Intercloud Message Exchange Middleware

Muhammad Bilal Amin
Department of Computer Engineering,
Kyung Hee University,
Korea
mbilalamin@oslab.khu.ac.kr

Wajahat Ali Khan
Department of Computer Engineering,
Kyung Hee University,
Korea
wajahat.alikhan@oslab.khu.ac.kr

Ammar Ahmad Awan
Department of Computer Engineering,
Kyung Hee University,
Korea
ammar@oslab.khu.ac.kr

Sungyoung Lee
Department of Computer Engineering,
Kyung Hee University,
Korea
sylee@oslab.khu.ac.kr

ABSTRACT
Cloud Interoperability has been a core issue pertaining Intercloud and Cloud Federation. Several vendor-based proprietary solutions and open-source middleware are present for the resolution; however, these solutions are highly coupled to particular cloud environments. For heterogeneous clouds to exist in an interoperable environment, the need of a vendor-independent, secure and reliable message exchange middleware is critical. In this paper, considering general cloud architecture, we are presenting a Publish-Subscribe based middleware for Intercloud Message Exchange. Intercloud Message Exchange is an implementation of Data Distribution Service (DDS). DDS's reliable pub-sub messaging in conjunction with our devised Information Model can be a novel candidate for messaging domain of Intercloud Interoperability Standards. This Information Model also hosts an OWL based Cloud Resource Description Ontology, utilized by cloud environments for resource cataloguing and possible matchmaking prior to workload migration between heterogeneous clouds.

Categories and Subject Descriptors
C.2.4 [Distributed Systems] : Client/server, Distributed applications

General Terms
Design, Experimentation, Standardization

Keywords
Cloud Computing, Interoperability, Intercloud, Cloud Federation, Messaging, Data Distribution Service (DDS), Cloud Resource Description Ontology

1. INTRODUCTION
Cloud computing has matured itself to a point where community has started identifying clouds as single entities rather than a collection of servers or a specialized datacenter. Cloud is a datacenter with specialized properties [1] and it has shown the potential of not only existing as a monolith but to shake-hands with other clouds for mutual benefit and collaborations as claimed in [2].

ICUIMC'12, February 20–22, 2012, Kuala Lumpur, Malaysia.

Cloud Computing definitions, service, and deployment models are well established. With their on-going implementations, research community and industry, understands their need of better and collaborative utilization. Subsequently, Hybrid Cloud Models, Intercloud [3], Cross-Cloud [4], and Cloud Federation have emerged as essential Topics of concern. Classification of applications and data as public and private, hosting them in their respective clouds with ubiquitous access via services is how we implement a Hybrid Cloud infrastructure.

Hybrid Cloud balances the trade-off between availability and privacy, which has been its key selling point. To implement Hybrid Clouds, proprietary solutions and standards are used for portability and interoperability. However, Intercloud and Cloud Federation, are generalization of two or more clouds collaboration for a mutual benefit model. While Cloud Federation is more focused towards vendor-centric solutions, Intercloud gives us the opportunity for future standards and open interfaces [5]. The required strategy must conquer the heterogeneity issues among various cloud environments by resolving Intercloud Interoperability based on standards, automation, scalability, security, and privacy.

Any two or more clouds can communicate with each other via http/https, like service calls; however, what required is the ability of a cloud to discover an interoperable cloud ready to exchange information over a secure channel for a possible workload scaling or migration keeping the transparency intact. There is a great need of Intercloud Interoperability Standard that systematically identifies the key-areas and provides guidelines for possible implementation. For a resolution, David Bernstein et al. [1] established a key set of domains, which gives researchers an opportunity to provide comprehensive solutions to address these domains individually. In this paper we are proposing solution for "Cloud Presence and Messaging" [1] domain.

For Intercloud Interoperability to execute, the necessity is not just the existence of another interoperable cloud, but their need to exist a protocol that caters the need of reliable Intercloud presence and messaging. Based on the work presented in [1], we are proposing an Intercloud Presence and Message Exchange Middleware (ICME). ICME utilizes a real-time messaging service particularly build for distributed applications called Data Distribution Service (DDS). DDS is an Object Modeling Group's (OMG) standard with several successful implementations and deployments in the field of real-time distributed applications. DDS being a well-established middleware, this paper approaches from a feasibility aspect of its implementation for Intercloud presence and message exchange. In contrast with Point-to-Point manner of Intercloud communication where each cloud provider carries a direct reference to another cloud provider which results in the n^2 complexity problem, ICME's inherited pub-sub nature from DDS will have the constant complexity for sending and receiving messages.

In addition to Intercloud presence and messaging, there is indeed a requirement for a formal definition of cloud infrastructure and resources. This definition must catalogue the entities of Cloud Computing domain and describe their relationship in a formal hierarchy. This hierarchy is not only beneficial for cloud implementation; however, can also be very effective in Cloud Description, Cloud Computing Requirements and even in Cloud Infrastructure Design for various deployment models. To formalize this hierarchy we are also presenting an OWL-based Cloud Resource Description Ontology that is utilized by ICME for Intercloud resource description and matchmaking for possible workload migration.

Use Cases of Virtual Machine (VM) Migration is discussed in Section VI of this paper. These Use Cases provide an end-to-end flow with identification of processes involved in a VM Migration scenario using ICME. These Use Cases capture utilization and behaviors of ICME during the execution of a likely-to-occur Intercloud task.

This paper is arranged as follows, Section II briefly describes some of the related work in the field of Cloud Interoperability, Intercloud Presence, and Messaging, and Cloud Federation. Section III explains the general terminologies of DDS. Section IV presents our approach by describing the ICME implementation in detail. Section V explains the Cloud Resource Description Ontology. Section VI executes the presented solution in the form of possible Use Cases. Section VII concludes this paper.

2. RELATED WORK

Apparently, several cloud implementations are available across the Cloud Computing industry. Some of them are vendor based proprietary solutions (i.e., Amazon EC2 [6], Microsoft Azure [7], Salesforce [8], Google AppEngine [9]), while few are the middleware implementations provided as open-source (i.e., OpenNebula [10], OpenQRM [11], Eucalyptus [12], Nimbus [13]). Apart from being state-of-the-art they are either vendor-closed or highly-coupled to particular environments solutions. OpenQRM is a closed platform with no federation and interoperability support while OpenNebula, Eucalyptus, and Nimbus are only compatible with Amazon EC2 environment. Libraries like jclouds[14], boto[15] and libclouds[16] are available to provide abstraction over Cloud Interoperability but fail to cater the needs of cataloguing cloud resources in a standardized format which we realizes as an integral part of Cloud Interoperability domain.

In an effort to present a standardized mechanism for message exchange, David Bernstein and Deepak Vij from Huawei Technologies, USA have presented an XMPP and RDF based Intercloud Directory and Message Exchange Protocol [17]. This protocol utilizes the UDDI-based nature of XMPP as a cloud registry module. This extends the ability of Intercloud communication from peer-to-peer to a broker-based 1-many resolution. Being the registry based nature of XMPP; this implementation has reliance on a third party server that manages the directory of all the connected cloud nodes. This solution does decouple the clouds successfully; however, it couples the cloud to a particular registry server.

For Cloud Federation a solution has been presented by Antonio et-al. [18], which explains a three-phase (discovery, matchmaking, and authentication) cross-cloud federation model for resource migration. An XML based solution is explained that utilizes XMPP, XACML, and SAML. Concept of this solution is presented by T. Bittman in [2].

A Multi-Agent Middleware system for isolated control and communications is proposed in [19] which combine DDS and ontology implementation for Intelligent Distributed Systems. Tree based ontology structure is used for defining quality of service policies combined with DDS to extract communication details and identify location of an agent in a certain space. Concept of an intelligent robot has been given as an example implementation. ICME uses similar technologies; however, its application context, implementation goals, approach of solution, and technique itself is focused on Intercloud messaging and cloud resource cataloguing.

Motivation of our solution is to remove the reliance on a third party registry and present a solution towards standardization. As the importance of Intercloud interoperability is fairly understood, we believe the discovery of a cloud by another cloud can be taken care by an Intercloud Domain where the middleware is embedded with the discovery and messaging system between cloud environments. Data Distribution Service by OMG based implementation is our underlying technology that fully supports our earlier stated motive. Our solution also benefits from the idea of RDF based catalogues and has defined a Cloud Resource Description Ontology based on the layered architecture of Cloud Datacenters as defined and explained in [20].

3. DDS TERMINOLOGIES

DDS is an OMG standard for real-time distributed systems. It works on publish-subscribe based data exchange model between entities. DDS was devised in 2003, since than it has been established as mainstream pub-sub technology for high performance distributed systems including Air Traffic Control, High Frequency Trading, and Military & Defense applications. DDS has gone through multiple iterations. The latest update is ver. 1.3 managed and used by OMG members. The details can be viewed in [21].

The key foundation of DDS implementation is Global Data Space (GDS). GDS is a contribution environment of participants. A GDS can be further classified into individual Domains depending on the application. Participants can join the GDS as publisher and/or subscriber as and when needed. The message being published by a participant is propagated across the domain and only the subscribers to the message are able to receive it, decoupling the publishers and subscribers. This type of subscription is defined as a Topic and gets encapsulated with various values in a Sample. In short, Samples are different values of Topics travelling over GDS. A subscriber subscribes to a Topic carried by the Sample and receives only the information from the Topic it subscribed to.

GDS configuration management performs the dynamic discovery of publishers and subscribers. Data flows are automatic and taken care by the DDS layer. For every Topic travelling over a particular domain has QoS definition attached as parameters. These parameters add reliability and customization on Topics and Samples.

DDS promotes the implementation of a Net-Centric Security Model, in which security can be defined like Topics or over a domain. Two possible implementations can be, a “Domain-based Mandatory Access Control” and “Topic-based Role-based Access Control” as defined in [22]. All the participating entities can be authenticated with confidential message communication across a particular domain.

DDS is a highly interoperable middleware. DDS “Interoperability Wire Protocol” guarantees the interoperability among various architectures. Data-Centric Publish Subscriber (DCPS) provides platform interoperability with implementation in various languages and frameworks. RTI’s Implementation of DDS offers C, C++, C# and Java versions of DDS API [23].

The application model of DDS can be compared with JMS and CORBA. However DDS is a non-broker/non-registry based architecture. DDS provides abstraction by user-defined data types through Topics. Unlike JMS and other technologies, DDS enables customization using QoS parameters, which caters the non-functional aspects of Topics also. The higher performance is achieved by very low latency of DDS i.e. million updates a sec. Detail Architecture of DDS is defined in [24].

4. INTERCLOUD MESSAGE EXCHANGE (ICME) FOR INTERCLOUD INTEROPERABILITY

This section describes the conceptual architecture for ICME over a cloud participating node, shown in Fig 1. Entity Definition Layer defines the cloud-oriented entities, instantiated by Entity Factory for a particular subscription or publication. Core components of ICME architecture are explained below in detail.

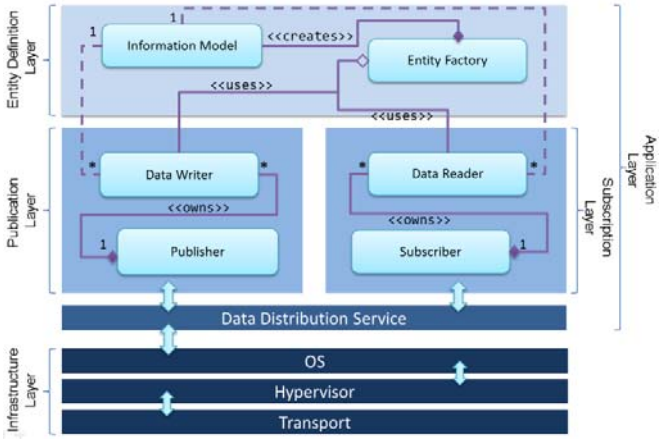


Figure 1. **Conceptual Architecture of ICME leveraging DDS**

4.1. Information Model

In any modern day software application, foundation lies in its object model. Objects as Entities, Controllers, and Interfaces build the structure of scalable and reliable software architecture. Important part is the discovery of these objects and their relationships among each other. ICME's architecture is not unlike. Apart from the publication and subscription, the conceptual objects in context with application, need to be defined. Information Model as shown in Fig 2 defines this abstraction. All the participants of this Information Model are called the *Entities*.

Message encapsulates request for discovery of interoperable cloud environment and response from friendly clouds within Cloud Domain. *Request* and *Response* are not to be confused with conventional request and response communication. *Request* and *Response* entities are both publications with-in the Cloud Domain. *Concrete Request* and *Concrete Response* are specializations of *Request* and *Response*.

Message Type enumeration can be used to classify a message further. For example, a resource migration request (MSG_RMR) or a previously sent message disposal request (MSG_DSP) which means the previously sent request is invalid and any processing in this regard needs to be stopped or cancelled.

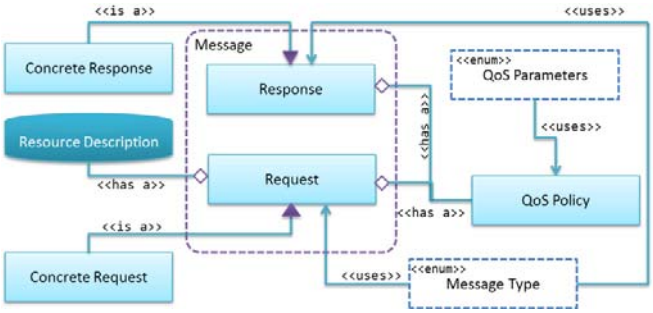


Figure 2. **ICME Information Model**

QoS Parameters enumeration provides the quality of service parameters defined by DDS. QoS Policy defines the collection of QoS Parameters as per publication or subscription. QoS Parameters enumeration can contain all the provided QoS Parameters by DDS or a specialized list, specific to the domain. For Example, message expiration time (Deadline). These conditions may exist with the lifecycle of *Request* and *Response*.

Resource Description is a container for formal cloud resource description i.e., Ontology-based specification of a cloud in Web Ontology Language (OWL) format. Section V describes cloud resource description ontology in detail.

4.2. Domain & Domain Participants

Every cloud has a participating root node that registers the cloud in GDS. Every root has the ability to run DDS for multipurpose applications. Domains classify these applications. There exist a many-to-one relationship between an application and a domain i.e., an application can belong to multiple domains; however DataReaders and DataWriters only belong to the domain in which they are created. DataReader and DataWriter belonging to different domains will never exchange data even being on the same machine. To implement Intercloud Interoperability based services, isolated application (ICME) is created and deployed under GDS (Cloud Domain) represented in Fig 3.

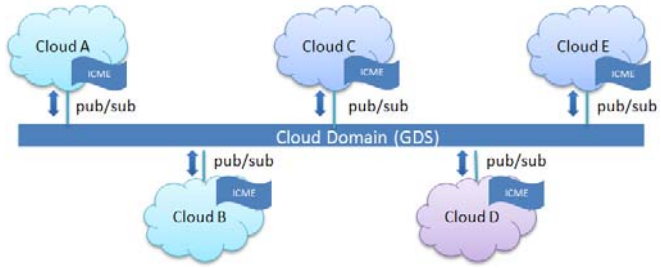


Figure 3. **Cloud Domain over Global Data Space (GDS)**

A Domain Index (Integer Value) identifies a particular domain. ICME deployed on all the root nodes creates DomainParticipants with the same domain index belonging to the Cloud Domain. Each DomainParticipant is implicitly the root node of every cloud with ICME deployment (Fig 1).

4.3. Samples and Topics

A Topic is the realization of a data type in ICME and the unit of data travelling over the Cloud Domain. In object oriented terms, the instantiation of request and response is called Topic-Instance, where request and response are the user-defined data-types (Entities). Multiple instances of a Topic can exist in an ICME environment, whereas, each instance can have mutable values except the key-value. A distinct set of value for a particular Topic-Instance is called a Sample. This enables ICME to carry multiple samples for a particular Topic-Instance. Following listings elaborate these concepts (Note: These code snippets are for explanation only, they do not follow any particular programing language syntax or any programming paradigm).

```
struct Request
{
        Time timestamp;
        string topicId; // Key-Value
        string OS;
        int RAM;
}
```

Listing 1.

In Listing 1, *Request* is a user-defined data type. String topicId is the key-value that uniquely identifies an instance. Every root node in Cloud Domain via ICME will subscribe all the Topics and Samples created for Request.

```
request_1 = (Topic: "Request")
                                + (Key: "VM1")
request_1.timeStamp = Time.Now
request_1.OS = "Win2K3SP2SRV"
request_1.RAM = 2048

request_2 = (Topic: "Request")
                                + (Key: "VM2")
request_2.timeStamp = Time.Now
request_2.OS = "WinXPSP3Pro"
request_2.RAM = 1024
```

Listing 2.

Listing 2 creates request_1 and request_2 as two Topic-Instances of type Request with distinct key-values, VM1 & VM2 respectively. Currently, this defines a single Sample per Topic-Instance. However multiple Samples per Topic Instance can be created (Listing 3).

```
request_1.timeStamp = Time.Now
request_1.OS = "Win2K3SP2SRV"
request_1.RAM = 4096
```

Listing 3.

Listing 3 appends to Listing 2, creating second Sample for Topic-Instance request_1. The key-value "VM1" remains unchanged, however the value for timestamp has changed during the execution flow i.e. the timestamp value for first Sample of request_1 is different from the second Sample of the same Topic-Instance. The value for RAM has also changed from 2048 to 4096.

These Listings are related to VM Migration between interoperable clouds using ICME. All the root nodes subscribe to Request, and get values for all the Topic-Instances and samples pertaining to Request. In addition, root nodes do not have to subscribe explicitly for a particular Sample or Topic-Instance i.e., a VM configuration in this case. As new configurations become available, all the root nodes will immediately start receiving requests for those VMs as well.

4.4. Quality of Service (QoS)

One of the novelties of DDS is the control over publications and subscriptions via Quality of Service parameters. These parameters add up to become a QoS Policy. QoS adds agility in communication between publishers and subscribers for a fairly simple to a very complex requirement. DDS provides a set of QoS parameters that can be utilized as per requirement. This innovation is fully utilized by ICME's Information Model. For our scope i.e., Cloud Domain, we can enlist the parameters fit for our needs. These parameters are defined in QoS Parameter's enumeration in our information model (Fig 2).

QoS Policy must be defined by mutual agreement between publishers and subscribers in the form of a contract. For two root nodes to communicate, their QoS Policy must be compatible. In case of incompatibility, DDS will flag the ICME DataReader and DataWriter of incompatible root nodes and mutual communication will not occur. Table 1, describes the candidate QoS Parameters in accordance with ICME's requirements. (Note: a "message" can be a request or a response type Topic)

Table 1 (QoS Policy Parameters).

Deadline	• For DataReader, Maximum expected elapsed time between the arrivals of message samples. • For DataWriter, a commitment to publish message samples with no greater elapsed time.
Durability	• Specifies whether to re-publish the message samples to other or new DataReaders.
LatencyBudget	• Allowed time to deliver a message.
Lifespan	• Duration while the message is valid.
Reliability	• Whether or not message needs to be delivered using a reliable mechanism.
Liveliness	• Mechanism to detect whether any root node is alive or dead.
TransportPriority	• Message classification by priority for DataWriter.

A QoS Policy is created prior to the creation of DataWriter and DataReader. Default QoS parameters can be also be tweaked and updated as shown in the Listing 4.

```
DataWriterQoS qos
publisher.getDefaultDataWriterQoS(default)
default.reliability.kind =
                    QoSParameters.Reliability
publisher.setDefaultDataWriterQos(default)
```

Listing 4.

4.5. Publication & Subscription

To publish and subscribe Topics at root nodes, DataReader and DataWriter of DDS are used via API. DataReader object is created when ICME has a Topic to write as publication; correspondingly DataReader object is created when ICME wants to receive values for a Topic.

In case of VM Migration, ICME uses DataWriter object to send request Topic. A DataWriter is associated with single request; however, multiple DataWriters and Topics in ICME can exist depending on the entity definition in Information Model. As mentioned earlier ICME uses DataWriter via DDS API i.e., a Publisher object. When ICME calls a `write()` on DataWriter, request object is passed to the publisher object, which does the actual serialization of data over Cloud Domain. Listing 5 shows the ICME write via DDS publisher object.

```
DataWriter ICMEWriter =
    publisher.createDataWriter(request, qos)
ICMEWriter.write()
```

Listing 5.

Similarly, ICME uses DataReader object to receive request Topic. A DataReader is associated with single request; however, multiple DataReaders and Topics in ICME can exist depending on the entity definition in Information Model. After receiving data, it is first processed (de-serialization) by a Subscriber object of the root node. The data sample is then stored in the appropriate DataReader. ICME can read the request either by registering to a listener or by calling `read()` and `take()` in frequent intervals. Listing 6 describes the ICME read via DDS subscriber.

```
DataReader ICMEReader =
   subscriber.createDataReader(request, qos,
                                    listener)
ICMEReader.enable()
```

Listing 6.

Calling `enable()` on a reader will change the request object to an operational state i.e., its Topic will be read by ICME. This eventually

binds a DataReader with the request Topic. A 'disabled state' can be called if the ICME is not ready to accept data i.e., root node is not accepting any VM migration requests or its resources aren't adequate or alive. (Note: There is no "disable" operation, the request can only be disabled by subscriber's QoS properties). In case of acknowledgment, a response object is created and published in similar fashion.

5. CLOUD RESOURCE DESRIPTION ONTOLOGY

As mentioned earlier, the need for a formal cloud resource description is eminent for Intercloud Interoperability. This formal description defines the resources and their relationship in the form of schema (OWL format), named as Cloud Resource Description Ontology. This ontology will behave as a resource catalogue being shared among cloud root nodes. Being part of the Information Model this catalogue enables request to be utilized for resource matching between different cloud environments. Depending on the resource match a cloud becomes interoperable to the other(s).

Similar solution for resource cataloguing is presented in [13]; however, its existence is entirely focused on UDDI based RDF framework. ICME instead, proposes the mechanism of Ontology-Conformance over GDS i.e., the Cloud Domain and redefines the ontology in a comprehensive fashion keeping the cloud service and deployment models in perspective [14].

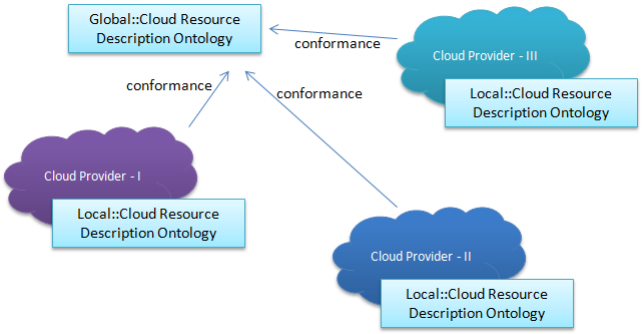


Figure 4. **Ontology Conformance**

Shown in the Fig 4, two types of ontologies participate in ICME implementation. Global Cloud Resource Description Ontology describes all the concepts existing in Cloud Domain. Local Cloud Resource Description Ontology is specific to every cloud deployment. Local ontology only covers the concepts local to a particular cloud vendor. Local ontologies conform to the global ontology owned by the Cloud Domain.

Each Cloud conforms to the global ontology on the basis of their needs leading to different local ontologies. Cloud Resource Description Ontology behaves as a catalyst between cloud environments through their local ontologies resolving interoperability issue.

5.1.Classes

The Cloud_Node class of the ontology contains the four subclasses that define the resources to be used by the cloud environment. The subclasses of the Cloud_Node of the ontology are shown in cloud resource description ontology diagram (Fig 5) and explained as following:

Kernel contains the information about the Kernel of the Cloud Node. The information about this resource is contained in further level of subclasses as Hypervisor, Operating System, VM Monitor, and Clustering Middleware.

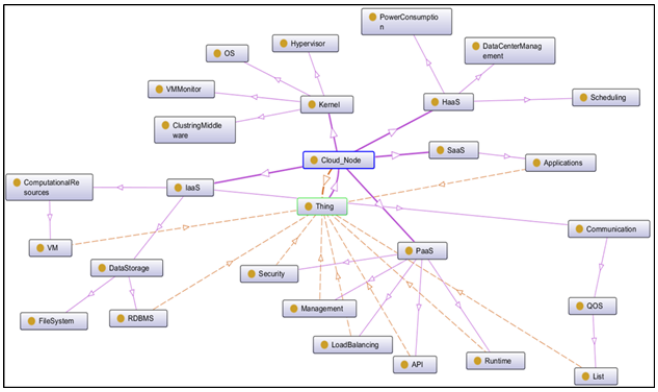


Figure 5. **Cloud Resource Description Ontology**

IaaS contains information of Cloud_Node related to infrastructure as a service. Its subclasses store information regarding Computational Resources (includes Virtual Machines), Communication (includes QoS), and Data-store (include File-System and RDBMS).

PaaS contains information related to the platform used by the Cloud_Node is contained in *PaaS* subclass. *PaaS* is further subdivided in to Security, Management, Load Balancing, API, and Runtime subclasses.

SaaS contains the information on cloud related to software as a service for the cloud node is contained in *SaaS* subclass. This subclass contains another class called Application.

Table 2 (Domain & Range of Cloud Resource Description Ontology).

S.No	Data-type Property	Domain	Range
1.	vendor	Hypervisor, OS	string
2.	info	ClusteringMiddleware,DataCenterManagement, FileSystem,LoadBalancing, Management, RDBMS, Scheduling,Security, PowerConsumption, VMMonitor	string
3.	name	API, Application, FileSystem, Hypervisor, List, Management, OS, RDBMS, Runtime, VM	string
4.	capacity	VM	string
5.	id	Cloud_Node	string
6.	os	VM	string
7.	ram	VM	string
8.	version	API, Hypervisor, Management, OS, Runtime	string

HaaS contains the hardware related information about the cloud node is stored in *HaaS* subclass. It is further divided into Power Consumption, Data Center Management, and Scheduling.

5.2. Properties

By defining property, we can restrict a relation by specifying its Domain and Range. Data type properties of the cloud resource description ontology, with domain, range and restriction are specified in table 2 and 3.

Table 3 (Properties of Cloud Resource Description Ontology).

Class	Property	Restriction
Cloud_Node	contains	≥ contains min 1
API, LoadBalancing, Management, Runtime, Security	contains	≥ contains min 0

6. USE CASES

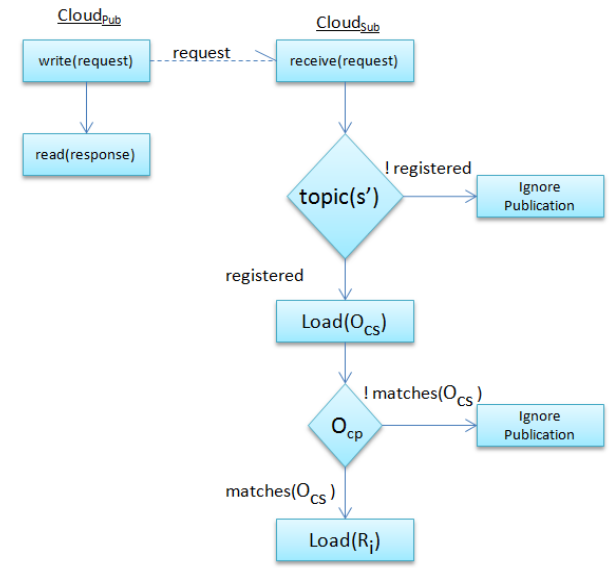
6.1. Use Case – I: A Cloud root node publishes a request for VM migration over Cloud Domain

Publishing cloud initiates the process by creating Sample(s) of request Topic. Local Cloud Resource Description Ontology is initialized with the values specific to the migrating VM requirements (OS, RAM, STORAGE). Ontology is created in OWL format and becomes part of the publication. DataWriter publishes the request with a unique request Id as asynchronous message over Cloud Domain. Subsequently, publishing cloud creates a response with the same request Id and registers the response to a listener for an expected read. Request Id act as a unique identifier, on which response publications can correspond to. Read's lifecycle depends upon the "Lifespan" parameter of QoS Policy defined while creating response Topic.

Publication is received by all the ICME implementations at participating root nodes over Cloud Domain. However, it is only processed by the ones registered for the request Topic. In case of registered subscription, appropriate DataReader stores and process the request sample. Local cloud ontology is loaded and matched with ontology of publishing cloud for further evaluation.

Ontology matching techniques and algorithms can be custom defined and tweaked for Cloud Domain. These techniques can be defined by mutual agreement among participating clouds as well as local to a cloud environment in accordance with cloud governance policies.

Consequently, ICME of the subscribing node loads local resource pool information for resource availability. Fig 6 describes this Use Case as activity diagram.



$$request = s' + O_{cp}$$

$$S' = \sum_{i=1}^{n} S_i$$

$S = \{sample_1, sample_2, \ldots sample_n\}$

O_{cp} = *Cloud Infrastructure Ontology (Pub)*

O_{cs} = *Cloud Infrastructure Ontology (Sub)*

R_i = *Resource Pool Information*

Figure 6. **Use Case-I Activity Diagram**

6.2. Use Case – II: A cloud root node publishes a response of acknowledgment for VM migration over Cloud Domain

In case of resource availability from the resource pool, ICME checks whether the request's Lifespan is still valid. If valid, DataWriter creates response with the request's Id. Afterward, this response is written as a publication over Cloud Domain. The entire participating root nodes receive this publication; however it's only processed by the root node with the request Id registered at the listener for a possible read. This unique identifier (request Id) guarantees the mapping of incoming responses to the original request.

After receiving the acknowledgment, response(s) is read by DataReader and processed. As there can be more than one response for a certain request, bringing forward multiple candidate clouds ready for accepting a VM. A certain evaluation mechanism needs to be in place to evaluate these responses. These mechanisms can vary from first-come-first-serve model to a highly technical evaluation model with intrinsic details. These mechanisms can be implemented over Cloud Domain with mutual agreement or even local to a certain cloud implementation in accordance with its cloud governance policies.

After the selection of the target cloud environment, Single Sign-On is initiated for possible handshake and VM migration. Fig 7 describes this Use Case as activity diagram.

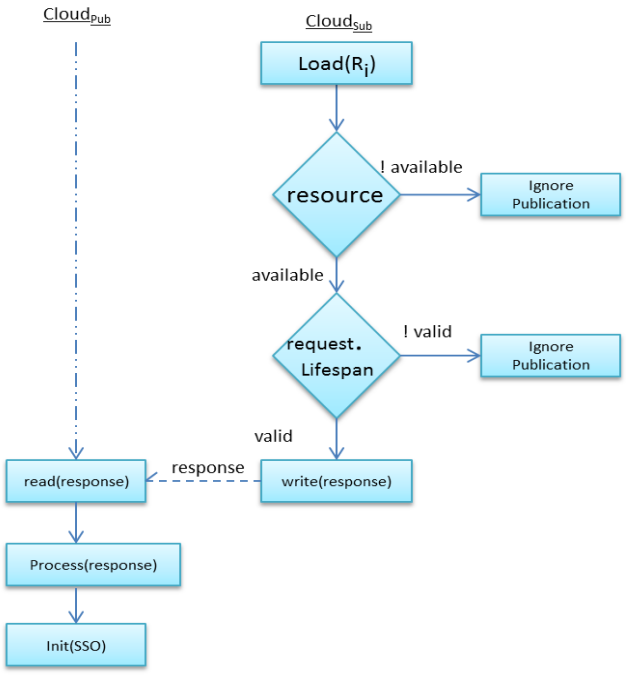


Figure 7. **Use Case-II Activity Diagram**

7. FUTURE WORK & CONCLUSION

ICME is a part of our larger and comprehensive effort to solve Intercloud interoperability issues among various cloud environments. Nevertheless, It is detailed enough to cater the needs of pub-sub based decoupled messaging among cloud nodes for workload sharing. ICME is one of the implementations DDS can provide in the field of Intercloud. The novelties like domain-based security, Topic-based subscription, quality-of-service based messaging, cross-platform deployment and a reliable pub-sub messaging model can be utilized very efficiently for the interoperability needs among heterogeneous clouds running proprietary or open-source middleware solutions. In

future we are planning on extending this implementation to cater the needs of efficient resource migration in real-time and a comprehensive security platform for Intercloud access control.

8. ACKNOWLEDGMENT

This research was fully supported by Microsoft Research Asia.

9. REFERENCES

[1] David Bernstein, Erik Ludvigson, Krishna Sankar, Steve Diamond and Monique Marrow, "Blueprint for the Intercloud – Protocols and Formats for Cloud Computing Interoperability," 2009 Fourth International Conference on Internet and Web Applications and Services.

[2] Thomas J. Bittman, " The Evolution of the Cloud Computing Market," Gartner Research Blog Network, http://blogs.gartner.com/thomas_bittman/2008/11/03/the-evolution-of-the-cloud-computing-market/, November 2008.

[3] Sun Microsystems, "Take your business to a Higher Level- Sun cloud computing technology scales your infrastructure to take advantage of new business opportunities," Guide April 2009.

[4] W. Li and L. Ping, "Trust model to enhance security and interoperability of cloud environment," Cloud Computing November 2009.

[5] Enterprise Cloud Computing Blog, http://www.cloudswitch.com/page/cloud-federation-and-the-intercloud

[6] Amazon Elastic Compute Cloud (EC2), http://aws.amazon.com/ec2/

[7] Microsoft Azure, http://www.microsoft.com/windowsazure/

[8] Salesforce, www.salesforce.com/cloudcomputing/

[9] Google AppEngine, http://code.google.com/appengine/

[10] OpenNebula, http://opennebula.org/

[11] OpenQRM,http:// www.openqrm.com/

[12] Euclyptus, http://www.eucalyptus.com/

[13] Nimbus, http:// www.nimbusproject.org/

[14] jclouds, http://code.google.com/p/jclouds/

[15] boto, http://code.google.com/p/boto/

[16] Apache libcloud, http://libcloud.apache.org/

[17] David Bernstein and Deepak Vij, "Intercloud Directory and Exchange Protocol Detail using XMPP and RDF," 2010 IEEE 6th World Congress on Services

[18] Antonio Celesti, Franco Tusa, Massimo Villari and Antonio Puliafito, "How to Enhance Cloud Architectures to Enable Cross-Federation," 2010 IEEE 3rd International Conference on Cloud Computing.

[19] Jose L Poza, Juan L. Posadas and Jose E. Simo, "Adding an Ontology to a Standardized QoS-Based MAS Middleware," DISTRIBUTED COMPUTING, ARTIFICIAL INTELLIGENCE, BIOINFORMATICS, SOFT COMPUTING, AND AMBIENT ASSISTED LIVING. Lecture Notes in Computer Science, 2009, Volume 5518/2009, 83-90.

[20] Youseff, L. and Butrice, M. and Da Silve, D., "Towards a Unified Ontology for Cloud Computing," Grid Computing Environments Workshop, 2008.

[21] Data Dstribution Service Specification, http://www.omg.org/technology/documents/dds_spec_catalog.htm

[22] Gerardo Pardo-Castellote, "Secure DDS, A security model sutable for Net-Centric, Publish- Subscribe and Data Distributed Systems," RTESS, Washington DC, July 2007.

[23] RTI Data Distribution Service, User's Manual. Real-Time Innovations, Inc, June 2010.

[24] J. M. Schlesselman, Gerardo Pardo-Castellote and Bert Farabaugh, "OMG Data-Distribution Service (DDS): Architectural Update," MILCOM 2004.