Blueprint for the Intercloud – Protocols and Formats for Cloud Computing Interoperability

David Bernstein Erik Ludvigson Krishna Sankar Steve Diamond Monique Morrow
Cisco Systems, Inc.
[daberns, eludvigs, ksankar, stdiamon, mmorrow]@cisco.com

Abstract

Cloud Computing is a term applied to large, hosted datacenters, usually geographically distributed, which offer various computational services on a "utility" basis. Most typically the configuration and provisioning of these datacenters, as far as the services for the subscribers go, is highly automated, to the point of the service being delivered within seconds of the subscriber request. Additionally, the datacenters typically use hypervisor based virtualization as a technique to deliver these services. The concept of a cloud operated by one service provider or enterprise interoperating with a clouds operated by another is a powerful idea. So far that is limited to use cases where code running on one cloud explicitly references a service on another cloud. There is no implicit and transparent interoperability. Use cases for interoperability, as well as work-in-progress around inter-cloud protocols and formats for enabling those use cases, are discussed in this paper.

1. Introduction

Cloud Computing has emerged recently as a label for a particular type of datacenter. A cloud may be hosted by anyone; an enterprise, a service provider, or a government.

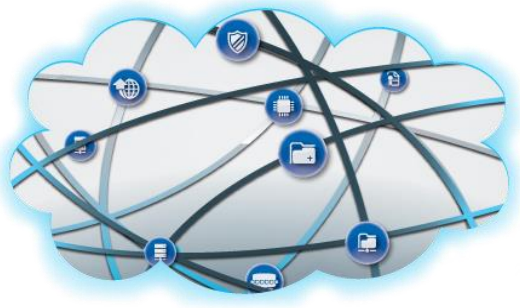

Figure 1. A Cloud is just a special kind of datacenter. We list seven key characteristics which make a large datacenter into a cloud.

For the purposes of this paper, we define Cloud Computing as a datacenter which:

1. Implements a pool of computing resources and services which are shared amongst subscribers.
2. Charges for resources and services using an "as used" metered and/or capacity based model.
3. Are usually geographically distributed, in a manner which is transparent to the subscriber (unless they explicitly ask for visibility of that).

Figure 2. Clouds may be in one geography or may spread over several geographies

4. Are automated in that the provisioning and configuration (and de-configuration and un-provisioning) of resources and services occur on the "self service", usually programmatic request of the subscriber, occur in an automated way with no human operator assistance, and are delivered in one or two orders of seconds.
5. Resources and services are delivered virtually, that is, although they may appear to be physical (servers, disks, network segments, etc) they are actually

virtual implementations of those on an underlying physical infrastructure which the subscriber never sees.

6. The physical infrastructure changes rarely. The virtually delivered resources and services are changing constantly.
7. Resources and services may be of a physical metaphor (servers, disks, network segments, etc) or they may be of an abstract metaphor (blob storage functions, message queue functions, email functions, multicast functions, etc). These may be intermixed.

Cloud Computing services as defined above are best exemplified by the Amazon Web Services (AWS) [1][2] or Google AppEngine [3][4]. Both of these systems exhibit all eight characteristics as detailed above. Various companies are beginning to offer similar services, such as the Microsoft Azure Service [5], and software companies such as VMware [6] and open source projects such as UCSB Eucalyptus [7][8] are creating software for building a cloud service. Each of these offerings embody Cloud Computing with a self-contained set of conventions, file formats, and programmer interfaces. If one wants to utilize that variation of cloud, one must create configurations and code specific to that cloud.

Of course from within one cloud, explicit instructions can be issued over the Internet to another cloud. For example, code executing within Google AppEngine can also reference storage residing on AWS. However there are no implicit ways that clouds resources and services can be exported or caused to interoperate.

Our work in progress examines this challenge. First we present some simple use cases and highlight the roadblocks for making that use case work. Next we detail the areas of protocols and formats which need to be developed and standardized to solve not only the specific use cases but we believe many additional use cases. Finally within each area of protocol and format, we detail specific technologies which we believe show promise as a solution or a basis for a solution and speak to the how that technology is applies to cloud computing interoperability.

We are calling this profile of protocols and formats for Cloud Computing interoperability the "Intercloud Protocols".

We consider code portability and common APIs amongst clouds as important but of a different scope, and not addressed in this paper.

2. Use Cases

We outline use cases which cover the two basic natures of Cloud Computing delivered resources and services; that is a use case involving a physical metaphor (servers, disks, network segments, etc) and a use case involving an abstract metaphor (blob storage functions, message queue, email functions, multicast functions, etc).

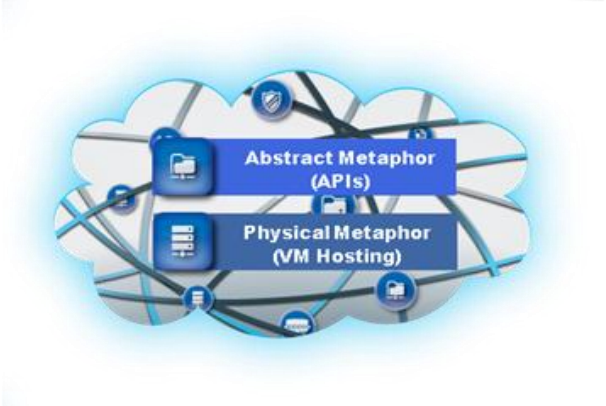

Figure 3. We look at cloud interoperability challenges using use cases illustrating the two major personality types of clouds

2.1. Virtual Machine Instantiation and Mobility

One of the most basic resources which Cloud Computing delivers is the Virtual Machine, which is a physical metaphor type of resource. One way or another, a subscriber requests the provisioning of a particularly configured virtual machine with certain quantities of resources such as memory processor speeds and quantities.. The format of this request varies widely by Cloud Computing platform and also is somewhat specific to the type of hypervisor (the virtualization layer of the operating system inside the Cloud Computing platform). In a few seconds they receive pointers and credentials with which to access it. The pointers are usually the MAC and IP addresses [9] and sometimes a DNS name [10] given to the VM. The credentials are usually a pair of RSA keys [11] (a public key and a private key, which one uses in the API to speak with the VM). Most often, the VM presents an x86 PC machine architecture [12]. On that VM, one boots a system image yielding a running system, and uses it in a similar manner as one would use a running system in your own datacenter.

VM Mobility is that feature in a particular hypervisor which allows a running system to be moved from one VM to another VM. As far as the running system is concerned it does not need to be reconfigured, all of the elements such as MAC and IP address and DNS name stay the same; any of the ways storage may be referenced (such as a World Wide Name in a SAN [13]) stay the same. Whatever needs to happen to make this work is not the concern of the running system.

VM Mobility has been implemented with several hypervisors but there are limitations. Usually these limitations are a result of the "scope" of applicability of the network and storage addressing. Typically, VM Mobility is restricted to a Layer 3 subnet [14] and a Layer 2 domain (for VLANs) [15] because the underlying network will support the VM operating outside of the local scope of those addresses. Needless to say, the network addressing scheme in a cloud operated by an entirely different service provider is not only a different subnet but a different class B or class A network altogether. Routers and switches simply would not know how to cope with the "rogue" running system.

Another aspect is that, the instantiation instructions of the VM for the running system are very specific to that Cloud Computing platform and the hypervisor which it uses. We would want to re-issue some of these instructions to the new Cloud so that the VM it delivered onto which the VM would move, was as suitable as the first VM which was provisioned for us. If the new Cloud takes an entirely different set of instructions, this is another barrier to VM Mobility.

All of this assumed that in the universe of Cloud Computing systems out there, I was able to find another cloud, which was ready, willing, and able to accept a VM mobility transaction with me. And that I was able to have a reliable conversation with that cloud, perhaps exchanging whatever subscription or usage related information which might have been needed as a pre-cursor to the transaction, and finally that I had a reliable transport on which to move the VM itself.

2.2. Storage Interoperability and Federation

No let us consider an interoperability use case involving an abstract metaphor. In this case, I am running script or code in my datacenter or in the cloud, which is utilizing Cloud based storage functions. In Cloud Computing, storage is not like disk access, there are several parameters around the storage which are inherent to the system, and one decides if they meet your needs or not For example, object storage is typically replicated to several places in the cloud, In AWS and in Azure it is replicated three places. The storage API is such that, a *write* will return as successful when one replicate of the storage has been effected, and then a "lazy" internal algorithm is used to replicate the object to two additional places. If one or two of the object replicates are lost the cloud platform will replicate it to another place or two such that it is now in three places. A user has some control over where the storage is, physically, for example, one can restrict the storage to replicate entirely in North America or in Europe. There is no ability to vary from these parameters; that is what the storage system provides. We do envision other providers implementing say, five replicates, or a deterministic replication algorithm, or a replicated (DR) *write* which doesn't return until and unless *n* replicates are persisted. One can create a large number of variations around "quality of storage" for Cloud.

In the interoperability scenario, suppose AWS is running short of storage, or wants to provide a geographic storage location for an AWS customer, where AWS does not have a datacenter, it would be sub-contracting the storage to another service provider. In either of these scenarios, AWS would need to find another cloud, which was ready, willing, and able to accept a storage subcontracting transaction with them. AWS would have to be able to have a reliable conversation with that cloud, again exchanging whatever subscription or usage related information which might have been needed as a pre-cursor to the transaction, and finally have a reliable transport on which to move the storage itself. Note, the S3 storage API is not guaranteed to succeed, if there is a failed *write* operation from AWS to a subscriber request, the subscriber code is supposed to deal with that (perhaps, via an application code level retry). However Cloud to Cloud, a target cloud *write* failing is not something the subscriber code can take care of. That needs to be reliable.

Although the addressing issues are not as severe in this case where an abstract metaphor is used, the naming, discovery, conversation setup items challenges all remain.

3. Intercloud Protocols Profile

To address interoperability use cases such as these, certain commonalities amongst clouds must be adopted. With the Internet, interoperability foundations were set with the basics of IP addressing, DNS, exchange and routing protocols such as BGP [16], OSPF [17], and peering conventions using AS [18] numbering. Clearly, analogous areas in Cloud Computing need to investigated and similar technologies, but for computing, need to be invented.

Our research involves a lab where we have constructed clouds of primarily two kinds, one using hypervisors from VMware and the associated tooling and conventions that are associated with that set of products, and another using open source hypervisors such as Xen and KVM from RedHat, and the associated tooling and conventions (Linux, and AWS-like) that are associated with that set of products. We are investigating and prototyping protocols, and formats, and common mechanisms, which implement cloud interoperability, or for brevity, Intercloud.

We call the protocols and formats, collectively, "Intercloud Protocols". We call the common mechanisms, collectively, an "Intercloud Root". Figure 4 shows the areas that we believe form a relatively complete picture of the domain of cloud standards.

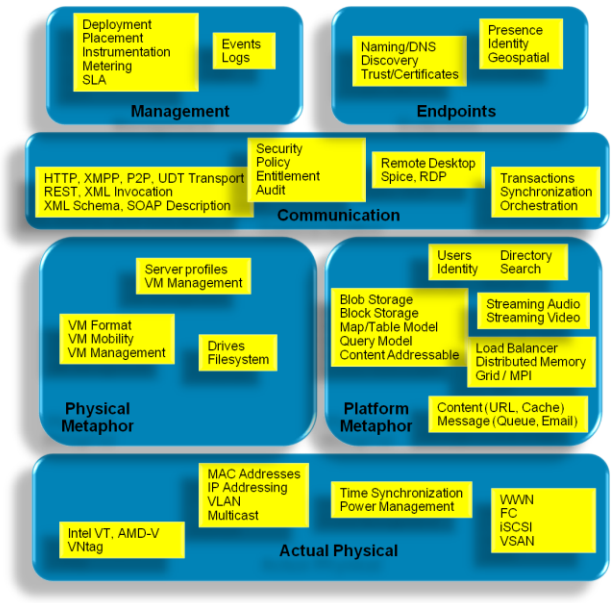

Figure 4. An Architecture for Intercloud Standards

The remainder of the paper describes each area we are investigating. We do not speak to every area, but to the ones needed to address the use cases which were described. First we speak to the nature of the interoperability challenge in each area, then we detail the candidate techniques for Intercloud Protocols and Root which we are looking into. At the end we put them together into a sequence showing how each standard is used to accomplish the interoperability goal of that use case.

3. Addressing

Interestingly, one area which imposes major challenges is network addressing. In a highly virtualized environment, IP address space explodes. Everything has multiple IP addresses; servers have IP addresses for management, for the physical NICs, for all of the virtual machines and the virtual NIC therein, and if any virtual appliances are installed they have multiple IP addresses as well.

Several areas are of concern here, on the one hand, the IPv4 address space simply starts to run out. Consider an environment inside the Cloud which has 1M actual servers. As explained above, assuming a 16 core server, each server could have 32 VM's, and each VM could have a handful of IP addresses associated with it (virtual NICs, etc). That could easily explode to a Cloud with well over 32M IP addresses. Even using Network Address Translation (NAT) [14], the 24-bit Class A reserved Private Network Range provides a total address space of only 16M unique IP addresses!

For this reason many Cloud operators are considering switching to IPv6 which provides for a much larger local address space [19] in the trillions of unique IP addresses. Switching to IPv6 is quite an undertaking, and some believe that switching from one static addressing scheme to another static addressing scheme (eg IPv4 to IPv6) might not be the right approach in a large highly virtualized environment such as Cloud Computing. If one is reconsidering addressing, one should consider the Mobility aspects of VMs in Cloud.

What becomes obvious in this discussion is that some cloud builders will use IPv4, and some will use IPv6. Is there a common IP Mobility scheme between the two?

3.1. IP Mobility

VM Mobility provides for new challenges in any static addressing scheme. When one moves a running VM from one location to another, the IP address goes with the running VM and any application runtimes hosted by the VM. IP addresses (of either traditional type) embody both Location and Identity in the IP address, eg, routers and switches use the form of the IP address not only to identify uniquely the endpoint, but by virtual of decoding the address, infer the Location of the endpoint (and how to reach that endpoint using switching and routing protocols). So while an addressing scheme is being reconsidered, let's consider two schemes which embody Mobility.

Mobile IPv4 [20] and Mobile IPv6 [21][22][23] mechanisms can be used in this case, but they are not interoperable. Because we are trying to solve the problem from one cloud to another, we need a protocol which has a common, interoperable mobility scheme which can be mapped/encapsulated in both IPv4 and IPv6.

3.2. Location Identity Separation Protocol

In an attempt to completely generalize the addressing solution in a way that interoperates with both IPv4 and IPv6, a completely dynamic scheme where Location and Identification have been separated has been developed. This new scheme is called Location Identity Separation Protocol (LISP) [24]. LISP based systems can interwork with both IPv4 and IPv6 based networks, through protocol support on edge routers. However, internal to a Cloud, which may in itself span several geographies, LISP addressing may be used.

The basic idea behind the Loc/ID split is that the current Internet routing and addressing architecture combines two functions: Routing Locators (RLOCs), which describe how a device is attached to the network, and Endpoint Identifiers (EIDs), which define "who" the device is, in a single numbering space, the IP address. Proponents of the Loc/ID split argue that this "overloading" of functions places the constraints on end-system use of addresses that we detailed. Splitting these functions apart by using different numbering spaces for EIDs and RLOCs yields several advantages, including improved scalability of the routing system through greater aggregation of RLOCs. To achieve this aggregation, we must allocate RLOCs in a way that is congruent with the topology of the network. EIDs, on the other hand, are typically allocated along organizational boundaries.

Because the network topology and organizational hierarchies are rarely congruent, it is difficult (if not impossible) to make a single numbering space efficiently serve both purposes without imposing unacceptable constraints (such as requiring renumbering upon provider changes) on the use of that space. LISP, as a specific instance of the Loc/ID split, aims to decouple location and identity. This decoupling will facilitate improved aggregation of the RLOC space, implement persistent identity in the EID space, and hopefully increase the security and efficiency of network mobility.

To this end current experimentation is being done to assess the viability of using this protocol in conjunction with virtualization and in particular with VM Mobility. Of course, if and when LISP becomes a proven solution for the Cloud scenario, it must propagate into many forms of networking equipment which will take some time.

4. Naming, Identity, Trust

Clouds are not endpoints, in the way servers or clients are. They are resources, and as such are typically identified using a URI [25]. However, a simple name lookup allowing one to access a URI over the Internet is not sufficient for Cloud Computing, we would like to for example have assurance that this is indeed the service we think it is, more detail about what service levels, capabilities, and requirements this service may offer, and since we are using something outside of our local trust domain, perhaps have some audit capabilities.

We are looking at something which, like DNS can be part of an Intercloud Root, and can also be part of a Cloud Computing instance. In addition to DNS-like capabilities, we would like a rich capability for expressing names and services, like a directory service such as LDAP [26] or Active Directory [27]. We would also like a system which allows clouds communicating over a non-secure network to prove their identity to one another in a secure manner, such as Kerberos [28]. Also, we would like a system which can supply trusted security certificates, such as the X.509 [29] which provides for a public key infrastructure (PKI) for single sign-on and Privilege Management Infrastructure (PMI). X.509 specifies, amongst other things, standard formats for public key certificates, certificate revocation lists, attribute certificates, and a certification path validation algorithm.

4.1. IPA

We have been investigating using IPA [30]. IPA is an integrated security information management solution combining an open LDAP directory Server, MIT Kerberos and a X.509 Certificate Authority. IPA provides the functions of:

- Identity (machine, user, virtual machines, groups, authentication credentials)
- Policy (configuration settings, access control)
- Audit (events, logs, analysis thereof)

In IPA one user ID is shared between LDAP and Kerberos, and Kerberos gets the benefit of the directory server's multimaster replication. IPA provides an XML over RPC interface to allow for automation and self service with Cloud infrastructure. IPA is a centralized authentication point which tracks what persons or services logged onto what and when.

Services mutually authenticate and encrypt with Kerberos. DNS and Certificate Authority are currently being integrated into IPA.

From our perspective IPA is a leading candidate for this function in Cloud implementations as well as in the Intercloud Root.

5. Presence and Messaging

Part of interoperability is, that cloud instances must be able to dialog with each other. As the use cases explained, one cloud must be able to find another cloud, which for a particular interoperability scenarios, is ready, willing, and able to accept an interoperability transaction with and furthermore, exchanging whatever subscription or usage related information which might have been needed as a pre-cursor to the transaction. Thus, an Intercloud Protocol for presence and messaging needs to exist.

5.1 XMPP

Extensible Messaging and Presence Protocol (XMPP) [31][32] is exactly such a protocol. XMPP is a

set of open XML technologies for presence and real-time communication developed by the Jabber open-source community in 1999, formalized by the IETF in 2002-2004, continuously extended through the standards process of the XMPP Standards Foundation. XMPP supports presence, structured conversation, lightweight middleware, content syndication, and generalized routing of XML data.

We are experimenting with using XMPP as a control plane presence and dialog protocol for Intercloud. XMPP root servers for this purpose, would be in the Intercloud Root.

6. Virtual Machines

Most Cloud Computing implementations have a capability to deliver a Virtual Machine "on demand" to a subscriber, who requests the provisioning of a particularly configured virtual machine with certain quantities of resources. At that point the Virtual Machine is "booted" with an image (or via instructions) to result in a running system.

The metadata which specifies the image or the system is a crucial abstraction which is at the center of VM interoperability, a key feature for Intercloud. One would like to see an open, secure, portable, efficient, and flexible format for the packaging and distribution of one or more virtual machines to this end.

6.1. Virt-Image

One approach to this is called virt-image [33], which relies on an XML descriptor to create virtual machines from virtual machine images. In general, a virtual machine image consists of the XML descriptor (usually in a file image.xml) and a number of files for the virtual machine's disks. The virt-image tool defines a simple XML format which can be used to describe a virtual appliance. It specifies things like minimum recommended RAM and VCPUs, the disks associated with the appliance, and the hypervisor requirements for booting it.

This is quite interesting, however the resultant XML format is describing a specific deployment of a virtual machine on a specific hypervisor. For more general interoperability, we have turned to another proposed standard.

6.2. OVF

Open Virtualization Format (OVF) [34] is a platform independent, efficient, extensible, and open packaging and distribution format for virtual machines. OVF is virtualization platform neutral, while also enabling platform-specific enhancements to be captured. Even though VMware was the original creator of OVF, there is also an Open-source library and tools to support it [35].

There is much work to do in this area. AWS for example, support their own format called an Amazon Machine Image (AMI), for example, and although the Xen community has worked on OVF the KVM community is just starting to. We are encouraged by the possibility of convergence of this space on OVF by the recent open source conversion utilities such as Thincrust virt-convert [36] which are a proof point that VM meta-data for instantiation and for mobility can be solved eventually.

6.3 Lib Virt

Once you package a VM for deployment, you must be able to talk to the VM to control them (for Mobility, for example). Most virtualization systems do not allow for direct communication to the VM, rather, they provide API's to their management toolsets. For example, this is the case with VMware. One speaks through an API to a client side intermediary or to the management tool [37]. In order to complete the interoperability picture with VM manageability, we are impressed with the *libvirt* [38] project. Libvirt supports features such as remote management using encryption and X.509 certificates, remote management authenticating with Kerberos, discovery using DNS, and management of virtual machines, virtual networks and storage.

We believe that this area is ripe for early standardization.

7. Multicast

Although our use cases don't require it, an area of particular interest is where applications running on clouds are rich media enabled, or are collaboration applications. Application enabling large numbers of people to work together and are audio and video enabled are exciting applications for Cloud Computing. Augmentation of social applications such as Facebook and MySpace with rich media, multi-point collaboration is an challenge to the infrastructure which supports them. It is well known that massive scale, real-time, multi-point applications such as those are well served by IP Multicast [39].

7.1. IP Multicast, Interdomain IP Multicast

IP Multicast is a well understood technology. However, most service provider infrastructures do not

currently allow one to transit IP Multicast on their networks, as it is very demanding on their routers. Within a Cloud Computing environment, we see this as a crucial element for Intercloud, in that applications which want to use API's which ultimately will use IP Multicast for implementation must be supported.

More importantly, for these types of applications to work in an Intercloud context, IP Multicast must work in between and amongst clouds. This is requires Interdomain IP Multicast [40][41].

7.2. LISP IP Multicast

Further complicating the matter, if a LISP addressing scheme has been adopted, as discussed above, a LISP-enabled Multicasting architecture would need to be implemented. Cisco has active work in this area [42] which we expect to be extremely important as Intercloud based media rich, collaboration applications are to broadly work.

8. Time Synchronization

Depending on the applications, time synchronization may not be very important. Network Time Protocol (NTP) [43] may be sufficient for Cloud Computing instances in terms of keeping accurate time, and in synchronizing the distributed computing elements in the Cloud accurately. However, our research has shown considerable clock drift in a distributed system and applications which depend on accurate time will not return correct results or in some cases, function incorrectly.

8.1. IEEE1588

Precision time synchronization will likely be an important aspect of Cloud Computing. We have spent considerable time on a precision time capability called IEEE 1588 [44][45][46].

In the context of Cloud Computing there is nothing additional for industry to do here, except perhaps to realize that Intercloud Protocol capability may rely on having precision timing in the Cloud. It is a consideration of ours that the Intercloud Root may be a source for this time synchronization in the IEEE 1588 format as well.

9. Reliable Application Transport

Using XMPP for control plane information is sufficient. However, when services need to move payloads in a transactional manner, like exchanging business records, customer data, critical storage blocks, or anything which requires a reliable, transactional application transport, a different mechanism is required.

Applications needing this functionality have traditionally turned to MQseries from IBM, JMS from BEA Weblogic or other J2EE provider, or the The Information Bus from TIBCO. In the Cloud Computing world, AWS includes a service called SQS. None of the applications message bus technologies interoperate as their on-the-wire formats are all different.

9.1. AMQP based message bus

We have been working on an interoperable message queue standard called Advanced Message Queuing Protocol (AMQP) [47]. AMQP is an open standard application layer protocol for Message Oriented Middleware. The defining features of AMQP are message orientation, queuing, routing (including point-to-point and publish-and-subscribe), reliability and security. AMQP mandates the behaviour of the messaging provider and client to the extent that implementations from different vendors are truly interoperable, in the same way as SMTP, HTTP, FTP, etc. have created interoperable systems. Previous attempts to standardize middleware have happened at the API level (e.g. JMS) and this did not create interoperability. Unlike JMS, which merely defines an API, AMQP is a wire-level protocol.

Reliable messaging at this level is likely a requirement for Intercloud Protocol.

10. Sequencing the Protocols in the Use Cases

Now, let us put together the protocols into implementing the use cases we have chosen. First we will look at the use case of a VM moving from one cloud to another, called Dynamic Workload Migration.

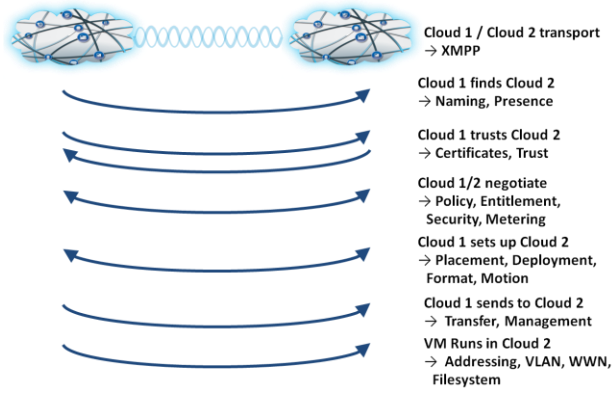

Figure 5. Virtual Machine Mobility and Instantiation

In Figure 5 each of the areas of standards in Figure 4 are referenced. It is easy to see that even with the long list of protocols we have identified not every case has been covered

The next use case covers storage interoperability and federation, which is actually a special case of services interoperability and federation:

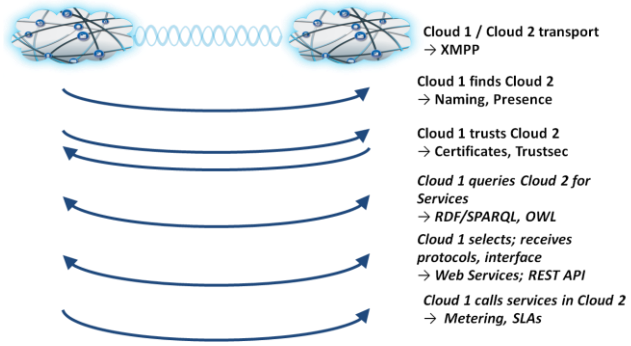

Figure 6. Services Interoperability and Federation

Here we realize that in the conversation between clouds, if one knows the service in question is specific and always the same between clouds, for example storage, that is a simplifying assumption. When one cloud asks to find if the service description on another cloud meets the constraints of the first cloud's interest, it must have a dialog based on a resource description language and a constraints query. As can be seen we are now investigating RDF and OWL [48] for this.

11. Conclusion

More and more service providers are constructing these new, planet-scale virtualized datacenters which are popularly called Cloud Computing. As software and expertise becomes more available, enterprises and smaller service providers are also building Cloud Computing implementations. Active work needs to occur to create interoperability amongst the varied implementations of these Clouds. From the lower level challenges around network addressing, to multicast enablement, to virtual machine mechanics, to the higher level interoperability desires of services, this is an area deserving of much progress and will require the cooperation of several large industry players.

Our initial work shows that, identifying a profile of protocols and formats is one part of the interoperability puzzle. This paper has for the first time enumerated a candidate base set of those and called them collectively "Intercloud Protocols". Our work also shows that, a set of common mechanisms must also be present, both inside the Clouds, and in-between the Clouds. This paper has for the first time enumerated a candidate base set of those and called them collectively "Intercloud Root".

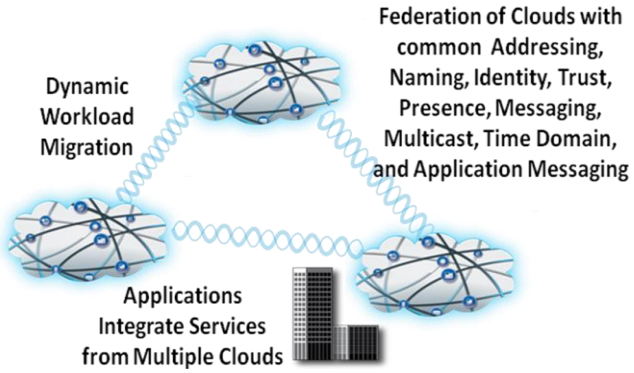

Figure 7. The Intercloud Vision

We will continue our work in showing disparate Cloud Computing instances operating together through Intercloud Protocols using an Intercloud Root mechanism.

In particular, future work will also include a more detail analysis and recommendation around Grid capabilities in Cloud Computing.

12. References

[1] Amazon Web Services at http://aws.amazon.com/

[2] James Murty, *Programming Amazon Web Services; S3, EC2, SQS, FPS, and SimpleDB*, O'Reilly Press, 2008.

[3] Google AppEngine at http://code.google.com/appengine/

[4] Eugene Ciurana, *Developing with Google App Engine*, Firstpress, 2009.

[5] Microsoft Azure, at http://www.microsoft.com/azure/default.mspx

[6] VMware VCloud Initiative at http://www.vmware.com/technology/cloud-computing.html

[7] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, *The Eucalyptus Open-source Cloud-computing System*, Proceedings of Cloud Computing and Its Applications, Chicago, Illinois (October 2008)

[8] Daniel Nurmi, Rich Wolski, Chris Grzegorczyk, Graziano Obertelli, Sunil Soman, Lamia Youseff, Dmitrii Zagorodnov, *Eucalyptus: A Technical Report on an Elastic Utility Computing Architecture Linking Your Programs to Useful Systems*, UCSB Computer Science Technical Report Number 2008-10 (August 2008)

[9] IEEE 802-2001 *Standard for Local and Metropolitan Area Networks: Overview and Architecture*, at http://standards.ieee.org/getieee802/802.html

[10] *Domain Names – Concepts and Facilities*, and related other RFCs, at http://www.ietf.org/rfc/rfc1034.txt

[11] Rivest, R.; A. Shamir; L. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communications of the ACM 21 (2) 1978: pp.120–126, http://theory.lcs.mit.edu/~rivest/rsapaper.pdf
[12] *Intel 64 and IA-32 Architectures Software Developer's Manuals*, from http://developer.intel.com/products/processor/manuals/index.htm
[13] *Guidelines for Fibre Channel Use of the Organizationally Unique Identifier*, at http://standards.ieee.org/regauth/oui/tutorials/fibreformat.html
[14] *Address Allocation for Private Internets*, and related other RFCs, at http://tools.ietf.org/html/rfc1918
[15] *IEEE 802.1Q - Virtual LANs* at http://www.ieee802.org/1/pages/802.1Q.html
[16] *A Border Gateway Protocol 3 (BGP-3),* and related other RFCs, at http://tools.ietf.org/html/rfc1267
[17] *OSPF Version 2*, and related RFCs at http://www.ietf.org/rfc/rfc1583.txt
[18] *Guidelines for creation, selection, and registration of an Autonomous System (AS)*, and related other RFCs at http://tools.ietf.org/html/rfc1930
[19] *Unique Local IPv6 Unicast Addresses*, and related other RFCs, at http://tools.ietf.org/html/rfc4193
[20] *IP Mobility Support for IPv4, revised*, at http://www.ietf.org/rfc/rfc3344.txt
[21] *Mobility Support in IPv6*, at http://www.ietf.org/rfc/rfc3775.txt
[22] *Enhanced Route Optimization for Mobile IPv6*, at http://www.ietf.org/rfc/rfc4866.txt
[23] Carlos J. Bernardos, Ignacio Soto, and María Calderón, *IPv6 Network Mobility*, The Internet Protocol Journal, Volume 10, No. 2, June 2007
[24] *Locator/ID Separation Protocol (LISP)*, at http://tools.ietf.org/html/draft-farinacci-lisp-10
[25] *Uniform Resource Identifiers (URI): Generic Syntax*, and related other RFCs, at http://www.ietf.org/rfc/rfc2396.txt
[26] *Lightweight Directory Access Protocol (LDAP); Technical Specification Road Map*, and related other RFCs at http://tools.ietf.org/html/rfc4510
[27] *[MS-ADTS]: Active Directory Technical Specification*, at http://msdn.microsoft.com/en-us/library/cc200343.aspx
[28] *The Kerberos Network Authentication Service (V5)*, and related other RFCs at http://tools.ietf.org/html/rfc4120
[29] *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*, and related other RFCs at http://tools.ietf.org/html/rfc3280
[30] *The FreeIPA Project* at http://freeipa.org
[31] *Extensible Messaging and Presence Protocol (XMPP): Core*, and related other RFCs at http://xmpp.org/rfcs/rfc3920.html
[32] *XMPP Standards Foundation* at http://xmpp.org/
[33] *virt-image(5) - Linux man page,* at http://linux.die.net/man/5/virt-image
[34] *Virtualization Management (VMAN) Initiative*, Distributed Management Task Force, Inc. at http://www.dmtf.org/standards/mgmt/vman/
[35] see http://xml.coverpages.org/ni2007-09-11-a.html
[36] *virt-convert - modular conversion tool to convert between virtual machine formats(currently supports conversion between vmware, virt-image, and AMI formts*, at http://thincrust.org/tooling.html
[37] *VMware Infrastructure SDK*, at http://www.vmware.com/support/developer/vc-sdk/
[38] *Libvirt virtualization API project*, at http://libvirt.org/
[39] *Host Extensions for IP Multicasting,* and related other RFCs at http://www.ietf.org/rfc/rfc1112.txt
[40] Ohmori, M., Okamura, K., Araki, K., *Design of scalable interdomain IP multicast architecture*, Proceedings. 15th International Conference on Information Networking, 2001.
[41] *Interdomain Multicast Solutions Guide* , Cisco Press Networking Technology Series., Cisco Systems 2004
[42] *LISP for Multicast Environments*, at http://tools.ietf.org/html/draft-farinacci-lisp-multicast-01
[43] D. Mills, "RFC 1305 Network Time Protocol (Version 3) Specification, Implementation and Analysis." *IETF* March 1992
[44] IEEE Std. 1588-2004 "Precision clock synchronization protocol for networked measurement and control systems", 2004, *TC9-Technical Committee on Sensor Technology of the IEEE I&M Society*
[45] Eidson, John; Garner, Geoffrey M.; Mackay, John; Skendzic, Veselin; "Provision of Precise Timing via IEEE 1588 Application Interfaces", *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, 2007*. ISPCS 2007. 1-3 Oct. 2007 Page(s):1 – 6
[46] Lee, Kang; Song, Eugene; "Object-oriented Model for IEEE 1588 Standard", *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication, 2007*. ISPCS 2007., 1-3 Oct. 2007 Page(s): 7 – 12
[47] *Advanced Message Queuing Protocol*, at http://jira.amqp.org
[48] *W3C Semantic Web Activity*, at http://www.w3.org/2001/sw/