. http://dx.doi.org/10.3217/jucs-020-13-1738

[35] Ileri, O., Mau, S.C., Mandayam, N.B. (2005). Pricing for enabling forwarding in self-configuring ad hoc networks. IEEE Journal on Selected Areas in Communications, 23(1): 151-162. https://doi.org/10.1109/JSAC.2004.837356

[36] Jakobsson, M., Hubaux, J.P., Buttyán, L. (2003). A micro-payment scheme encouraging collaboration in multi-hop cellular networks. In Financial Cryptography: 7th International Conference, FC 2003, Guadeloupe, French West Indies, pp. 15-33. https://doi.org/10.1007/978-3-540-45126-6_2

[37] Antoniadis, P., Le Grand, B. (2007). Incentives for resource sharing in self-organized communities: From economics to social psychology. In 2007 2nd International Conference on Digital Information Management, Lyon, France, pp. 756-761. https://doi.org/10.1109/ICDIM.2007.4444315

[38] Santos, A., Fernández Anta, A., López Fernández, L. (2013). Quid Pro Quo: A mechanism for fair collaboration in networked systems. PloS One, 8(9): e66575. https://doi.org/10.1371/journal.pone.0066575

[39] Zhang, Y., van der Schaar, M. (2014). Collective ratings for online communities with strategic users. IEEE Transactions on Signal Processing, 62(12): 3069-3083. https://doi.org/10.1109/TSP.2014.2320457

[40] Rahman, R., Meulpolder, M., Hales, D., Pouwelse, J., Epema, D., Sips, H. (2010). Improving efficiency and fairness in p2p systems with effort-based incentives. In 2010 IEEE International Conference on Communications, Cape Town, South Africa, pp. 1-5. https://doi.org/10.1109/ICC.2010.5502544

[41] Cohen, B. (2019). The BitTorrent Protocol Specification, http://www.BitTorrent.org/beps/bep_0003.html.