Peer Clouds: A P2P-Based Resource Discovery Mechanism for the Intercloud

LOHIT KAPOOR[1,2], SEEMA BAWA[1], and ANKUR GUPTA[2]
[1]Thapar University,[2]Model Institute of Engineering and Technology

The Intercloud represents the next logical step in the evolution of cloud computing overcoming issues of data/vendor lock-in and dealing with volatile service requests. However, resource discovery across heterogeneous Cloud Service Providers (CSPs) remains a challenge. In this paper we present a P2P-based distributed resource discovery mechanism based on spatial-awareness of cloud data-centers belonging to different Cloud Service Providers. The scheme is based upon exploiting location information of Data Centers and organizing them into DHT peers for optimal communication. It thus allows for QoS-compliant resource/service provisioning across Cloud Service Providers (CSPs). Simulation results establish the effectiveness of the proposed scheme.

Keywords: Intercloud, Resource/Service Discovery

1. INTRODUCTION

Small or medium-sized Cloud Service Providers (CSPs) are usually limited in terms of serving capability due to limited compute services in their data centers. The problem gets exacerbated during peak hours when the demand is very high increasing the probability of non-servicing of user service request. Due to the nature of cloud computing cloud vendors need to dynamically provision resources from other vendors to create the illusion of "on-demand elasticity". An intercloud [Buyya et al. 2010] architecture connecting different cloud-service providers, therefore becomes unavoidable in this context. An Intercloud system is a federated environment comprising data centers belonging to different cloud vendors facilitating resource discovery and provisioning based on well-defined economic principles. For small and medium CSPs anintercloud environment allows dynamic scaling of resources reducing request drop and violation ServiceLevel Agreements (SLA's).

Resource discovery is a major challenge in the successful implementation of a federated intercloud environment. Discovery and management of resources in an intercloud federation can typically be done in a centralized or decentralized manner. Most of the existing techniques [Buyya et al. 2010; Nikolay and Buyya 2012] for resource discovery and scheduling utilize a centralized mechanism. In this method each cloud interacts with a central entity or a meta-broker, submits all the required information to it and then a meta-scheduler takes control to assign services across CSPs to a job accordingly. However, a centralized approach to service management and discovery does suffer from some obvious shortcomings like performance vs. scalability, security vulnerabilities and single-point-of-failure. Further, in thecentralized approach, intercloud resource allocation requests are forwarded to the meta-brokerwhich then directs these jobs to the local brokers at each CSP. In this case regular coordination between local brokers and the meta-broker is required since local resource availability changes dynamically. The meta-broker cannot make any presumptions based on the previous known state of the local services. Thus, implementing a best-fit approach in this case requires collating real-time information from all the participating local-brokers which can be challenging. Resource discovery in an intercloud environment plays a critical role in order to implement a well coordinated federation of CSPs to avoid user request drop and delayed request response. Moreover, resource information in a

Resources in an intercloud represent virtual machines, platforms, native and third-party services across all cloud models - IaaS, PaaS and SaaS. Resource and service discovery is therefore used interchangeably throughout the paper.

federated environment should be up to date and each CSP in the federation should be aware of the status of the other CSPs. Due to the geographical distribution of the data centers belonging to different CSPs communication latency can become a major performance bottleneck. Thus, any efficient service discovery strategy for the intercloud environment should attempt to minimize the communication latency by taking into account the geographical location of the data centers. With centralized brokers and schedulers, it is not always possible to place them in close proximity to all data centers. Thus, some CSPs end up paying a higher communication cost than others each time resources from other CSPs are requisitioned.

2. RELEVANT WORK

Authors in [Buyya et al. 2010],[Nikolay and Buyya 2012]has presented several early works related to federated clouds with service discovery based on negotiation held in a centralized exchange. This market-based centralized model is prone to single point of failure besides presenting scalability issues. NWIRE (Net-Wide-Services) [Schwiegelshohn and Yahyapour 1999] is a meta-computing scheduling architecture based on brokerage and trading and is a market system between sub domains. Global Inter-Cloud Technology Forum (GICTF) [GICTF] is an intercloud forum where service discovery is based on collection of services, their selection by a central entity. Another intercloud service discovery strategy based on clustering of services based on past service experience is presented in [Sotiriadis et al. 2012].

However, the clustering scheme presented is based on putting together transient services and suffers from overheads of creating and disbanding of the clusters. Moreover, keeping track of past service experiences of each participant involves its own overheads. InterGrid [Huang et al. 2012] is a cross-grid cooperation architecture composed of a set of InterGrid Gateways (IGGs) responsible for managing peering arrangements between grids. The InterGrid Gateways employed upon the top of each participating grid are distributed in a decentralized manner for efficient service discovery. However, the framework provides no fault-tolerance mechanism for the IGGs, failure of which can result in islands of grids being created resulting in a disconnected network. Authors in [Gupta et al. 2011] suggested a completely decentralized peer-to-peer framework for dynamic service provisioning across cloud service providers. However, the scheme is not optimized for latency by considering the geographical location of the data centers. Bessis et al. [Sotiriadis et al. 2012] also presented meta-scheduling model in intercloud environment to engage in drawbacks exist in centralized models. Along with it also undertake bottleneck in concurrent requests in intercloud environment during peak hours. Nelson et al [Nelson and Uma 2012] present an Intercloud Service Provisioning System (IRPS) in which each service and task represented semantically using service ontology. Further they use present a set of inference rules for discovery and semantic scheduler. Some instances of decentralized service discovery are available in grid computing. This paper presents a peer-to-peer based decentralized and distributed service discovery and selection mechanism for the intercloud environment. This proposed model ensures that communication latency within the network of Data Centers is minimized and service requests are serviced by data centers which are relatively closer to the requesting data center. The rest of the paper is organized as follows: Section II presents a detailed discussion of the proposed system model. In Section III the sequence of operations of the proposed framework are illustrated, while in Section IV some early simulation results based on a custom simulator are presented. Finally, Section V concludes the paper and presents some directions for future work.

3. SYSTEM MODEL

A Cloud Service Provider (CSP) consistsof multiple data-centers located in different geographic locations across globe. A central broker manages the service requests from users within the CSP. It is assumed that each CSP under consideration participates in a federation of CSPs. In the model, each data-center of a CSP has a Resource Manager (RM) for maintenance of internal services of a data-center and a Remote Resource Manager (RRM) which keeps track of resource

information from other participating data centers. The RRMs belonging to a particular geographical location are organized into the different Local Groups (LG). One RRM in each group assumes responsibility for acquiring all the required resource information from other peer RRMs located in the respective LG through resource availability advertisements.A virtual network overlay of allsuch RRMs is created to facilitate exchange of resource information. This virtual network overlay is called the Super Group (SG). Figure 1 provides a schematic of the proposed scheme in which different datacenter belonging to different CSPs forms different local groups with a chosen RRM from each LG participating in the global Super Group.

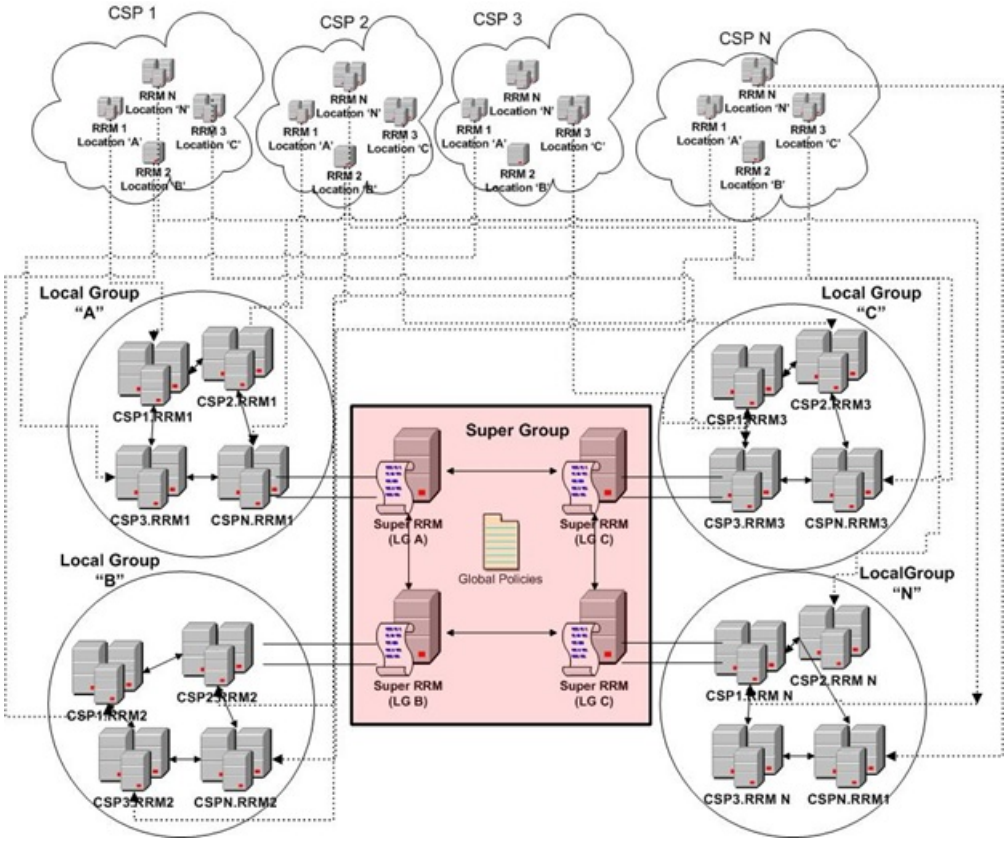

Figure. 1: Schematic view of Resource discovery in intercloud

Let RRMi (i=1, 2, 3?, M) be the set of M Remote Resource Managers (RRMs), representing individual data centers in the federated intercloud environment. Each RRMi belongs to a LG comprising M data centers, which may be designated by DCi1, DCi2???DCir?. DCiM. Theoretically Mcan vary as data centers join and leave the P2P network, but for simplicity we assume that data centers continue to be a part of the federation even if they have no services to offer or are not actively seeking services. Thus,

$$\|LG_i\| = \|RRM_{ir}\| = \|DC_{ir}\|$$

$$\text{where } i > 0, i = 1, 2?., M \text{ and } i \leqslant r \leqslant M$$

Each data center within the federation puts out a Resource Availability (RA) status periodically in the form of advertisements. The RA is typically expressed in terms of Resources (RES) and their associated cost (C), where each resource can be a virtual machine, platform or service.

Thus, RA is the set of resource, cost tuples advertised by each RRM within the LG and cached by the super RRM which participates in the SG.

$$RA = (RES_1, C_1), (RES_2, C_2).??.(RES_N, C_N)$$

Where X is the number of resources offered for remote use by a particular data center at a particular time. X varies based on the resource demand at the data center. Thus, other RRMs need to cache only the last RA's issued by RRMs of other data centers since it accurately represents the state of available services. Moreover, the cost associated with the resources is also a part of the RA.

Other data centers which are desirous of availing services within the federation put out a Resource Request (RR) advertisement which is again expressed in terms of required RES and desired cost.

$$RR = (RES_1, C_1), (RES_2, C_2).??.(RES_K, C_K)$$

where K is the number of resources required by the requesting RRM. The objective for the requesting RRM is to locate another RRM such that

$$RRK \approx RAX$$

where K¡¡=X, so that the number of resources available at the prospective partner RRM is more than or equal to the number of resources requested. The RR from a particular RRM is first attempted to be serviced within the LG. Each RRM already has the cached RA advertisements from other RRMs within the LG. If the resource availability within the LG is not met, the requesting RRM sends a "Remote Resource Request" (RRR) to SG. If the resources requested in the RRR are available at a particular RRM, the RRM sends the details of the RMs to the requesting RRM. If none of the RRMs within an LG meet the requested services, the RRR is propagated further within the SG until the request is met or all options are exhausted.

An obvious challenge in all resource discovery strategies is to manage the trade-off between resource cost and latency. The cheapest resources could be located the farthest and latency adds its own costs in terms of data transfer costs and communication overheads. This choice needs to be made by the requesting RRM. For instance, a high-priority user service request with stated SLAs may be serviced by choosing a RRM with the lowest latency i.e. closest to the requesting RRM which also meets the cost criteria.On the other hand a low priority user request may be serviced by a best-fit approach in which cost may be given more weight over latency to maximize the profit of the requesting RRM. The RR or the RRR requests issued can reflect the relevant priorityof cost or latency. Further, to handle scenarios where an RRM may not want its data to be processed at a particular geographical location, a conditional RRR can be issued which prevents the query being forwarded to the excluded locations.

4. SEQUENCE OF OPERATIONS

4.1 Joining Process for new RRM

If the RRM is first in the network, it assumes the role of the Super RRM. Since RRMs represent data centers, they can be assumed to be available at all times and hence usual mechanisms of having seed peers or landmark peers to assist in the peer join process are not needed. For subsequent joins of RRM's the request is responded by the super RRM. With the increase in the size of LG, requests get cached on all intermediary RRMs that they pass through. ThusRRM join timesare subsequently lowered. The newly entered RRM is now capable to receive RA and sent RR advertisement from/to other RRMs. RRMs joining the LG in Peer Clouds specified in Algorithm 1.

Algorithm 1 RRMs joining the LG in Peer Clouds

```
1: for all RRM ∈ Peer Cloud do
2:   if RRM does not ∈ LGi then
3:     locateSuperRRM(myRRMID, regionID);
4:     if !superRRM then
5:       newSuperRRMID = becomeSuperRRM(myRRMID);
6:       LG = createLG(myRegionID);
7:       joinSG(newSuperRRMID, regionID);
8:     else
9:       registerWithSuperRRM(myRRMID);
10:      joinLG (LG);
11:    end if
12:  end if
13: end for
```

4.2 Super RRM Selection

Selection of RRM as a Super RRM is done on first come first serve basis i.e. the first RRM to join a LG nominates itself as SuperRRM for a particular region. Subsequent RRMs retain their joining rank in the LG. The Super RRM acts as a Gateway to the SG by collating resource advertisements from other RRMs within the LG and sharing it within the group of super RRMs. In the unlikely event that the Super RRM fails, the next ranking RRM takes over as the Super RRM. This process is initiated if the Super RRM does not send out a special status message during a designated time period. Each RRM continuously generates an RA (resource availability) status message 5 minutes which holds the current status of resources and their associated cost and circulates it within the LG. Each RA message has a time-to-live parameter associated with it to ensure that older messages do not remain in circulation. The RA status messages are cached by other RRMs in the LG and used to initiate a contract agreement with them based on future service.

4.3 Resource Discovery

The process of resource discovery is coveredby two types of constraints a) costor b) resource specification which are part of the resource request advertisements.Specific requests which are not serviceable within the LG due to lack of resources or not meeting cost constraints are then put out in the SG for possible resource provisioning. The SuperRRM propagates the request to other SuperRRMs in the SG which propagate the requests further within their respective LGs. RRMs which fulfill the resource criteria specified in the advertisement contact the advertising RRM directly. The algorithm for resource provisioning is illustrated below (Algorithm 2) while a sample resource advertisement is depicted in Figure 2. Selection of resources by any RRM can be performed on the basis of "latency" (proximity)or "cost" or both.

Algorithm 2 Algorithm for resource lookup and provisioning in PeerClouds at each RRM.

```
1: advertiseResources(resourceVector, costVector) // RA
2: advertiseRequirements(resourceVector, constraintsVector) //RR
3: processResponse (responseVector)
4: for all each response in responseVector do
5:    rankResponse(response)
6:    selectedRRM = getTopRRM()
7:    sendCon
      rmation(selectedRRM)
8:    processRequest (request)
9:    if evaluateRequest(request) then
10:      sendConfirmation(request.getRRM())
11:   end if
12: end for
```

<**RRM:Resource Request Advertisement**>
< Resource Description="Resource Description for Individual RRM" >
<RRM ID type="UUID" description ="RRM's ID"/>
< Resource type ="String" description =Virtual Machine/Service/>
<Resource quantity = "Uint32" description = "Number of VMs">
<Bandwidth Type ="String" description ="Minimum bandwidth required"/>
<Platform type="String" description ="Specific operating system/platform required"/>
<VM Config>
< CPU type ="Uint32" description ="Number of cores"/>
<Storage type ="Uint32" description = "Hard disk space "/>
<Memory type = "Uint32" description = "Minimum RAM">
</VM Config>
<Constraints>
<Cost type = "double" description = "cost constraint for resource/hour"/>
<Cost Weight type = "double" description = "weight "/>
<Latency type="double" description = "desired latency"/>
<Latency Weight type="double" description = "desired weight"/>
</Constraints>
<Service Description = "Service Description for individual RRM">
<Service name="String" description = "Service ID">
<Service instances="Uint32" description = "Number of instances required">
<Platform type="String" description ="Specific operating system/platform required"/>
</Service Description >
</Resource Description>
</**RRM:Resource Request Advertisement**>

Figure 2: Sample Resource Request Advertisement

5. EXPERIMENTAL SETUP AND RESULTS

To evaluate the effectiveness of scheme 30 physical machines each with configuration shown in Table I are deployed. Devstack [OPENSTACK] is used to create a local cloud which provides an option to install and run Openstack (software to control the cloud) on local systems. It enables user to create, control and destroy virtual machines. A number of 150 virtual machines with configuration as shown in Table II are created.

For peer to peer deployment, we also implemented the JXTA [JXTA] java based protocol for

Table I: Physical Machine configuration

OS	CPU	HDD	Memory(MB)
Ubuntu 14.04 (Trusty)	Intel Core i7-2600 @3.4 GHhz	500 GB SATA	4GB

Table II: Virtual Machine configuration

OS	CPU	Cores	Memory(MB)
Windows Server 2012	Intel Xeon E52670	1	1024

creation and maintenance of our P2P network. JXTA utilizes the Distributed Hash Table (DHT) for organizing the P2P overlay as a hierarchical topology. However, it relies on rendezvous peers to maintain and distribute routing indices for normal peers and the resources/services that they provide. Queries are forwarded to rendezvous peers to locate the actual peer on which the desired resource/service resides. The reason for using JXTA is: a) Supports Interoperability required in intercloud b) Platform and Language independence for heterogeneous environment in intercloud c) Ubiquity (any virtual machine can be a peer) d) Open standards (XML) for advertisement and communications

Each VM constitutes a JXTA peer which depicts a RRM corresponding to each datacenter. Therefore a P2P network of participating RRMs is created. We have used real world network latency measurements by [NetworkDelay]. These latency measurements are utilized for the optimized LG construction. Inter-continental network latency measurements were also used to model communication delays within the SG. Cloudsim 3.0.1 is used to generate the workload in the form of cloudlets for each VM. These cloudlets are then converted in the form of resource queries for each RRM under following parameters (Table 3):

In the first experiment we measured the startup time for 10 to 50 participating RRMs with one designated SuperRRM in a Local Group. The aim of the experiment is to observe the cumulative time for the initial configuration and organization of a Local Group. It is clear from Figure 3 that as the number of participating RRMs increases the overall startup time per RRM reduces from 9.3 seconds/RRM (for 10 RRMs) to 8.2 seconds/RRM (for 50 RRMs). This is due to the impact of super RRM startup time and resource aggregation on the overall time gets averaged out. The startup time includes the JXTA initialization time per peer/RRM as well.

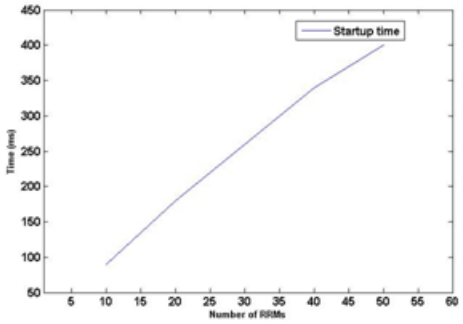

Figure 3: Startup time with varying number of RRMs

A variety of timing measurements for two different types of operations and resource discovery queries within the test setup were obtained for varying topology sizes.

Figure 4 provides the time taken for a new RRM to join the existing setup. Average time ranges from 770 to 860 ms for topologies with 10 to 50 RRMs within a LG. The join process for a new RRM comprises initialization time plus JXTA peer join time plus the time taken for the RRM to connect with the Super RRM.

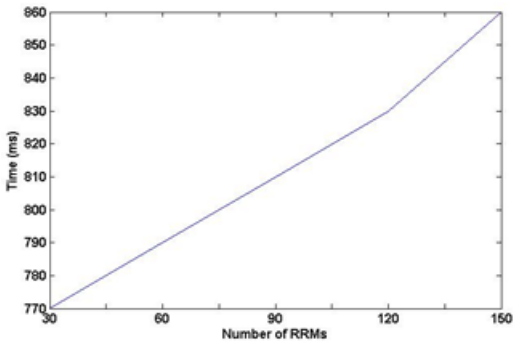

Figure 4: Average join time for a new RRM as a function of LG size

To evaluate the performance of resource queries following parameters (Table 3) were used:

Table III: Cloudlets/Queries parameters

Parameters Name	Ranges
cloudletLength (the length or size (in mips) of this cloudlet to be executed per VM)	(1000 to 5000 mips)
pesNumber (CPU cores per VM)	1
Resource request frequency (The number of requests per unit time)	2 5 per minute
Duration of resource usage (Time to hold a resources)	30 60 minutes
Flash-crowd scenario frequency (Peak hour time)	once every 3 hours
Flash-crowd scenario duration (Time duration of peak hour)	10 minutes
Flash-crowd resource request frequency (The number of requests per unit time)	15 20 per minute
resCost (Cost requested per resource)	0.20−0.40 per hour

In Figure 5 to 6 we present the Request Service Rate (RSR) and Response Time (RT) within an LG for varying number of RRMs. We observe that the RSRremains linear with varying number of queries. The size of the LG has a direct bearing on RSR. Thus, a larger size of LG results in lower number of resource queries being forwarded to the SG.

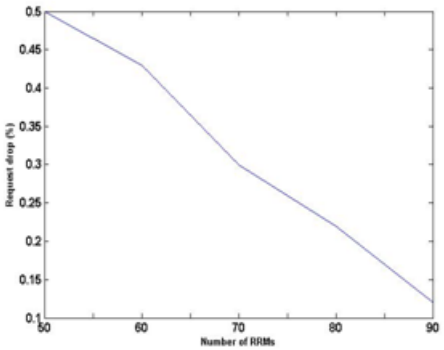

Figure 5. Request Service Rate

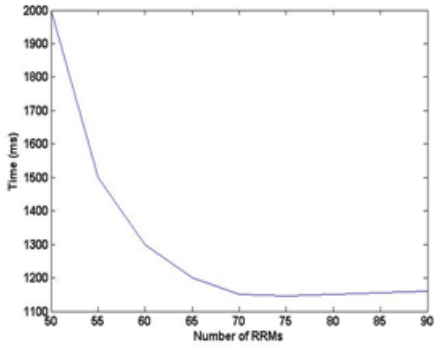

Figure 6. Response Time

In the coming experiments we evaluated resources query responses from LG and SG under following preferences set by resource query generator/user:
a) Latency based resource query (LRQ): In this type of resource query there is an attempt to look out for resources which fall under pre-defined latency.
b) Cost based resource query (CRQ): In this type of resource query there is an attempt to look out for resources which falls under pre-defined cost.
c) Hybrid resource query (HRQ): It attempts to find resources which fall under the response time while maintaining the requested costs.

For LRQ, about 7 % of the queries were serviced by the SG and 93% of the queries were serviced by the LG. Further there is an average increase of 4 1%in response time when the responses come from SG as compared to LG owing primarily to communication delays shown in Figure 7. For CRQ, about 43% of the queries were serviced by the SG and 57% of the queries were serviced

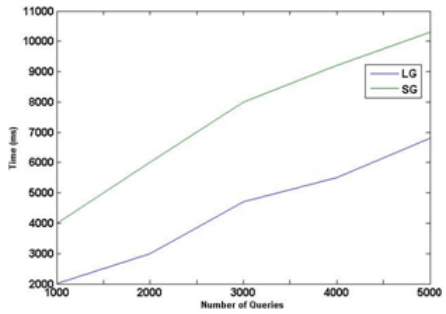

Figure 7: Average resource query response time for varying number of queries (LRQ)

by the LG. As shown in the Figure 8, the queries serviced by SG suffers very high overhead (communication delay), resulting in high response time. However for HRQ as shown in Figure 9, 93% of queries were serviced within LG and 7% from SG and the resultant response time remains marginal high to LRQ and below CRQ. In Figure 10, a complete 24 hrs result is displayed where we can observe that during flash crowd scenario (i.e. after every 3 hrs) CRQ responded in lowest time followed by HRQ and then LRQ. This is due to the reason that in CRQ, 43% of requests are serviced by SG which hold sufficient resources for the requests, while in the case of LRQ 93% requests are serviced in LG which are insufficient during peak hours resulting in high waiting time for the requests. However in normal conditions LRQ serviced the requests in lowest time if compare to CRQ and HRQ.

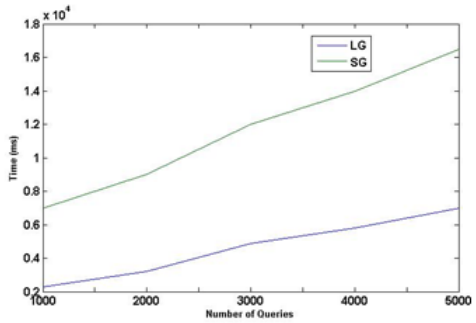

Figure 8: Average resource query response time for varying number of queries (CRQ)

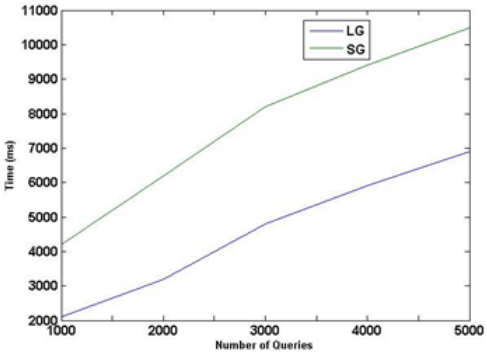

Figure 9. Average resource query response time for varying number of queries (HRQ)

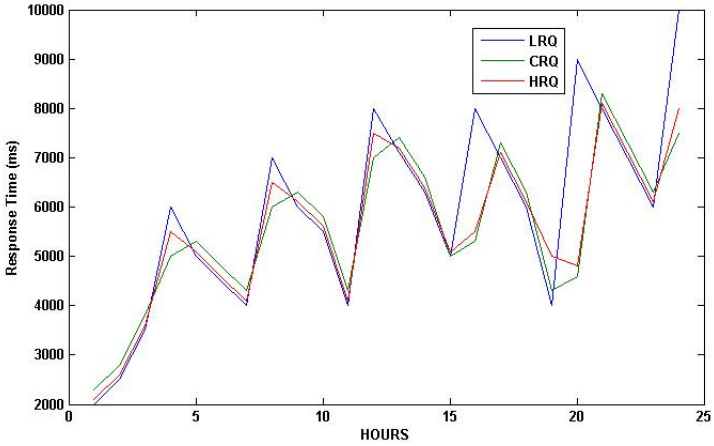

Figure 10. Comparative view of CRQ, LRQ and HRQ

6. CONCLUSION AND FUTURE DIRECTIONS

This paper presents an intercloud service discovery mechanism which consists of two levels of groups local and global, inter-connected to each other. The application of P2P strategies for service discovery in the intercloud environment has not been explored before. The use of JXTA

based implementation provides some inherent benefits such minimized response time which are suited to the intercloud environment. Future work shall involve incorporating elements of quality of service parameters (like availability, reputation etc.) for service discovery and selection mechanism allowing for greater QoS to be leveraged by participating data centers.

REFERENCES

Buyya, R., Ranjan, R., and Calheiros, R. 2010. Intercloud: Scaling of applications across multiple cloud computing environments pp 13-31. In *ICA3PP 2010. 10th International Conference on Algorithms and Architectures for Parallel Processing*.

GICTF. Gictf white paper.

Gupta, A., Kapoor, L., and Wattal, M. 2011. Cloud-to-cloud (c2c): An ecosystem of cloud service providers for dynamic resource provisioning. In *CCIS 190*. Springer, 501–510.

Huang, Y., Bessis, N., Norrington, P., Kuonen, P., and Hirsbrunner, B. 2012. Exploring decentralized dynamic scheduling for grids and clouds using the community-aware scheduling algorithm. In *Future Generation Computer Systems*. Elsevier, 402–415.

JXTA.

Nelson, V. and Uma, V. 2012. Semantic based resource provisioning and scheduling in inter-cloud environment. In *ICRTIT*. IEEE, 250–254.

NetworkDelay.

Nikolay, N. and Buyya, R. 2012. Inter-cloud architectures and application brokering: taxonomy and survey. In *John Wiley and Sons*.

OPENSTACK.

Schwiegelshohn, U. and Yahyapour, R. 1999. In *High-Performance Computing and Networking*. Volume 1593 of the series Lecture Notes in Computer Science. Springer-Verlag, 851–860.

Sotiriadis, S., Bessis, N., and Antonpoulos, N. 2012. Decentralized meta-brokers for inter-cloud: Modeling brokering coordinators for interoperable resource management. In *FSKD*. IEEE, 2462–2468.

Sotiriadis, S., Bessis, N., and Kuonen, P. 2012. Advancing inter-cloud resource discovery based on past service experiences of transient resource clustering. In *Third International Conference on Emerging Intelligent Data and Web Technologies (EIDWT)*. IEEE, 38–45.

Lohit Kapoor is a Research Scholar in Computer Science and Engineering Department at Thapar University, Patiala, India under the guidance of Dr. Seema Bawa and Dr. Ankur Gupta. He has presented and published a number of papers in various reputed International Conferences (IEEE, Springer, ACM etc). His area of interest includes Cloud Computing and Distributed Systems.

Dr. Seema Bawa holds M.Tech (Computer Science) degree form IIT Khargpur and Ph.D. from Thapar Institute of Engineering & Technology, Patiala. She is currently Professor, Computer Science and Engineering and Dean (Student Affairs) at Thapar University, Patiala since September 2010. As Dean (Student Affairs) she has made students excel in diverse skills and areas with determination and conviction. She has demonstrated wonderful managerial skills by heading the department for more than six years. The department has grown in all dimensions including academics, research, man power development and finances. Her areas of research interests include Parallel, Distributed Grid and Cloud Computing, VLSI Testing, Energy aware computing and Cultural Computing. Dr. Bawa has rich teaching, research and industry experience. She has worked as Software Engineer, Project Leader and Project Manager, in software industry for more than five years before joining Thapar University. She has been Coordinator of two national level research & development projects sponsored by Ministry of Information and Communication Technology. She is the author/co-author 111 research publications in technical journals and conferences of international repute. She has served as Advisor / Track chair for various national and international conferences. She has supervised eight Ph.D. and forty four M.E theses so far. Prof. Bawa is an active member of IEEE, ACM, Computer Society of India, and VLSI Society of India. She has been rendering her services across the globe as an editor and reviewer of various reputed journals of these societies.

Dr. Ankur Gupta is the Joint Director at the Model Institute of Engineering and Technology, Jammu, India, besides being a Professor in the Department of Computer Science and Engineering. Prior to joining academia, he worked as a Technical Team Lead at Hewlett Packard, developing software in the network management and e-Commerce domains. He obtained B.E (Hons) Computer Science and MS Software Systems degrees from BITS, Pilani and his PhD from the National Institute of Technology in India. His main areas of interest include peer-to-peer networks, network management, software engineering and cloud computing. He has published over 40 peer-reviewed papers in reputed international journals and conferences and is a recipient of the AICTE's (All India Council for Technical Education) Career Award. He has filed 10 patents in diverse technical domains and is the founding managing editor of the International Journal of Next-Generation Computing (IJNGC). He is a senior member of both the IEEE and ACM and a life member of the Computer Society of India.