EAI Endorsed Transactions on Scalable Information Systems Research Article **EAI.EU**

A Scalable Architecture for Secured Access to Distributed Services

Telesphore Tiendrebeogo ,*

Polytechnic University of Bobo, Department of Mathematics and Computer Science, 01 BP 822 Bobo-Dioulasso 01, Bobo-Dioulasso, Burkina Faso

Abstract

Web services which are an implementation of Services Oriented Architecture (SOA), are emergent technologies and promising the development, the deployment and the integration of Internet applications. They are initially based on three main layers that are SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language) and UDDI(Universal Description, Discovery and Integration). The used language which underlies these protocols is XML (eXtensible Markup Language), what returns Web Services independent from platforms and from programming languages. Web service discovery is a procedure which consists to obtain descriptions of Web services based on the requirements functional or not functional. In this context, we propose, new architecture which replaces SOAP and UDDI to offer a secured discovery and access to the Web services. The unified protocol allowing the secured discovery and access to the Web services is called: Secured Access Protocol for Distributed Services (SAPDS). It is based on a Distributed Hash Table (DHT) defined in the hyperbolic space from the Poincaré disk model. However, users keep obtaining beforehand a WSDL file associated with a key of access to the service. This paper presents SAPDS as an intermediary third to store and discover Web services by using a Service of Storage and Discovery Tree (SSDT).

Received on 28 March 2017; accepted on 23 April 2017; published on 29 June 2017

Keywords: Web service, UDDI, Hyperbolic Tree, Distributed Hashing, Storage, Lookup, Intermediary Third.

Copyright © 2017 Telesphore Tiendrebeogo, licensed to EAI. This is an open access article distributed under the terms of the Creative Commons Attribution license (http://creativecommons.org/licenses/by/3.0/), which permits unlimited use, distribution and reproduction in any medium so long as the original work is properly cited.

doi:10.4108/eai.28-6-2017.152751

1. Introduction

Nowadays, Web is not any more simply an enormous warehouse of text and images, its evolution made that it is also a service provider. The notion of "Web service" indicates essentially an application put at the disposal on the Internet by a service provider, and accessible by the customers through standard Internet protocols. Service Oriented Architecture is an architectural concept which is used to build infrastructures allowing to put in relation of the entities with needs (Consumers) and those with capacities of sharing (Suppliers). This interaction is made via services through the varied technological domains. Services act as the main facilitator of the digital data exchanges, but require additional mechanisms in order to function. The basic SOA architecture concerns different processings , such as service request, service discovery service invocation and service organization (storage and Orchestration). The service lookup success depends largely on the effective analysis of the query request and on the discovery of web service interfaces that facilitates the discovery of Web service. Web service discovery is the process of location, or discovery of one or several bound documents which describe an XML detail of the Web service using the WSDL. At the beginning, the Web services are implemented through three standard technologies: WSDL, UDDI and SOAP. These technologies facilitate the description, the discovery and the communication between services. However, this basic infrastructure does not still allow the Web services to keep their promise of a widely automated management [1]. This automation is nevertheless essential to face the requirements of passage in the scale and the will to reduce costs of development, maintenance of the services and secured access. Fundamentally, it has to adapt of a way to describe the Web services in

*Telesphore Tiendrebeogo. Email: tetiendreb@gmail.com an understandable way by a machine and secured. Many approaches and models have been proposed that describes different methodologies for efficient discovery of web services using Web Ontology Language for Web Services (OWL-S) [2], language semantics [3]. Many of them actually don't talk about a framework for the discovery mechanism [4][3][5][6]. Kabir et al. [7] presents different results concerning a conditional purpose-based access control model with dynamic roles, but this system is not very automated and scalable. Wang et al. [8] have proposed a flexible payment system that is scalable and permits privacy, however, their works are particularly adapted to the specific domain of e-commerce, but not the general service distribution. We propose in this paper a new system based on Distributed Hash Table (DHT) which permit to ensure a secured service access. This system allows, not only the discovery of the services via a directory based on a structure of hyperbolic tree (our DHT) on one hand, but also guarantees a distributed storage of the services. This approach, proposes an intermediary third, who is a structure of hyperbolic tree which participates in the decoupling between the consumer and the provider. A DHT is used to improve the safety of the accesses in service. In the following of this paper: we shall present in Section 2, the related works to the Web service discovery. Section 3, is an overview of basic principle and functioning. Section 4, shows new architecture of secured management of web service. In Section 5, we present the Poincaré disk model. Next, Section 6, we show the mechanism of web services routing using a specific algorithm. Then, in the Section 7, we shall present the principle of naming and binding of web services and services server. In Section 8, we present some results of simulation which shows the performances of our structure with regard of different strategies. Finally, in Section 9, we shall conclude our works and shall present our perspectives.

2. Related work

There is since a few years, a lot craze around the web services considered as software which offers services to clients. In [4] it is proposed that Dynamic web Service invocations and hence selection are not independent operations but rather composite in nature and impose some form of ordering thus proposes that reliability is an issue for a composite web service invocation. It uses a FSM(Finite-State Machine) approach to draw the order among the operations in a given WS. The work mainly addresses the reliability issue of a composite web service selection, given a set of available web services.

In [9] three layers of services represents an architecture for a transparent access to the underlying Web service by a service client is presented.An eXtended SOA architecture (XSOA) allows to support capacities as the services orchestration, the "intelligent" routing, the provisioning and the services management. It explores an easy description of the metadata exchanges for the discovery of service description. An extension of this architecture concerns the work which we present here.

In [3] In [20] queries semantics analysis is based on a technique of tree data structure exploration and allows to discover the Web services by assigning weighty values to every node of the tree. Based on the assigned weights, the similarity of semantics is calculated between the web service requested, and web service registered. It presents a methodology to identify the most similar web services, but not the most appropriate services. Furthermore, web services quality and the service level agreement remains an area of concern for the requester.

In [5] a keyword based approach is implemented triggered by the partitioning approach that is used in database design. The idea is used to cluster relevant and irrelevant services according to the user query which in turn helps the user to relieve itself from the burden of selecting web services from a huge set. The key approach is in the group the services in a group of learnt latent variables which is realized by calculating the probability. But a simple keyword is associated in semantics of the query.

In [2] a model of web service discovery, based on descriptions of semantics abstracted from web service and light, using the ontology of service profile of OWL-S is focused. The Purpose is to determine first group of web services, called candidate services then a specific request will allow afterward a fine grain discovery. Here, is proposed the matchmaking algorithm who uses techniques of correspondence with object allowing the recovery of web services based on relations of sub-supposition and the structural information of ontologies of OWL in whom, we proceed by the exploitation of classification of web services in taxonomies of profile, executing the discovery dependent on domain. No structured executive of discovery suited is defined.

In [6] a neural network based approach is used that is best known for their ability for generalization. A novel neural network model is proposed that is used for classification of most suitable web services.

The UDDI specification [10] supplies the survey bound to no QoS in the interface of discovery. The applicants of service cannot filter the disqualified service they cannot obtain either and compare between different services without testing them at first. To solve this problem, a little of work was made to improve the interface of inquiry/publish some recording UDDI to associate the information of QoS in the message. For example, UDDIe project [11] is targeted mainly towards the QoS-supported interface enhancement for UDDI. UDDIe extends the UDDI registry to support searching on attributes of a service and develops the find method to enable queries for UDDI with numerical and logical (AND/OR) ranges.

CSG model [12], the purpose of the model of Cooperating Server Graph model (CSG) is to optimize and dynamically to manage links between cooperative servers over a wide area network. Based on the model of CSG, we extend it in the arena of Web services. A general federal service was conceived and cooperated in our system of extension UDDI as the layer of distribution of message. The approach CSG uses a minimal weight spanning tree to optimize links between ergonomics servers automatically. The shortest trees of path are not chosen to avoid the generation forest of trees. The weight is defined by a distance function that represent the cost of communication enter the couple of nodes (e.g., the inquiry latency , the hop number, etc.).

Prim [13] algorithm is used to calculate the minimum poid of the tree. In our general model of federal service, every node in CSG is called a Federated Server (FS). The model of CSG and the tree of distribution have to adapt itself dynamically to the change of cooperative servers and the underlying topology of network. To be more effective for the management of the graph and reduce the control overhead, different control events are accordingly handled.

In [14], we propose an algorithm corresponding to structure based on the concept of distribution of similarity. The strategy works on the plans which are converted in labelled graphs.

The distribution in the graph is executed as the flood makes for packages IP in the communication of broadcast. This strategy was implemented and tested in Rondo [15].

3. Basic principle and functioning

The interweaving of these three standards (WSDL, SOAP, UDDI) in the discovery and use of services can be schematize in the following sequence diagram.

In Figure 1, the directory UDDI stores the descriptions of services interfaces and gives them to consumers who make it the request. The directory is present as no central access for every consumer, and it is thanks to it that a consumer will discover not only the functions structure exposed by the service, but also the physical location of this one. It is, normally, also thanks to the directory that a web service architecture can insure the mechanisms of services dynamic discovery. The first exchanges between the directory and the future service consumer take place on the basis of a request of document WSDL, herself packed in a document SOAP.

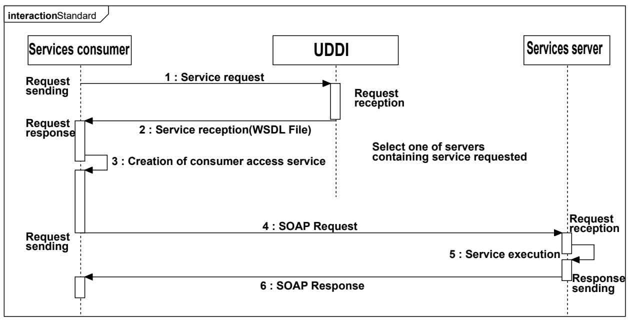

Figure 1. Standard Web services access via SOAP, UDDI, WSDL.

The directory puts back the document WSDL to the consumer, who creates locally a function necessary for the remote invocation from the service. This operation was made, the service invocation is possible: any call of one of the service functions activates the generation of a SOAP message, a messenger directly to the service, which sends back in return the result of its execution to the consumer. The directory is used only to reach the description and the service interface.

This classic architecture of the Web services has a services directory (UDDI) which by default is not scalable, the mechanism which directly in relation, the service consumer and the service provider after service discovery, creates a security's failure. Besides, without the security of the Web Services, the SOAP message is sent to clear text and the personal information such as ID user or the account number is not protected. In this context, we propose a new architecture. We shall see after how the use of the services directory based on the hyperbolic tree manages the exchanges between the services consumer and the service provider allowing thus the construction of a real SOA, by by-passing the limits of the direct access. The hyperbolic tree is in this context an element associated to the SSDT. To understand better the progress of our mechanism, we are going to present at first the Poincaré disk model which is the model on which bases our system of secured access to the services.

4. Architecture of services' secured management

In the architecture which we propose a structure of the hyperbolic tree called intermediary third. our structure of hyperbolic tree (build on hyperbolic space) based on a Poincaré Disk model allows to store a pair (key, value) concerning the referencing of the services as the following Figure 2 shows it. This mechanism, allows to guarantee a secured access to the Web service providers.

Our system must have following properties:

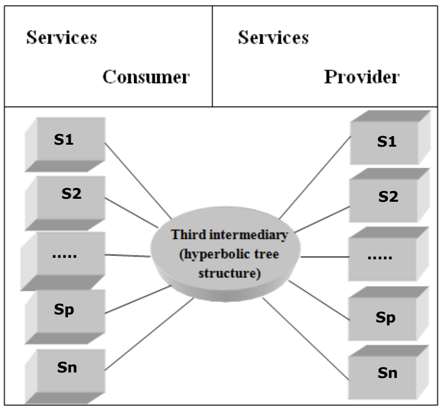

Figure 2. Service access's architecture.

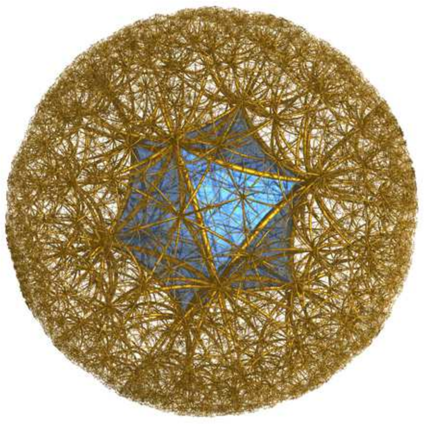

Figure 3. Poincaré "ball" model view of the hyperbolic regular icosahedral honeycomb {3,5,3} [21].

- A guarantee of answer and unique answer to a request on behalf of a consumer of service. It is going to be translated into particular by the introduction of technology of asynchronous transport, as it is custom to meet them in the subjects bound of the integration EAI (Enterprise Application Integration).
- A capacity of load increase and fault tolerance, assured by a distribution of directory on all the points of the network (Or nodes) where the system is activated.
- A management of the safety of the data and the exchanges (encryption, certification, non-rejection), in particular in a context of exchange with partners, the services, as well as we also saw it, to be exposed to the outside world.

5. Poincaré disk model

5.1. Concept and Metrics properties of the Poincaré disk model

In geometry, the Poincaré disk model, also called the conformal disk model, is a model of 2-dimensional hyperbolic geometry [16][17][18] in which the points of the geometry are inside the unit disk, and the straight lines consist of all segments of circles contained within that disk that are orthogonal to the boundary of the disk, plus all diameters of the disk. Along with the Klein model [19] and the Poincaré half-space model [20], it was proposed by Eugenio Beltrami, who used these models to show that hyperbolic geometry was equiconsistent with Euclidean geometry. It is named after Henri Poincaré. The Poincaré ball model is the similar model for 3 or n-dimensional hyperbolic geometry in which the points of the geometry are in the n-dimensional unit ball. Figure 3 is a Poincaré ball model in 3-dimensional hyperbolic geometry.

Hyperbolic Straight lines consist of all arcs of Euclidean circles contained within the disk that are orthogonal to the boundary of the disk, plus all diameters of the disk.

Distances in this model are Cayley-Klein metrics [22]. In mathematics, a Cayley-Klein metric is a metric on the complement of a fixed quadric in a projective space is defined using a cross-ratio. Given two distinct points p and q inside the disk, the unique hyperbolic line connecting them intersects the boundary at two ideal points, a and b, label them so that the points are, in order, a, p, q, b and $|aq| > |ap|$ and $|pb| > |qb|$. The hyperbolic distance between p and q is then $d(p,q) = log\frac{|aq||pb|}{|ap||qb|}$.

The vertical bars indicate the Euclidean length of the line segment connecting the points between them in the model (not along the circle arc), the log is the natural logarithm. Metrics associated with Poincaré disk model can be presented as follows. If u and v are two vectors in real n-dimensional vector space $\mathbb{R}^n$ with the usual Euclidean norm, both of which have norm less than 1, then we may define an isometric invariant by:

$delta(u,v) = 2\frac{|u-v|^2}{(1-|u|^2)(1-|v|^2)}$, where $\|.\|$ denotes the usual Euclidean norm. Then the distance function is:

$d(u,v) = arcosh(1+\delta(u,v))$. Such a distance function is defined for any two vectors of norm less than one, and makes the set of such vectors into a metric space which is a model of hyperbolic space of *constant curvature* -1. The model has the conformal property that the angle between two intersecting curves in hyperbolic space is the same as the angle in the model.

5.2. Poincaré 's disk tiling

In this sub-section, we very sketchily remember the minimal data about the Poincaré disk model tiling which constitutes the general frame of our construction.

Let us fix an open disk U of the Euclidean plane. Its points constitute the points of the hyperbolic plane $\mathbb{H}^2$. The border of U, ∂U, is called the set of points at infinity. Lines are the trace in U of its diameters or the trace in U of circles which are orthogonal to ∂U. The model has a very remarkable property, which it shares with the half-plane model: hyperbolic angle s between the lines are the Euclidean angles between the corresponding circles. The model is easily generalized to higher dimension, see [24] for definitions and properties of such generalizations as well as references for further reading.

Tessellations are a particular case of tiling. They are generated from a regular polygon by reflection in its sides and, recursively, of the images on their sides. In the Euclidean case, there are, up to isomorphism and up to similarities, three tessellations, respectively, based on the square, the equilateral triangle and on the regular hexagon.

In the hyperbolic plane, there are infinitely many tessellations. They are based on the regular polygons with p sides and with $\frac{2\pi}{q}$ as vertex angle and they are denoted by $\{p, q\}$. This is a consequence of a famous theorem by Poincaré which characterizes the triangles starting from which a tiling can be generated by the recursive reflection process which we already mentioned. Any triangle tiles the hyperbolic plane if its vertex angles are of the form $\frac{\pi}{p}$, $\frac{\pi}{q}$ and $\frac{\pi}{r}$ with the condition that $\frac{1}{p} + \frac{1}{q} + \frac{1}{r} < 1$.

Here, we just suggest the use of this method which allows to exhibit a tree, spanning the tiling: the Hyperbolic-tree (Figure 4).

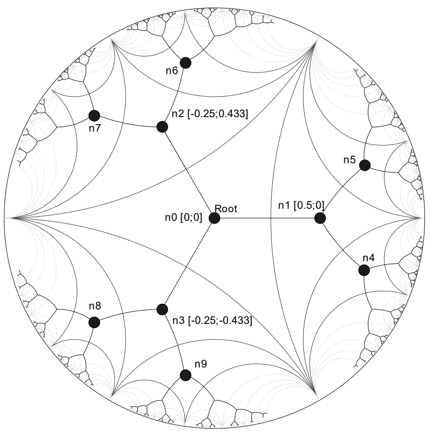

Figure 4. 3-regular tree in the hyperbolic plane.

6. Web services routing principle in the SSDT

We now explain in this section how we built the hyperbolic addressing tree associated to our intermediaries thirds and how different nodes can communicate via queries routing in the distributed system. We propose in our work, a dynamic, reliable and scalable hyperbolic greedy routing algorithm. At the beginning of the building of a distributed directory structure system is the starting of the first node (directory server) containing the service descriptions files that must be shared and the choice of the degree of the addressing tree. We recall that the hyperbolic coordinates (i.e., a complex number) of the addressing tree node are used to locate web services server where is stored a service or a service description on Poincaré disk via an hyperbolic tree. A node of the tree can give the addresses corresponding to its children in the tree. The degree determines how many addresses each node will be able to give. The discovery and storage structure based on the various directories servers interconnected in hyperbolic tree, is then built incrementally, with each new node (directory server) joining one or more existing directories servers. This method is scalable because unlike [23], we do not have to make a two-pass algorithm over the whole directories system to find its highest degree. Also in our discovery structure, a node can connect to any other node at any time in order to obtain an address. The first step is thus to define the degree of the tree because it allows building the dual, namely the regular q-gon. We nail the root of the tree at the origin of the primal and we begin the tiling at the origin of the disk in function of q. Each splitting of the space in order to create disjoint subspaces is ensured once the half spaces are tangent; hence the primal is an infinite $q - regular$ tree. We use the theoretical infinite $q - regular$ tree to construct the greedy embedding of our $q - regular$ tree. Thus, the regular degree of the tree is the number of sides of the polygon used to build the dual (see Figure 4). In other words, the space is allocated for q child nodes. Each node repeats the computation for its own half space. In half space, the space is again allocated for $q1$ children. Each child can distribute its addresses in its half space. The algorithm 1 shows how to calculate the addresses that can be given to the children of a node. The first node takes the hyperbolic address $(0; 0)$ and is the root of the tree.

This distributed algorithm ensures that the nodes representing the services servers are contained in distinct spaces and have unique coordinates. All the steps of the presented algorithm are suitable for distributed and asynchronous computation. This algorithm allows the assignment of addresses as coordinates in dynamic topologies. As the global knowledge of the distributed directories structure system is not necessary, a new node can obtain coordinates simply by asking an existing node to be its parent and to give it an address for itself. If the asked node has already given all its addresses, the new node must ask an address to another existing node. When a new node obtains an address, it computes

Algorithm 1 Coordinates's computing of the Node's children .

```
 1: procedure CalcChildrenCoor($nod, q$)
 2:     $/* nod corresponds to services server */$
 3:     $/* q corresponds to the tree degree */$
 4:     $step \leftarrow argcosh(1/sin(\pi/q))$
 5:     $angle \leftarrow 2\pi/q$
 6:     $childCoor \leftarrow nod.Coor$
 7:     for $i \leftarrow 1, q$ do
 8:         $ChildCoor.rotationLeft(angle)$
 9:         $ChildCoor.translation(step)$
10:         $ChildCoor.rotationRight(\pi)$
11:         if $ChildCoor \neq nod.ParenCoor$ then
12:             StoreChildCoor($ChildCoor$)
13:         end if
14:     end for
15: end procedure
```

the addresses (i.e., hyperbolic coordinates) of its future children. When a new node is connected to the system, it share these informations with others nodes of the distributed directories structure system, by sending queries. The routing process is done step by step from source to target by using the greedy algorithm 2 based on the hyperbolic distances between the nodes.

Algorithm 2 Routing a query in the distributed directories.

```
 1: function getNextHop($nod, packet$) return nod
 2:     $/* nod corresponds to services server */$
 3:     $w = packet.destinationnodeCoor$
 4:     $m = nod.Coor$
 5:     $d_{min} = argcosh\left(1 + 2\frac{|m-w|^2}{(1-|m|^2)(1-|w|^2)}\right)$
 6:     $p_{min} = nod$
 7:     for all $neighbor \in nod.Neighbors$ do
 8:         $n = neighbor.Coor$
 9:         $d = argcosh\left(1 + 2\frac{|n-w|^2}{(1-|n|^2)(1-|w|^2)}\right)$
10:         if $d < d_{min}$ then
11:             $d_{min} = d$
12:             $p_{min} = neighbor$
13:         end if
14:     end for
15:     return $p_{min}$
16: end function
```

In a real distributed directories system environment, link and node failures are expected to happen often. If the addressing tree is broken by the failure of a node or link, we flush the addresses attributed to the nodes beyond the failed node or link and reassign new addresses to those nodes (some nodes may have first to reconnect with other nodes in order to restore connectivity).

7. Naming and binding principle in the SSDT

In this section we explain how our distributed directories structure system stores and retrieves the $(key, value)$ pairs which will be store in SSDT is such as the key corresponds to the hashing (using the SHA-512 [26] algorithm) of the identifier associated with to the service description and the value to the concerned service. Figure 5 indicates the sequence model realized between web service storage and web service discover on our SDDT system by using SAPDS protocol.

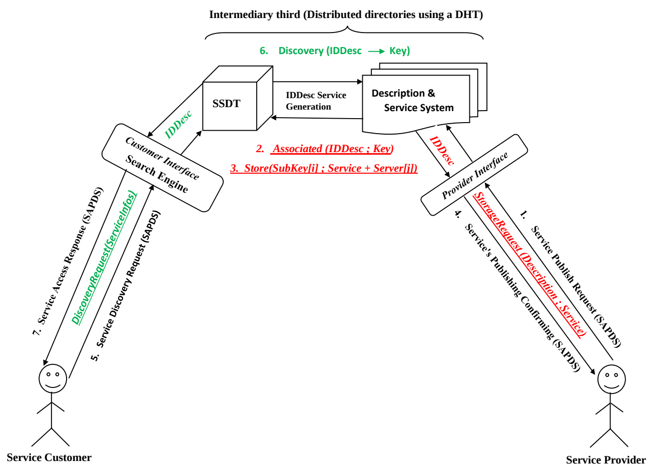
Intermediary third (Distributed directories using a DHT)
6. Discovery (IDDesc ⟶ Key)
SSDT
IDDesc Service Generation
Description & Service System
IDDesc
Customer Interface Search Engine
2. Associated (IDDesc ; Key)
3. Store(SubKey[i] ; Service + Server[j])
IDDesc
Provider Interface
7. Service Access Response (SAPDS)
DiscoveryRequest(ServiceInfos)
5. Service Discovery Request (SAPDS)
4. Service's Publishing Confirming (SAPDS)
StorageRequest (Description ; Service)
1. Service Publish Request (SAPDS)
Service Customer
Service Provider

Figure 5. Use case for Service Storage and Discovery.

7.1. Web service storage process

Our solution is a structured DHT system that uses the local addressing and the greedy routing algorithms presented in Section 6. In Figure 6, we describe the sequence model of the storage process.

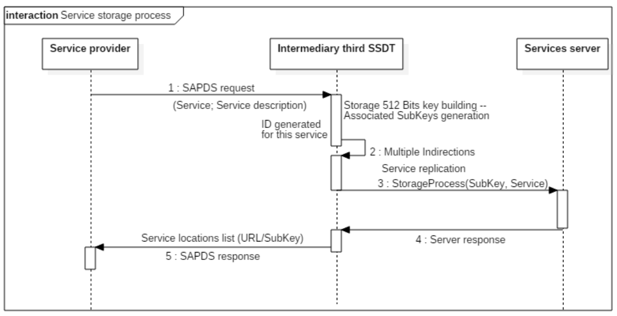
interaction Service storage process
Service provider
Intermediary third SSDT
Services server
1 : SAPDS request
(Service; Service description)
Storage 512 Bits key building -- Associated SubKeys generation
ID generated for this service
2 : Multiple Indirections
Service replication
3 : StorageProcess(SubKey, Service)
Service locations list (URL/SubKey)
4 : Server response
5 : SAPDS response

Figure 6. Service Secured Storage.

At the beginning, when a new service has to be to store in our system SSDT, it uses the web services supplier interface to send as SAPDS request, the service description as well as the service. From the interface, the description and service system associates the description (in particular the service location URL) with a service identifier called ”IDDesc“. Afterwards, this IDDesc is transformed into 512-bits key, by SHA-512 algorithm. The 512-bits key is then subdivided into 16 subkeys of 32-bits. Every subkey will be use to compute the service location in the system SSDT (particularly on hyperbolic tree) following a linear transformation given in Equation 2.

The description and service system selects the each subkey and SSDT system maps it to an angle by a linear transformation. The angle is given by:

$$\alpha = 2\pi \times \frac{\texttt{32-bit subkey}}{\texttt{0xFFFFFFFF}} \tag{1}$$

The node then computes a virtual point v on the unit circle by using this angle:

$$v(x, y) \text{ with} \begin{cases} x = cos(\alpha) \\ y = sin(\alpha) \end{cases} \tag{2}$$

We call here *binder*, the node of hyperbolic tree and *binding* (subkey, service) pair. The service will then be stored in binder which address (coordinates) is the closest to the point situated on the unit circle computed from the determined angle previously. Figure 7 illustrate *binding* model.

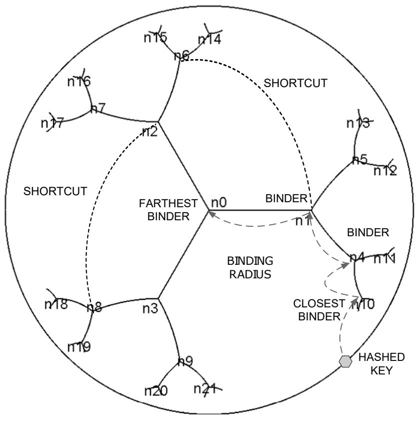

Figure 7. Hyperbolic DHT system.

At the beginning, we can define two replication or redundancy mechanisms for storing copies of a given binding:

1. We can use more than one binding radius by creating several uniformly distributed subkeys.

2. We can store the pair in more than one binder in the same binding radius.

Algorithm 3 Storage algorithm in replication context

```
 1: procedure StoreReplic($Node\ node, Degree\ q$)
 2:     $SrcURL \leftarrow Source.GetURL()$
 3:     $Key \leftarrow Hash(SrcURL)$
 4:     for ($r \leftarrow 0, R_{Circular}$) do
 5:         $depth \leftarrow Depth_{Max}$
 6:         $i \leftarrow 1$
 7:         while ($i \leq \left\lfloor \frac{1}{2} \times \frac{log(N)}{log(Q)} \right\rfloor$ && $d \geq 0$) do
 8:             $SrcSubK[r][d] \leftarrow CSubK(Key)[r][d]$
 9:             $TgtAd[r][d] \leftarrow CAd(SrcSubKey[r])[d]$
10:             $Tgt \leftarrow GetTgt(TgtAd[r][d])$
11:             if ($route(Source, Tgt)$) then
12:                 $i++$
13:                 $put(key, TgtAd[r][d])$
14:             end if
15:             $d--$
16:         end while
17:     end for
18: end procedure
```

The depth of a node in the addressing tree is defined as the number of parent nodes to go through for reaching the root of the tree (including the root itself). For each *binder*, the process is repeat from child *binder* to parent *binder* until to reach root or number of replications specify is reached. Algorithm 3 illustrate storage process. This mechanism gives as number of radial replication $R_{Radial} = Depth_{Max}$ or $R_{Radial} = \frac{log(N)}{log(Q)}$ with N equal to the number of server of SSDT system and Q equal to the degree of hyperbolic tree.

These mechanisms enable our DHT system to cope with an non-uniform growth of the hyperbolic tree and they ensure that a pair will be stored in a redundant way that will maximize the success rate of its retrieval. The numbers of subkeys and the numbers of copies in a radius are parameters that can be set at the creation of the intermediary third. Increasing them leads to a tradeoff between improved reliability and lost storage space in binders.

The division in $R_{Circular} = 16$ subkeys of identical sizes will later allow to set up a strategy of services storages replication on various servers. The aim being to increase the performance in term of mean number of hops to be realized to reach a service and therefore to minimize the mean responses time of the SAPDS requests. When the successful storage, an SAPDS response is sent to the service provider in the aim to inform him.

Our solution has the property of consistent hashing: if one node fails, only its keys are lost, but the other binders are not impacted and the whole system remains coherent. As in many existing systems, pairs will be stored by following an hybrid soft and the hard state strategy. A pair will have to be stored by its creator every x period of time otherwise it will be flushed by the binders that store it. A delete message may be sent by the creator to remove the pair before the end of the period. This mechanism is going to facilitate the substitution strategy of the failed nodes which we propose and simulate to improve the reliability of our distributed system in dynamic context (Section 8).

7.2. Web services discovery process

Now, we are going to present the mechanism of service discovery in our intermediary third based on a DHT structure. In this context, Figure 8 presents the sequence model that we use to realize web service lookup.

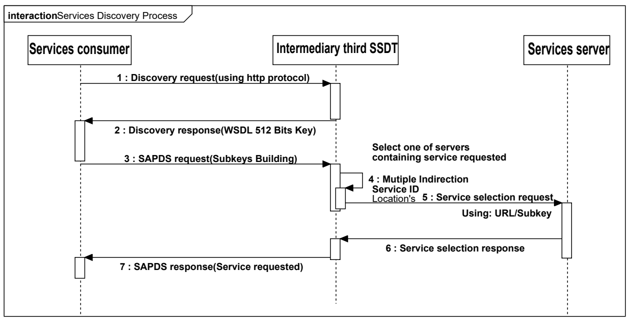
interactionServices Discovery Process
Services consumer
Intermediary third SSDT
Services server
1 : Discovery request(using http protocol)
2 : Discovery response(WSDL 512 Bits Key)
Select one of servers containing service requested
3 : SAPDS request(Subkeys Building)
4 : Mutiple Indirection Service ID Location's
5 : Service selection request Using: URL/Subkey
6 : Service selection response
7 : SAPDS response(Service requested)

Figure 8. Service Secured Discovery.

When a user wants to discover a service, were supplied the related service information in a dedicated search engine. This system will make the correspondence between the information of service and the service identifiers stored in our system SDDT. In near this stage, comes the step of service identifier choice (IDDesc) which then transformed into a key of $512-bits$ by the SHA-512 algorithm . This key is then subdivided into subkeys then begin the step of server search susceptible to contain the service for every subkey. We stop as soon as one subkey is satisfactory, that is allows to find the sought service stored in a server. In the worst of the cases, the process continues with all the others under key. Indeed, because of the dynamics of the system, certain server can become inaccessible thus we let us have in our mechanism of recovering hyperbolic tree, shortcuts allowing to create alternative links (Figure 7).

Algorithm 4 shows the web service discovery process.

It to note that our system allows to change service version. To do it, we proceed to a mechanism of discovery from the service identifier (IDDesc) given to the service provider at the end of his process of service storage. In the case or this new version comes to replace the previous, we proceed to a new storage from the first step as if it was again a question service. Otherwise, we want to update the service, by replacing all case of the previous version of service by a news version and by keeping the previous service identifier.

Algorithm 4 Lookup Algorithme in Replication Context

```
1:  procedure LookupReplic($Node\ Source, Node\ Tgt$)
2:      $Tgtname \leftarrow Tgt.GetName()$
3:      $Key \leftarrow Hash(Tgtname)$
4:      for ($r \leftarrow 0, R_{Circular}$) do
5:          $d \leftarrow Depth_{Max}$
6:          $i \leftarrow 1$
7:          while ($i \leq \left\lfloor \frac{1}{2} \times \frac{log(N)}{log(Q)} \right\rfloor$ && $d \geq 0$) do
8:              $TgtAddress[r][d] \leftarrow GetValue(Key)$
9:              $Tgt \leftarrow GetNode(TgtAddress[r][d])$
10:             if ($Tgt\ != null$) then
11:                 $i++$
12:             end if
13:             $d--$
14:         end while
15:     end for
16: end procedure
```

7.3. Substitution strategy

In this section we are going to present a strategy of replacement of nodes failed in the system. In the Section 7 we saw that periodically nodes have to send of storage messages to their binders at the risk of being to delete of the table of storage of each of their binders.

Algorithm 5 Substitution Algorithm of Failed Node

```
1:  procedure Substitution($Node\ Failed, Node\ Leaf,$
    $Node\ New$)
2:      $Neighbor[d] \leftarrow Leaf.getNeig()$
3:      $Failed \leftarrow Node.SetFails()$
4:      $Binder[i] \leftarrow CompBinders(Failed)$
5:      for ($j \leftarrow 0, Neighbor[i].length$) do
6:          $Neighbor[i].SetEntry(Failed)$
7:      end for
8:      $Binder[i].Clean()$
9:      for ($k \leftarrow prof, 0$) do
10:         if (($TargNode \leftarrow PNode.getEntry()$) $!=$
    $null$) then $TargNode.SetP(New)$ $break$
11:         end if
12:         $PNode.delete(TargNode)$
13:         $d--$
14:     end for
15: end procedure
```

In the strategy of substitution, during a failure of node in the system, instead of deleting the entries of the node of the storage table of each of these binders, we move these entries to a table buffer of each of neighbour nodes of the node having failed. This table keeps deleted entries only during a limited time called ”time of hope“. So, when a node wants to connect, it questions randomly a node of the system which passes on father’s request in father possibly until find a similar node which is ready to accept the new node because in the table buffer, there is an entry of a node having failed the system either until reaching the root. If it reaches the root unsuccessfully, the first son node if it arranges to always have free address, allows the new node to connect to become a node leaf of the system. Otherwise, the request is redirected towards another node son. This approach is executed by Algorithm 5.

8. Simulations results

8.1. Simulations context

In this section, we make an experimental study of our solution of directories distributed of 1st and of 2nd level. For that purpose, let us consider a system of 10.000 directories servers interconnected in a dynamic context (i.e. where there is departure and arrived from new directories servers) following an exponential law of median 10 min. We suppose that service providers store WSDL files as well as services every 100 ms. Besides, every nodes or directories server can store only the most 10.000 WSDL files in the case of the 1st intermediary third and 10.000 services in the case of the 2nd intermediary third. We also consider that the structure of hyperbolic tree is of degree 3 and depth 8.

8.2. Load balancing and scalability of our distributed directories structure

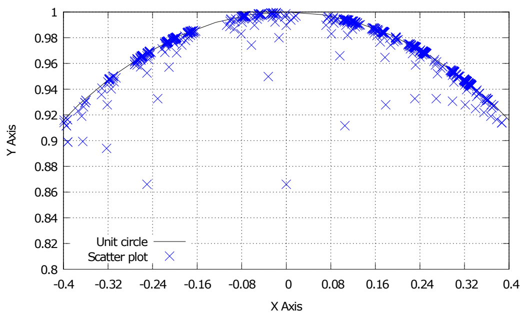

Figure 9. Scatter plot of distributed directories system.

In this subsection, we try to show how our hyperbolic tree structure stemming from the tessellation of Poincaré disk support is scalable. Indeed, the Figure 9 shows that in spite of 10.000 servers of directories whom we use, all the nodes representing the latter remain inside the unit circle according to the Poincaré disk model which has an edge in the infinity. This figure presents the distribution of directories servers to the neighborhood on the edge of the circle.

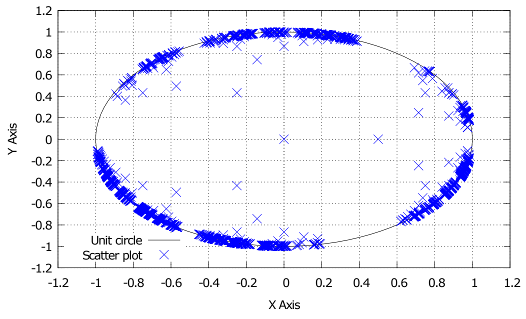

Figure 10. Scatter plot of distributed directories system.

Besides, we also show that our system allows realize the load balancing. Indeed, we can remark that 10.000 nodes are almost uniformly distributed on the Poincar é disk as Figure 10 shows it.

8.3. Analysis of the primary number of binders depending on the number of sub-keys

In our works, the notion of primary binder made a reference to the node the closest to the border of the circle. In our case, the ideal is 16 primary binders in reference to the number of key obtained by fragmentation of the key of 512 bits stemming from the hashing of the service description URL or from the service URL.

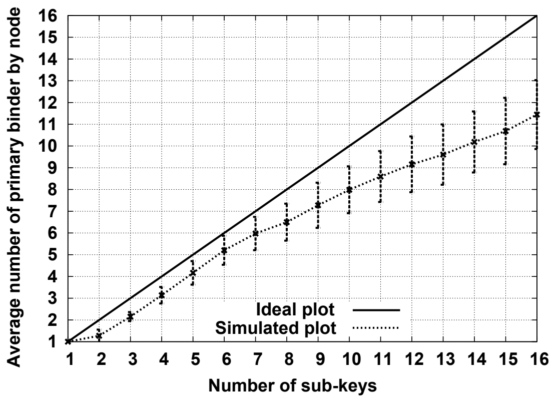

Figure 11. Variation du nombre de stockeur en fonction du nombre de sous-clefs

Figure 11 presents the evolution of the number of primary binders depending on the number of sub-keys chooses for the simulation. This plot shows a continuous growth of the number of binders depending on the number of sub-keys. Furthermore, we can observe that this plot is below the ideal case or the number of binders always corresponds to the number of sub-key. Simulated plot is besides very close to the ideal plot, what shows that we have a satisfactory situation for all the levels of replication. Indeed, this simulated plot shows that the trend is for a primary differentiation of binders, that increases at the same time the resistance of the system in front of breakdown. For example, when we have 6 used sub-keys, it corresponds to the identification of 5 different binders to store the pair key-value of any node. When we have 15 different sub-keys, we have for every node 11 different binders in average which store its pair key-value. Where from the interest for the system, then even when binder leaves the system, the information remains persistent. This study contributes to watch that the system which we propose remains reliable rest in many dynamic contexts.

8.4. Substitution strategy performance evaluation

Figure 12 shows us the impact of the substitution strategy on the average number of binders reached by by every node during the storage process. Indeed, we compare a situation or the new nodes join in a random way the system via the first node having free address and the case or the new nodes try to substitute themselves for binders having left the network has in the aim to continue to assume this role.

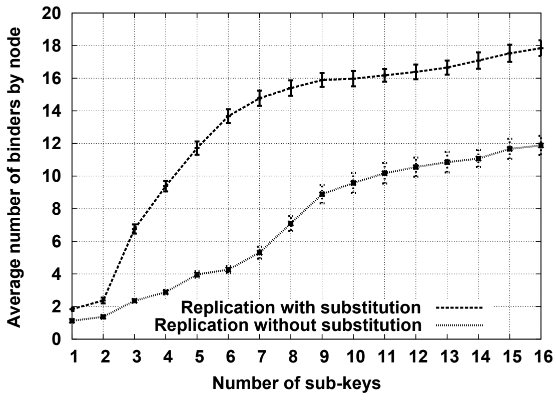

Figure 12. Performances evaluation of the substitution strategy

In Figure 12, we present the variation of average numbers of binders depending of number of sub-keys, thus, we can observe that the average numbers of binders reached in the case of the substitution is much more important than in the case of a connection of new nodes to the first random address. This established fact shows that the substitution strategy returns our more the hardness of the system in the sense that the pairs keys-values of the various nodes are distributed on a largest number of binders so returning the available information even in case of departure of binders. So, we have for example by using 8 sub-keys, approximately 7 binders on average during all the simulation in the strategy of circular replication (depending on number of subkeys) and simple radial road (depending on tree depth (d)) against about 14 binders when we use the substitution method in more. Furthermore, for 16 sub-keys, we have approximately 12 binders against about 18 on average in the case of the substitution.

8.5. Storage and discovery success rates evaluation

Figure 13 presents two curves illustrating respectively the evolution of the rate of success of the requests of storage in the case of a classic replication (circular and radial road) then in the case of a classic replication with which we associated the substitution strategy.

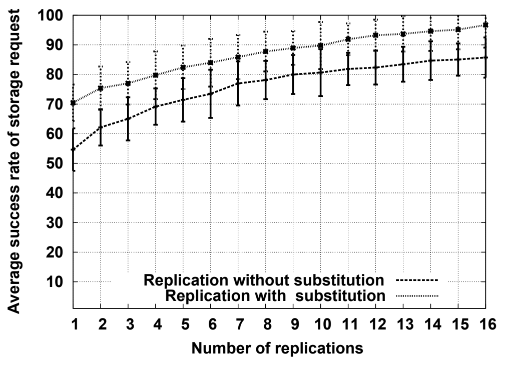

Figure 13. Success rate evaluation in the storage

In Figure 13, we present average success rate of storage request depending of the number of replication. Thus compared to the classic replication results, the success rate offered by the substitution strategy is better. Indeed, in a replication, we note a rate of average success about 62% of successes for the classic replication against about 75% of the method of substitution. In the case of 7 replications, we note 78% of successes in the case without substitution against 89% in the case using this strategy. With 15 replications, we observe 85% of rates of success in without substitution against 97% of successes in the case of the substitution. Generally, we can notice an average an earnings of about 10% in the success of storage requests.

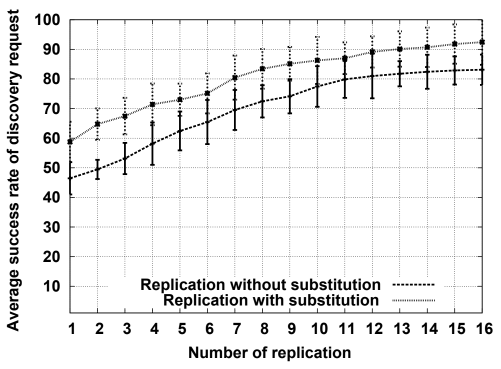

Figure 14. Success rate evaluation in the lookup (discovery)

In the case of discovery requests, the situation is almost similar because to observe us in Figure 14 a better success rate when we use the substitution strategy. Indeed, also in it an earnings of an average 9% of successes on requests for a discovery. Whether it is for the storage or for the discovery, we can say that our strategy is reliable because, she allows us to increase considerably the success rates.

9. Conclusion and perspectives

In this paper we made an analysis of the inadequacies of the classic web service model. Thus, we show that the triplet WSDL, SOAP and UDDI does not allow to assure on one hand an availability of the services with regard to the dynamic context of Internet and other one by it does not guarantee the secured access to the service because it creates a direct link of connection between the service consumer and the service provider. To fill these failures, we propose new architecture of web services management which we name SSDT which allows to assure the scalability in the services storage thanks to DHT structure, an availability of its services thanks to the replication principle and a secured access thanks to a decoupling between the service provider and the services consumer via an intermediary third based on a spanning hyperbolic tree structure.

The continuation of its works is going to consist on one hand to propose other strategies to improve the performances of our system then implement the protocol SAPDS then our middleware SSDT in a realistic context.

References

[1] Z. Yun and S. Huayou and Q. Hengnian and N. Yulin (2010). An approach to discover semantic web services in distributed environment based on Chord. International Conference on Intelligent Systems and Knowledge Engineering, pp. 401-405.

[2] G. Meditskos and N. Bassiliades (2010). Structural and role-oriented web service discovery with taxonomies in OWL-S. IEEE Trans. Knowl. Data Eng., Volume 22(2):278-290.

[3] W. Ren and Z. Xu (2008). A new web service discovery method based on semantic. Workshop on Power Electronics and Intelligent Transportation System, ,pp. 223-226.

[4] S. Hwang, E. Lim, C. Lee, and C. Chen. On composing a reliable composite web service: A study of dynamic web service selection. In 2007 IEEE International Conference on Web Services (ICWS 2007), July 9-13, 2007, Salt Lake City, Utah, USA, pp. 184-191, 2007.

[5] J. Ma, Y. Zhang, and J. He (2008). Efficiently finding web services using a clustering semantic approach. In Proceedings of the International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation: Organized with the 17th International World Wide Web Conference (WWW 2008), CSSSIA '08, New York, NY, USA, ACM, pp. pp. 1-5.

[6] E. Al-Masri and Q. H. Mahmoud (2009). Discovering the best web service: A neural network-based solution. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, pp. 4250-4255.

[7] E. Kabir and H. Wang and E. Bertino(2011). A conditional purpose-based access control model with dynamic roles. In Expert Systems with Applications, Volume 38(3): 1482-1489.

[8] H. Wang and J. Cao and Y. Zhang(2011). A Flexible Payment Scheme and Its Role-Based Access Control. In IEEE Transactions on knowledge and Data Engineering, Volume 17(3): 425-436.

[9] M. P. Papazoglou and W.-J. Heuvel (2007). Service oriented architectures: Approaches, technologies and research issues. The VLDB Journal, Volume 16(3):389-415.

[10] B. T. et. al. Uddi specification index page. (Date last accessed 19-July-2002).

[11] A. ShaikhAli, O. F. Rana, R. Al-Ali, and D. W. Walker (2003). UDDIe: An Extended Registry for Web Services. In Proceedings of the 2003 Symposium on Applications and the Internet Workshops (SAINT'03 Workshops), SAINT-W '03, Washington, DC, USA, IEEE Computer Society, pp. 85-89.

[12] D. Bela´id, N. Provenzano, and C. Taconet (1998). Dynamic management of corba trader federation. In Proceedings of the 4th Conference on USENIX Conference on Object-Oriented Technologies and Systems, COOTS'98, USENIX Association, , Berkeley, CA, USA, Volume 4, pp. 4-4.

[13] R. C. Prim (1957). Shortest connection networks and some generalizations. The Bell Systems Technical Journal, Volume 36(6), pp. 1389-1401.

[14] S. Melnik, H. Garcia-Molina, and E. Rahm (2002). Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In Proceedings of the 18th International Conference on Data Engineering, ICDE '02, IEEE Computer Society, Washington, DC, USA,

pp. 117-128.

[15] S. Melnik, E. Rahm, and P. A. Bernstein (2003). Rondo: A programming platform for generic model management. In Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, SIGMOD ’03, , New York, NY, USA, ACM, pp. 193-204.

[16] A. Norbert and P. Athanase (2012). Notes on hyperbolic geometry. Strasbourg Master class on Geometry, Volume 18, pp. 1-182.

[17] A. Papadopoulos (2010). Lobachevsky : Pangeometry. Heritage of European Mathematics. European Mathematical Society.

[18] J. Milnor (1982). Hyperbolic geometry: the first 150 years. Bulletin of the American Mathematical Society, Volum 6 (1), pp. 9-24.

[19] S.-G. Taherian(2010). On algebraic structures related to Beltrami-Klein model of hyperbolic geometry. Bull. Amer. Math. Soc., New Ser, Volume 57, pp. 205-219.

[20] J. Stillwell(1998). An elementary introduction to the poincaré half-plane model of the hyperbolic plane. Springer-Verlag, pp. 100-104.

[21] Wikiwand (1957). Poincaré disk model. (Date last accessed 19-January-2016). Volume 36(6), pp. 1389-1401.

[22] Y. Bi, B. Fan, and F. Wu (2015). Beyond mahalanobis metric: Cayley-Klein metric learning, pp. 2339-2347.

[23] R. Kleinberg (2007). Geographic routing using hyperbolic space. In in Proceedings of the 26th Annual Joint Conference of INFOCOM,Computer and Communications Societies, IEEE, pp. 1902-1909.

[24] M. Margenstern (2007). Cellular automata in hyperbolic spaces. Encyclopedia of Complexity and Systems Science, Paris, pp. 791–800.

[25] M. Margenstern (2000). New tools for cellular automata in the hyperbolic plane. J. UCS, Volume 6(12), pp. 1226-1252.

[26] R. Kayalvizhi, R. Subramanian, R. Santhosh, J. Gurubaran, and V. Vaidehi(2010). CNSA, Communications in Computer and Information Science, Springer, Volume 89, pp. 105-113.