DOI : 10.3745/JIPS.2008.4.2.053

Journal of Information Processing Systems, Vol.4, No.2, June 2008 **53**

An Autonomic <K, D>-Interleaving Registry Overlay Network for Efficient Ubiquities Web Services Discovery Service

Khaled Ragab*

Abstract: The Web Services infrastructure is a distributed computing environment for service-sharing. Mechanisms for Web services Discovery proposed so far have assumed a centralized and peer-to-peer (*P2P*) registry. A discovery service with centralized architecture, such as *UDDI*, restricts the scalability of this environment, induces performance bottleneck and may result in single points of failure. A discovery service with P2P architecture enables a scalable and an efficient ubiquities web service discovery service that needs to be run in self-organized fashions. In this paper, we propose an autonomic <*K*, *D*>-interleaving *Registry Overlay Network* (*RgON*) that enables web-services' providers/consumers to publish/discover services' advertisements, *WSDL* documents. The *RgON*, *doubtless* empowers consumers to discover web services associated with these advertisements within constant *D* logical hops over constant *K* physical hops with reasonable storage and bandwidth utilization as shown through simulation.

Keywords: *Ubiquities Web Service Discovery Service, Registry Overlay Network P2P.*

1. Introduction

Discovery of ubiquities web services is becoming a hot topic as Web service (*WS*) has drawn increasing attention nowadays. *Web Services* (*WSs*) are self-contained, loosely coupled application modules with well described functionality that can be published, located and invoked across the web. The growing number of *WS* lunched in the web raises new challenges, such as discovery of *WS*. Much of the research and work on *WS* discovery are based on centralized registries as UDDI [16]. Each registry enables every *WS* coming on line to advertise its existence and its capabilities, and functionalities with this registry. Moreover, every service requester accesses the registry to discover the most appropriate *WS*. In fact, the centralized registries guarantee discovery of services that have registered. On the other hand, they suffer from performance bottleneck and single point of failure. In addition, they may be having experiences to denial of service attack. Moreover, the storage of huge number of advertisements on centralized registries hampers the timely update. These conundrums can be partially mitigated through replication of servers against single point of failure and performance bottlenecks. P2P techniques provide an alternative that does not rely on centralized service; rather it allows *WSs* to discover each other dynamically. A discovery services with P2P architecture facilitates a scalable and an efficient *WS* discovery service. Most of P2P *WS* architectures so far have assumed one overlay network contains *WS*' nodes and end-users' nodes such as [2], [3], [6]. Thus, multiple of data packets and service discovery packets undergo severe network overloads on this overlay network. Thus, self-organizing this overlay network into two types of overlay networks provides an implicit scoping mechanism for restricting the propagation of discovery and search requests. This paper briefly introduces the proposed *Autonomic Community Computing Infrastructure* (*ACCI*) that enables the service providers (e.g. small and medium retailers) to outsource their *WSs* to the end-users' nodes with the required QoS. Moreover, it allows end-users to discover *WSs* in an autonomic P2P fashion. The *ACCI* organizes nodes (entities) of the end-users into an *Autonomic Community Overlay Network* (*ACON*) [1]. It is a self-organized logical topology as shown in figure 1. Entities of the *ACCI* are set in contact without the intervention of any third partner. Every resource in the *ACON* (node, node's resources, group, pipe, *WS*, etc.) is self-described and published using advertisements; *XML* and *WSDL* documents. Advertisments (*Advs*) describe WS' life-time, functionalities, location, QoS and etc. *ACCI* introduces two different types of nodes. First, *Edge Node* (*EN*) offers a computing power and storage area for hosting *WSs* and publishing the associated *Advs*. Second, *Registry Node* (*RN)* offers a storage area for *Advs* that is published by *EN*. *ACON* is divided into two types of overlay networks. First, *Registry Overlay Network* (*RgON*) is a self-organized logical topology of registry nodes that enables publishing *WS*' *Advs* and discovering *WSs* associated with these *Advs*.

Manuscript received April 8, 2008; revised April 23, 2008; accepted May 27, 2008.

Corresponding Author: Khaled Ragab

* College of Computer and Information Technology, King Faisal University, Al Ahsa, P.O. Box 400, Saudi Arabia (kabdultawab@kfu.edu.sa, k_ragab2005@yahoo.com)

Web Service Overlay Network (W*sON*) is made up of several *EN* that provide same *WS*. In this paper, we focus on exploring a <K, D>-interleaving *RgON* construction scheme. For restricting the propagation of *WS* discovery quires, *RgON* provides an implicit scoping mechanism.

This paper is organized as follows. Section 2 briefly clarifies the proposed *ACCI* concept, exhibits the system architecture and illustrates the self-organized *ACON*. Section 3 expose the autonomic <K, D>-interleaving RgON step by step construction scheme. Section 4 presents simulation results based on realistic Internet settings showing improvement in dense underlying networks. Section 5 draws conclusions.

2. Autonomic Community Computing Infrastructure: Concept and Architecture

2.1 Concept

The Web Services infrastructures [10] have reached a level of complexity, heterogeneity, and dynamism for which current programming environments and infrastructure are becoming unmanageable and brittle. The management systems cost reaches 80% of the IT [19]. Induced from the strategies used by the biological systems to deal with complexity, heterogeneity and uncertainty an approach called *Autonomic Computing* [4], [7] is loomed ahead. An autonomic computing system is one that has the capabilities of being *self-defining*, *self-healing*, *self-configuring*, *self-optimizing*, *self-protecting*, contextually aware, and open [8]. Inspired from the *Autonomic Computing* and the cooperation in the social communities, this paper defines *Autonomic Community Computing Infrastructure* (*ACCI*), self-defining infrastructure able to simultaneously federate and autonomously manage and discover multiple of *WS* with assuring the required quality and reliability under the evolving situations. Moreover, it autonomously manages (e.g. monitoring, planning, analyzing, executing) the pool of the nodes' resources and *WSs* in self-configuring and self-optimizing fashions].

2.2 Architecture

ACCI organizes the EN and RN into a self-organized logical topology as shown in figure 1 and called *Autonomic Community Overlay Network* (*ACON*) [1]. For example, figure 1 shows that each node knows few neighbors. Lines are the logical links (e.g. *Pipe* in *JXTA* [11]) among the nodes. Pipes are virtual communication channels used to send and receive messages. We employ *JXTA* [11] that provides a set of protocols for forming a virtual overlay network on top of current existing Internet and non-IP based networks. It eases of dynamically creating and transforming overlay network topology that enables the deployment of *ACON*. Every resource in the *ACON* (node, node's resources, group, pipe, service, etc.) is described and published using *Advs*. The communication among *ACON*'s nodes based upon *JXTA* protocols [11] and *SOAP* (*Simple Object Access Protocol*) [12], [13]. *JXTA* protocols enable nodes to discover node services and handle message propagation among nodes through input/output pipes. SOAP provides the means communication for *WS* and customer application. To wrap the SOAP messages into *JXTA* pipe messages and transport them through *JXTA* pipes *JXTA-SOAP* [14] is used. The *ACCI* uses the *WSDL* (Web Services Definition Language) [12] to expose the service functionalities and interfaces. *UDDI* [16] is used for locating *WS*. Unlike in the *WS*, in *ACCI* there is no centralized discovery mechanism to locate services. Instead an autonomic peer-peer *RgON* is used for locating services. Moreover, the current *UDDI* model limits the service discovery to functional requirements only. It is foreseeable that there may be more than one *WS* available that can meet the functional requirements with different quality of service attributes. Alike [20], *ACCI* provides the ability of incorporating QoS into service discovery process but in decentralized peer-peer approach.

2.3 Registry Overlay Network

RgON is a set of *RN* nodes. It offers a storage area for resources' *Advs* that have been published by *EN*. Every resource in the *ACON* (node, node's resources, group, pipe, *WS*, etc.) is described and published using *Advs*. *Advs* are structured *XML* documents except *Advs* of *WS* are structured using *WSDL*. Each *RN* autonomously adds/deletes *WS*' *Advs* according the *WS* life-time. To publish these *Advs* they should be replicated into *RgON*. Registry nodes have the extra ability of forwarding the request they receive to other registry nodes in *RgON*. *JXTA* provides to *RgON* an easy interface for publishing and discovering *WS* in a peer to peer manner. WSs can be described semantically including QoS properties, for example using *WSMO*†. In addition, the *JXTA* discovery query and publish message should be updated to allow *RgON* to guarantee the QoS. For example, the *JXTA* discovery query message *XML* should include an element that describes the QoS such as *<qualityInfo> <availability> 0.9 </availability> </qualityInfo>*. The following section describes the *<K, D>*-interleaving scheme that forms *RgON* into *RgON-Clusters* for providing an implicit scoping mechanism for restricting the

† Web Service Modeling Ontology http://www.wmso.org/

propagation of discovery and search requests. Thus, the service discovery delay is improved as manifested in section 4 through the simulation results.

2.4 Web Service Overlay Network

WsON is a set of *EN* nodes. It offers computing power, storage, etc for hosting WSs and publishing the associated *Advs*. For example, figure 1 shows an example of *ACON* that consists of two *WsON S1* and *S2*. The main motivation of creating *WsON* is to tolerate the node failure/leave within the group. *WsON* serve to subdivide the *ACON* into regions, providing *WSs* with different qualities. The choice to construct *WsON* providing a *WS* either with different QoS levels or one QoS level is the undergoing research in our research group. In this paper, we consider that all *EN* participate in *WsON* provide same service with same QoS. In addition, the *WsON* autonomously adapts the dynamic network's changes by outstretching (adding more replica nodes) or shrinking (removing some replica nodes) to guarantee the QoS delivered to consumer with minimum resource utilization. *EN* can be a member of one or several *WsONs* such as node *C* in figures 1. Node *C* hosts *WSs* and monitors its resources utilization (e.g. load, bandwidth, etc.) to guarantee their QoS; otherwise it should leave one of these *WsONs*. Indeed, fairness among members in *WsON* is significant to encourage end-users to join it. Constructing *WsON* is out the scope of this paper.

2.5 Web Service Advertisement and Discovery

Any *SP* wishing to provide its service and has no or shortage of resources need to first bootstrap into *ACON*, join the root peer group *NetPeerGroup* of *ACON* and then replicate its service into *ACON*. For example, as shown in figure 1 node *SP1* joins *ACON* and then forms *S1* containing nodes *A, C, D, E, F* and *G*. Then each node sends an advertisement of *S1* to a registry node from *L, M, N, etc*. Similarly, any customer wishing to utilize a service replicated in *ACON* needs to first bootstrap into *ACON*, join it, advertise its resources, discover the services' *Advs* and then follow the service's advertisement to open a *JXTA* output pipe. The output pipe allows the node to send a query to *WS* discovery module. In addition, it opens *JXTA* input pipe to wait for the query result. For example, in figure 1 *EN H* sends a query to its *RN Q* to discover *WS S2*. The *ACCI* utilizes a flooding approach of the *WS* discovery queries within *RgON* to lookup the service's advertisement. When the number of *RN* increases, *RgON* undergoes a high amount of communication overhead. Thus, following section explores an efficient clustering technique for *RgON* We formed and engaged *RgON* to publish and discover *WS*' *Advs* efficiently. This section focuses on the construction of *RgON* with preserving a specific property "*<K, D>-interleaving*" for efficient service discovery as follows.

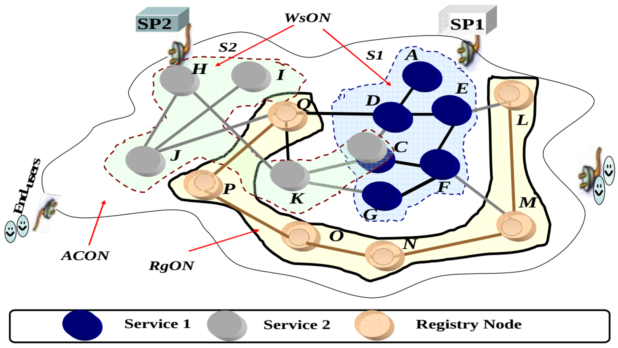

Fig. 1. ACON architecture.

3. Autonomic <K, D>-Interleaving RgON Construction Scheme

3.1 <K, D>-Interleaving RgON

Figure 2 shows the 2-tiers architecture of *RgON*. It is composed of multiple *RgON-Clusters* RC_j ; $j=1, \ldots, \beta$ in level 1. Each RC_j has a *landmark* registry node (Seed RN) called L_j. The physical distance between any $u \in RC_j$ and L_j is $\delta(u, L_j) \leq r;\ K=2r$ as shown in figure 2. The *distance* between two vertices u and v, $h(u,v)$, is the number of logical hops of a shortest path between u and v, and its maximum value over all pair of vertices, $D_t^{(j)} = max\{h(u,v);\ \forall\ u, v \in RC_j\}$, is the *diameter* of the *RgON-Cluster j* at instance of time t where $D_t^{(j)}$ is less than or equal the upper bound diameter D. In level 2 a seed registry cluster containing all landmark registry nodes is constructed and called *Seed-Cluster* as shown in figure 2. The size of the Seed-Cluster is denoted by β and equaled to the number of *RgON-Clusters* in level one. All *Advs* had published in *RgON* must be stored in each *RgON-Cluster*.

Definition *1* (***<K, D>-Interleaving***): Given a graph $G = (V, E)$, where each edge $e \in E$ is associated with a positive number $l(e)$ called its *length* and each vertex $v \in V$ is associated with a non-negative integer $\omega(v)$ it is the number of color slots the node v has. Each node $v \in V$ is colored by $\omega(v)$ different colors. The graph G and pair *<K, D>* are integers that construct a coloring of G so that no connected sub-graph G\` contains two vertices colored same. Each G\` has a diameter less than or equal the upper bound diameter D over K physical hops, or, equivalently, the distance between any two vertices in a label-set is at least *<K, D>*, where *K, D* are the *interleaving parameters*. The history of the work on interleaving schemes is rather brief. Blaum et al. [21] introduced interleaving schemes and analyzed them on two- and three-dimensional arrays. The follow-up paper [22] generalized interleaving schemes to those with repetitions, where in any connected cluster of size $|S|$ any label is repeated at most ρ times. Asymptotically optimal constructions on *2-dimension* arrays were presented for the case ρ *=2*. In this paper we extend interleaving schemes beyond. This paper organizes *RgON* into <*K, D*>-interleaving *RgON-Clusters*. Each *RgON-Cluster* (*RC*) is a self-organized cluster that maintains the <*K, D*>-interleaving property with ρ *=1* as shown in the following sections. There are no two RN nodes belongs to an *RgON-Cluster* stored same advertisement. The intersection between any two *RgON* clusters is empty. Each *RC* stores all *Advs* published in *RgON*. Similar to *Gnutella* [24], each *RC* utilizes flooding of the discovery query locally within the cluster to lookup the service's advertisement. Each advertisement is hashed into a *24*-bit RGB color number. Each RN node belongs to *RCr* is colored by $\omega(v)$ color slots associated with the *Advs* it stored. Thus, the overhead of any discovery request for any advertisement is bounded by *D,* the number of logical hops over *K* physical hops where D, K << $|V|$. *Chord* performs service discovery over a DHT. Similarly, [26] performs a registry discover over a DHT. Lookup delay over *Chord* is *O(log(N))* [24] in contrast it is *O(D)* in *RC*. Most of discovery algorithms cannot provide such a lookup delay scale.

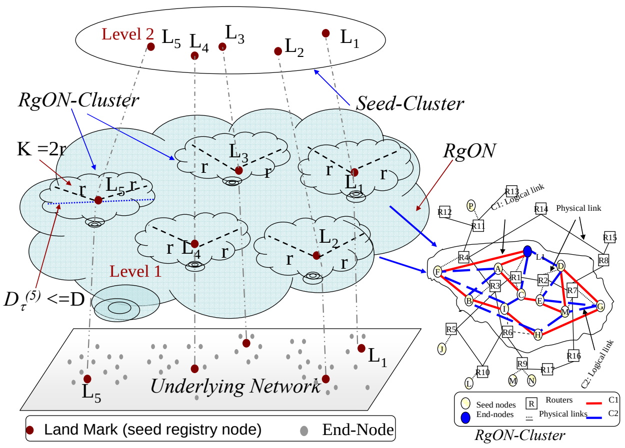

Fig. 2. RgON architecture.

3.2 RgON Cluster

Each *RgON-Cluster* (*RC*) is constructed as a *2m-regular* graph composed of *m* independent edge-disjoint *Hamilton Cycles* (*HC*) [23]. Each node has *2m* neighbors (node connectivity). Those neighbors are labeled as $g_p^{(1)}$, $g_s^{(1)}$, $g_p^{(2)}$, $g_s^{(2)}$, …, $g_p^{(d)}$, $g_s^{(m)}$. For each i, $g_p^{(i)}$ denotes the neighbor node's predecessor and $g_s^{(i)}$ denotes the neighbor node's successor on the *i-th HC*. Figure 2 shows an example of *RC* consists of two *Hamilton* cycles with diameter D_t =3 (logical hops) over the underlying network. The physical distance between the landmark *L1* and any RN node is less than or equal two (i.e. *radius r = 2* and *diameter K=2r*) backbone physical hops between their routers. This paper constructs *RC* as *regular-graph* for three arguments as follows. First, regular graphs are chosen because it is required that all nodes having the same degree for load balance. Second, we construct the *RC* as an intermediate of a completely ordered regular network and a fully random network for achieving two interesting features: high clustering i.e., there is a high density of connections between nearby nodes, which is a characteristic of the regular topologies, and short network diameter. Finally, the *RC* composed of *HC* having the advantage that joining or leaving processes will require only local changes, similar to our previous work [5].

Definition *2* (*Coloring RgON-Cluster*): Assume $|RC_j|_t$ is the number of registry nodes in the RC_j at instance of time *t* and Adv_t is the number of *Advs* at *t*. If $Adv_t \le |RC_j|_t$ $\forall$ RC_j ; *j=1, …, β,* then each RN node stores one advertisement and colored by an associated color. Otherwise, each RN node stores different *Advs* and the associated ω*(v)* different colors slots are filled.

Definition *3 (RgON-Cluster Diameter*): The diameter of the *2m*-regular graph (RC_j) at instance of time *t* is bounded by

$$1+2m+2m(2m-1)+\ldots+2m(2m-1)^{D_t^{(j)}-1} = \frac{2m(2m-1)^{D_t^{(j)}}-2}{2m-2} = N_t(2m, D_t^{(j)}) = |RC_j|_t \tag{1}$$

This value is called the *Moore bound* [9], and it is known that, for $D_t^{(j)} \ge 2$ and $2m \ge 3$. Where $|RC_j|_t$ is the estimated size of *RgON-Cluster* RC_j at instance of time *t*. From equation 1 if $|RC_j|_t - 1 \le (2m)^{D_t^{(j)}}$ [15] then

$$D_t^{(j)} \ge \frac{\log(|RC_j|_t - 1)}{\log 2m} \tag{2}$$

Thus, by estimating the size of the i-th *RC* the diameter $D_t^{(j)}$ can be determined.

The adjacent RN nodes in *RC* are colored with different colors. The non-colored registry node that did not store any advertisement is called *Standby Registry Node* (*SRN*). The number of advertisements Adv_t at an instance of time *t* is equal to the number of different colors C_t. Periodically, each seed (landmark) registry node L_j estimates $|RC_j|_t$ and also it counts C_t by flooding counting request to all RN nodes in RC_j. The *Group Size Estimation* problem is representative of a large class of problems for collecting *statistics* about a large-scale distributed system, in a decentralized manner. Several efforts have been made to provide decentralized solutions to the group size estimation problem. M. Bawa, et al presents several mechanisms for aggregation and group size estimation [18]. One algorithm has an estimation query sent along a random walk within the group. When the estimation query hits a node for the second time, the protocol stops and the number of hops traversed so far can be used to estimate the group size – according to the birthday paradox, it takes $O(\sqrt{n})$ hops for a random walk query to encounter a node a second time. Further studies concerning the network's size estimation is out the scope of this paper.

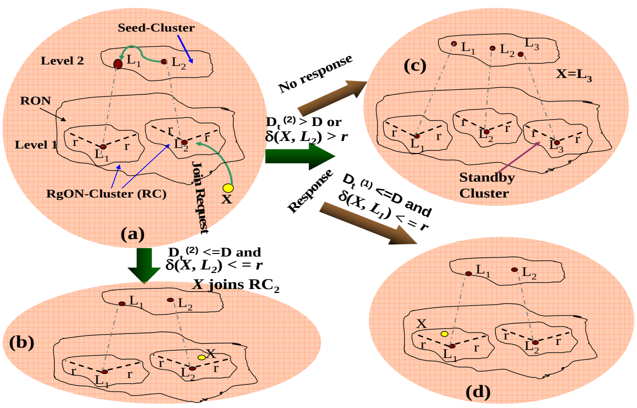

Fig. 3. Example: RgON-Cluster Step-Step Construction.

3.3 RgON Step-Step Self-Construction Algorithm

When an end-user node X is stable and has enough capabilities to be a registry node then it calls *Join_RgON()* process that runs the following steps.

1. X looks up for a seed registry node L_j (landmark RN) by using random walk or physical IP multicast. For example, figure *3-a* shows that node X discovers the seed RN (L_2) of *RgON-Cluster* RC_2 with diameter $D_t^{(2)}$ Then X sends a join request to L_2.
2. If $[(D_t^{(j)}+1 \le D)$ and $(|RC_j|_t > C_t)$ and $(\delta(X, L_j) \le r;$ $K=2r)]$ then node X calls *Join_RgONCluster(RC_j)* process to join the *RgON-Cluster* RC_j as shown in figure *3-b*. The *Join_RgONCluster(RC_j)* process chooses m random registry nodes $u_i \in RC_j; (i = 1,..., m)$ and then inserts node X between each node u_i and its successor node ($u_i \rightarrow$ $\mathbf{g}_s^{(i)}$) in the *i-th HC* similar to our previous work in [5]. Finally node X becomes a standby registry node *SRN* and doesn't store any advertisement.
3. If $[(D_t^{(j)}+1 \le D)$ and $(|RC_j|_t \le C_t)_t$ and $(\delta(X, L_j) \le r;$ $K=2r)]$ then the registry node X calls *Join_RgONCluster(RC_j)* process to join RC_j. Then node X autonomously selects some colors slots form the existing RN nodes in RC_j (for Load balance) and stores the associated *Advs*. For example, node X joins RC_2 as shown in figure *3-b*.
4. Otherwise, if $[\delta(X, L_j) > r$ *or* $(D_t^{(j)}+1 > D)]$, the seed node L_j broadcasts a join request message in the Seed-Cluster and then waits *time-out* τ period for a reply from the seed registry nodes in the Seed-Cluster. There are two cases:
   - No response is received within the time-out interval τ, then node X autonomously creates its own new *RC* and becomes a seed registry node (land mark). The number of seed nodes (*RgON-Clusters*) β increases by one. For example, in figure *3-c* node X becomes a seed RN L_3 of the new *RgON-Cluster* RC_3. This new $RC_{\beta+1}$ is a standby cluster (i.e. all its RN nodes are standby). Its status changes from standby cluster to working cluster when enough number of RN nodes joins it. Thus, we can maintain the <*K, D*>-interleaving property with load balance among *RgON-Clusters*.
   - If L_j receives multiple of replies then it selects the L_s with the smallest distance to X where $\delta(X, L_s) \le r$, $(D_t^{(s)}+1 \le D)$ and the minimum $|RC_s|_t$. Then node X calls *Join_RgONCluster(RC_s)* process to join RC_s. For example node X joins RC_1 as shown in figure *3-d*.

4. Performance Evaluation

This section describes the simulations to demonstrate that the <*K, D*>*-interleaving RgON* step-step construction scheme enables efficient WS discovery. To evaluate the effectiveness of the proposed <*K, D*>*-interleaving* step-step construction scheme of *RgON* on the discovery of service's advertisement overhead, we first compare it with *K̃-interleaving* [17]. The *K-interleaving* constructs *RgON* with topology awareness. It organized the *RgON* into *RgON*-Clusters, where the physical distance between any nodes in each *RC* is less than or equal to *K* physical hops. The *K-interleaving* expands the *RC* without size limit. In addition, we use the following metrics.

- **Service Discovery Delay (*SDD*)**. It measures the communication delay to discover a service's advertisement within *RC*. It defines the communication delay between a requester RN to the RN that stores the required service's advertisement.

- ***Average Service Discovery Delay* (*ASDD*)**. It defines the average of the *SDD* values for all *RgON-Clusters*. In the simulation each RN belongs to *RC* sends a discovery request and then determines the required *SDD* to discover the required service's advertisement. Thus, each *RC* determines the average of SDD within it. Finally, the *ASDD* is calculated.
- ***Physical Link Stress***, a measure of how the proposed <K, D>-interleaving constructing scheme for *RgON* that is effective in distributing network load across different physical links. It refers to the number of identical copies of a packet carried by a physical link for service discovery. In the simulation we pick random node that send a service discovery requester and then measure the worst stress value of all physical links.

We argue the overhead for publishing (replicating) the services' *Advs* to two categories. First, the required communication delay to publish a new advertisement. All members of the Seed-Cluster are involved in such communication. In the simulation each seed registry node measures the required time to broadcast the new advertisement within the Seed-cluster. Then the average will be calculated so called ***Average Communication Overhead for Publishing*** (***ACOP***). Second, the *resource's* utilization (e.g. storage size) that is required to store the *Advs* in each *RC*. In the simulation, the total required storage in *RgON* is calculated as the summation of the required storage at each *RC* and is denoted by ***Total Storage Overhead (TSO)***.

4.1 Simulation Setup

The simulation consists of *100* backbone routers linked by core links over the underlying topology *transit-stub* model. The *Georigia* Tech [25] random graph generator is used to create that network model. Random link delay of *4-12*ms was assigned to each core link. The *RNs* were randomly assigned to routers in the core with uniform probability. Each RN was directly attached by a LAN link to its assigned router. The delay of each LAN link was set to be 1ms. The *transit-stub* network model consists of three stub domains per transit node, with no extra transit-stub or stub-stub edges. The edge probability between each pair of nodes within each stub domain is *0.42*, *0.6*, and *1.0* respectively. The simulation ran with different number of registry nodes *N* that is ranged from *100 - 1000*. It did not take into consideration the required queuing time for the advertisement discovery requests. Only one replica of each advertisement is replicated into each *RC* (i.e. ρ = *1*). Moreover, the simulation assumed that *100 Advs* have been published in *RgON* and each advertisement is *1* KB. Thus, the simulation easily determines the storage overhead required to publish these *Advs* based one the number of the *RgON-Clusters* β.

4.2 Simulation Results

Figure *4-a* plots the variations of the *ASDD* along with the number of registry-nodes over two registry overlay networks that have been constructed by *K*-interleaving [17] and <*K*, *D*>-interleaving schemes, where *K=4* and *D=4*. It shows that the *ASDD* over *RgON* constructed by <*K*, *D*>-interleaving scheme are shorter than the one constructed by *K*-interleaving scheme. The <*K*, *D*>-interleaving improves the *ASDD* by *20*%. Figure *4-b* shows the variations of the *ASDD* along with the network size where *K=4* and *D =3*, *4* and *5*. Clearly, the *ASDD* increases as *D*, the diameter of *RC* increases. The improvement ratios of the *ASDD* are as follows: <*4*, *3*>-interleaving over <*4*, *5*>-interleaving is *53*% and <*4*, *4*>-interleaving over <*4*, *5*>-interleaving is *20*%.

The right part of the figure *4-b* with *D=3* shows that the *ASDD* lies within the range *20-23*ms. Clearly the *ASDD* values are oscillated in small interval while the number of registry nodes is increased. The lesson from figure *4-b* is

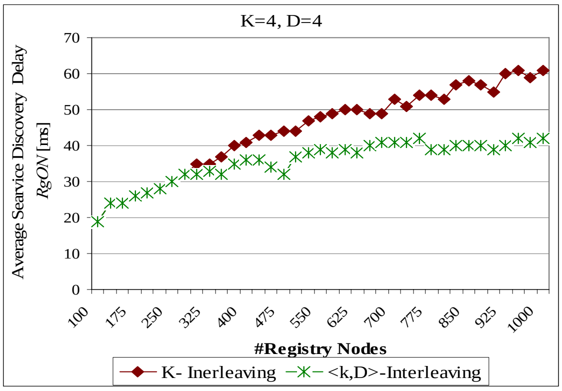

Fig. 4-a

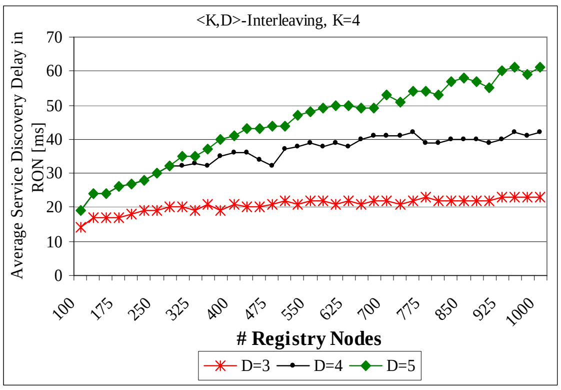

Fig. 4-b

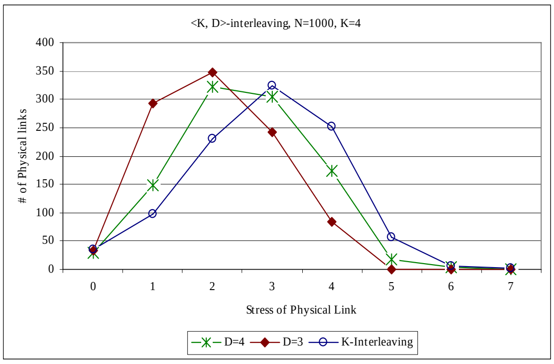

Fig.4-c Physical Links Stress

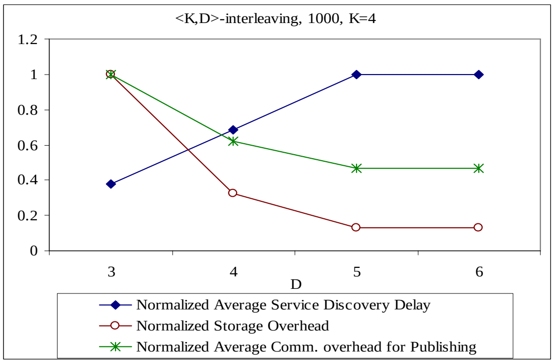

Fig. 4-d Trade-off

that *ASDD* grows slowly with the number of registry nodes, confirming the simulation results that demonstrate the proposed <*K*, *D*>-interleaving scheme constructs *RgON* that enables *efficient* and *scalable WS* discovery. In addition, Figure *4-c* plots the number of physical links suffering from a particular stress level on the Y-axis, against the range of stress levels on the X-axis. It proves that the traffic overhead induced over *RgON* constructed by <*K*, *D*>-interleaving is less than the traffics overhead induced over *RgON* constructed by *K*-interleaving by *11*% when *N=1000, K= 4, D=4* while it is *28*% when *D=3*. In addition, it shows that the number of physical links which have smallest stress are large when *D*=3 compared to when *D=4* and *5*. It proves that <*K*, *D*>-interleaving scheme achieves low traffics when *D* is small. Figure *4-d* represents the trade-off relation between the *ASDD* and the publishing overhead of the services' *Advs*. It plots the variations of the normalized values of the *ASDD*, *TSO* and *ACOP* along with the variation of the diameter *D*. It shows that the *ASDD* increases as the diameter *D* increases however, the *TSO* and *ACOP* decreases as the diameter increases. To control that trade-off relation adapting the couple *K*, *D* autonomously is required.

Moreover, Table 1 shows the efficiency and overhead of the <*K*, *D*>-interleaving compared to the *K*-interleaving in both dense and sparse networks, with *K=4*. Efficiency represents the improvement ratio of the *ASDD* in <*K*, *D*>-interleaving over K-interleaving. However, overhead represents the increasing ratio of the *ACOP* in <*K*, *D*>-interleaving over K-interleaving. The simulation employs the transit-stub domain network with different number of routers *50* and *500*. The registry *nodes, 1000*, were randomly assigned to routers in the core with uniform probability. Table 1 demonstrates that the <*K*, *D*>-interleaving is more efficient in dense networks than in sparse ones. The *RgON* network status changes from dense to sparse or vice versa due to registry nodes dynamically join and leave it. Further research direction is to develop a hybrid protocol that switches from *K*-interleaving scheme to <*K*, *D*>-interleaving and vice versa based on the *RgON* network status. Finally, the simulation results mentioned before give an insight that the proposed <*K*, *D*>-interleaving scheme constructs *RgON* that enables *efficient* and *scalable* WS discovery.

5. Conclusion

This paper, clarifies the concept, architecture, <K, D>-interleaving *RgON* construction and WS discovery technologies of the *ACCI*. The simulation results has depicted that the construction and *WS* discovery technologies are scalable. We are currently extending this work into several directions. First, *ACCI* considers QoS and consistency of WS replica in the *WsON* as important issues that need to be addressed. This paper relaxed the consistency problem by replicating read-only WSs. Second, we believed that *ACCI* must handle the security issues. Finally, we are studying how to enhance the *WS* discovery's cost with acceptable construction and maintenance overheads in a constant dynamic environment.

Table 1. Efficiency and Overhead of <k, D>- interleaving over K-interleaving in sparse and dense networks

	Dense (50 routers)		**Sparse** (500 routers)	
D	*Efficiency*	*Overhead*	*Efficiency*	*Overhead*
3	66%	65%	39%	35%
4	31%	53%	0%	8%
5	17%	30%	0%	0%
6	0%	0%	0%	0%

Reference

[1] K. Ragab, A. Yonezawa, "Autonomic K-interleaving Construction Scheme of P2P Overlay Network," Proc. LNCS of ATC 2006, September, 2006 China.

[2] F. B. kashani, C. Chen, C. Shahabi, "WPSDP: Web Services Peer-to-Peer Discovery Service, "Proc. Of Int. conf. on Internet Computing, pp 733-743, 2004.

[3] K. Verma, Et. al. "METEOR-S WDSI: A Scalable P2P

Infrastructure Registries for Semantic Publication and Discovery Web Services," Journal of Information technology and Management, 2004.

[4] Jeffrey O. Kephart, Divad M. Chess, "The Vision of Autonomic Computing," IEEE Computer, PP. 41-50, 2003.

[5] K. Ragab, Et. al. "Autonomous Decentralized Community Communication for Information Dissemination," IEEE Internet Computing Magazine, Vol.8, No.3, pp.29-36, May/June 2004.

[6] M. Paolucci, Et. al. "Using DAML-S for P2P Discovery, " Proc. Of the Int. conf erence on Web Services, 2003.

[7] Bantz, D.F., Et. al. "Autonomic personal computation," IBM Systems Journal, vol. 42, no.1, 2003, pp. 165-176

[8] "An architectural blueprint for Autonomic Computing," White page, 3rd edition June 2005.

[9] N. Biggs, "Algebraic Graph Theory", Cambridge Math. Library ISBN 0-521-45897-8 pbk, (1974, 1993 (2nd edition)).

[10] Kenneth P. Birman, "Reliable Distributed Systems Technologies, Web Services and Applications," Springer 2005.

[11] Brendon J. Wilson, "JXTA", New Riders Publishing 1st edition June, 2002.

[12] E. Cerami, "Web Services Essentials," O'Reilly Media, Inc.; 1st edition, February, 2002.

[13] "SOAP Specification version 1.2," http://www.w3.org/TR /soap12 , 2003.

[14] "JXTA Project. JXTA-soap Project," http:// soap.JXTA.org/ servlets/ProjectHome.

[15] A. H. Dekker, B. D. Colbert, "Network robustness and graph topology," ACM Proc. 27th Int. Conf. on Australian Computer Science, pp 359-368, 2004.

[16] OASIS standards consortium, "Universal Description, Discovery and Integration of Web Services (UDDI)," Version 2.0, 2002, http://www.uddi.org/

[17] K. Ragab, Y. Oyama, A. Yonezawa "K-Interleaving Rendezvous Overlay Network Construction Scheme," Proc. Of IEEE, ICIS2006, Julay 2006, USA.

[18] M. Bawa, H. Garcia-Molina, A. Gionis and R. Motwani, "Estimating aggregates on a peer-to-peer network," Technical Report, Dept. of Computer Science, Stanford University, 2003.

[19] Peter Brittenham, "Autonomic Computing: An insider's Perspective," June 2005, http://www-128.ibm.com/ developerworks/autonomic/library/ac-inside/index.html.

[20] Shuping Ran, "A model for web services discovery with QoS," ACM SIGeCom Exch. Journal, Vol. 4, Issue 1, pp. 1-10, 2003.

[21] M. Blaum, J. Bruck, and A. "Vardy. Interleaving schemes for multidimensional cluster errors," IEEE Transactions on Information Theory, 44(2):730–743, 1998.

[22] M. Blaum, J. Bruck and P.G. Farell, "Two-dimensional Interleaving Schemes with repetitions," IBM Research Report RJ 10047, Oct. 1996.

[23] B. Jackson and H. Li, "Hamilton Cycles in 2-Connected k-regular Bipartite Graphs," Comb. Theory Series A, 44 (2): 177-186, 1998.

[24] I. Sotica, R. Morris, D. Karager, F. Kasshoek, H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for Internet Applications," ACM SIGCOMM'01, August 2001.

Khaled Ragab
He is an assistant professor of College of Computer Science and Information Technology, King Faisal University, Saudi Arabia. Also he is on leave assistant professor of at Department of Mathematic, Computer Science division, Ain Shams University, Cairo, Egypt. He joined Department of Computer Science, Tokyo University in 2005 as postdoctoral position. He was born in 1968 and received his B.Sc., M.Sc. degrees in Computer Science from Ain Shams University, Cairo, Egypt in 1990, 1999, respectively and Ph.D. degree in Computer Science from Tokyo Institute of Technology in 2004. He has worked in Ain Shams University, Cairo Egypt in 1990-1999 as assistant lecturer. He has worked as research scientist in Computer Science Dept., Technical University of Chemnitz, Germany in 1999-2001. His research interests include autonomic computing, Peer-to-Peer Systems, Overlay Networks, Web-services and application-level multicast.