Windows Azure and SIP

RDC internship Technical Report
RDC-TR-11-1

Florence Guitard
Institut Supérieur d'électronique de Paris (ISEP)
Paris, France
fguitard@isep.fr

supervised by
Lukas Kencl
R&D Centre for Mobile Applications (RDC)
Faculty of Electrical Engineering, Czech Technical University in Prague
Prague, Czech Republic
lukas.kencl@fel.cvut.cz

July 2011

Abstract

Cloud Computing has emerged over the past few years as a new business paradigm for providing and obtaining IT services. By hosting and delivering services over the Internet, Cloud Computing allows business owners to focus on strategic projects rather than on managing their datacenters, cutting operational and capital costs. Built upon previous concepts such as Grid, Utility or Autonomic Computing, it meets the increasing requirements of power and mobility in various kinds of applications (websites, email services, office softwares), and particularly those imposing a variable load and needing massive scaling. However, many issues have still to be addressed.

Similarly, for the past couple of years, SIP has become the most widely used protocol for VoIP. SIP enables interactive user session that involves multimedia elements by allowing endpoints to exchange messages, register user location and move between networks. SIP servers play an important part as key elements in these networks.

The aim of this Technical Report is to offer a better understanding of these two emerging services and a discussion about the way to see both of them as a single service.

Thus, following an overview of cloud computing, we focus on a concrete cloud platform: Windows Azure. Then, we present the Session Initiation Protocol and a technical study of the different servers involved. To conclude, we discuss options to integrate SIP communications possibilities into a cloud system, Windows Azure.

Part I

Cloud Computing

0.1 A view of Cloud Computing

0.1.1 Key features

Definition Cloud computing refers to a new computing model in which resources (CPU, storage) are provided as general utilities that can be leased and released by users through a digital network (WAN, Internet) in an on-demand way. It is a location-independent computing. It refers to both the applications delivered as services over the Internet and the hardware and systems software in the data centers that provide those services [11]. Applications are provided and managed by the cloud server instead of having been downloaded and installed on the user own device and data is remotely stored in the cloud and processed by the cloud server too. Here is the definition given by the National Institute of Standards and Technology (NIST) :

> Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [12].

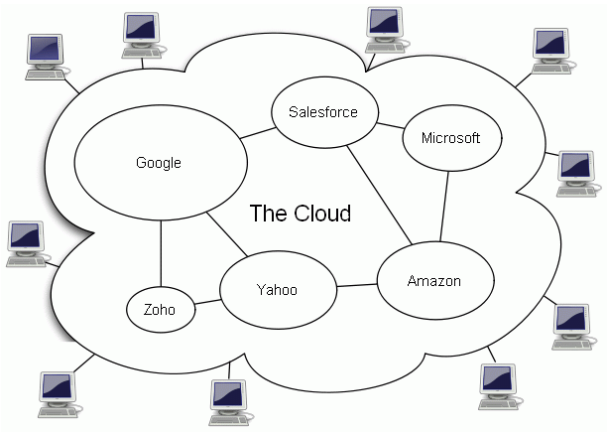


Figure 1: The Cloud [19]

Concepts To understand what is cloud computing, main concepts behind this idea have to be defined.
Cloud : the data center hardware and software [11].
Virtualization: A technology that abstracts away the details of physical hardware and provides virtualized resources for high-level applications. Virtualization forms the foundation of cloud computing, as it provides the capability of pooling computing resources from clusters of servers and dynamically assigning or reassigning virtual resources to applications on-demand [22]. It enables to freed up computers by allowing the execution of multiple instances of an operating system or computing environment on them.
Grid Computing : Pooling compute resources. A distributed computing paradigm that coordinates networked resources to achieve a common computational objective. Cloud computing is similar to Grid computing in that it also employs distributed resources to achieve application-level objectives [22]. These two aspects of cloud computing (Grid Computing and Virtualization) show its scaleability : thanks to them, an operating system can exist across many servers instead of one and applications could use the resources of the whole network of hardware instead of a single machine.

Utility Computing : Model of providing resources on-demand and charging customers based on usage rather than a flat rate. [22]. It's a pay-per-use billing model of metered services (CPU, memory, network bandwidth or storage for example). The goal is to maximize resource utilization and minimize their operating costs.

Autonomic Computing : Aims at building computing systems capable of self-management, i.e. reacting to internal and external observations without human intervention [22]. Cloud Computing includes this kind of computing but focuses on cutting down cost rather than on system complexity.

Characteristics These technologies enable the developpement of the new attractive characteristics of a Cloud system, compared to traditionnal data centers :
On Demand self service : Users who benefit of a cloud computing service can re-provision technological infrastructure resources. When they need more computational power or storage, they needn't buy more servers but more service from cloud provider, following the "Pay as you go" model.
Broad network access : Services are mainly web-based, so hardware is becoming less important : users can access anywhere over any web-enabled device, with an infinite availability.
Resource pooling : Providers pool large amount of resources from data centers and make them easily accessible to multiple resource consumers. It provides much flexibility to infrastructure providers for managing their own resource usage and operating costs.
Rapid elasticity : Resources can be rapidly (near real-time) allocated and de-allocated on demand within the same logical system. No time spent for servers or softwares to be installed and configured. Dynamic resource provisioning allows users to provide resources based on the current demand and not the peak load, which can considerably lower the operating costs.
Measured service : Users can use more hardware resources only when there is an increase in their needs and pricing is based on the utility computing model.

0.1.2 Classification of Cloud Implementations

Let's describe the architecture of a cloud computing system.

The service models We can share cloud computing services within 5 layers. Cloud services belong to the same layer if they have equivalent levels of abstraction [16].

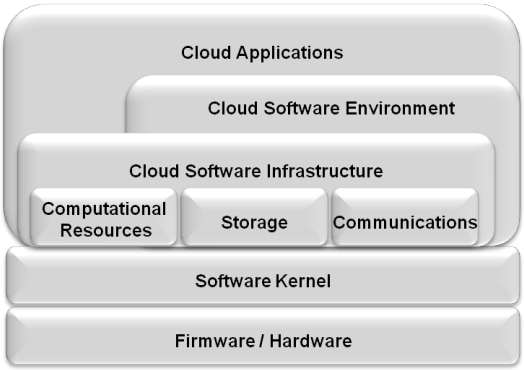


Figure 2: The service models [16]

The servers layer includes the hardware layer and the software kernel :
Firmware/Hardware Layer : This layer is responsible for operating, managing and upgrading the hardware on behalf of its consumers. It includes : physical computing, switching, and routing hardware.
Software Kernel : This layer is responsible for managing basic software of physical servers. It includes OS kernel, hypervisor (virtual machine monitor VMM), clustering middleware.

VMM displays a uniform view of underlying hardware, making machines from different users look the same. VMM also map and remap virtual machines to available hardware resources.
Cloud Software Infrastructure or Infrastructure as a Service (IaaS) : Provides fundamental software resources (a pool of storage and computing resources) by partitioning the physical resources

thanks to virtualization technology seen in 0.1.1, page 2. It enables dynamic resource assignment.
Cloud Software Environment or Platform as a Service (PaaS) : Using the infrastructure layer, it sustains cloud applications for cloud applications' developers (contains operating systems, application frameworks, runtimes, SOA integration, databases, server softwares).
Cloud Applications or Software as a Service (SaaS) : Provides specific cloud applications.

The deployment models Cloud services can be divided into different types based on access and location :
Public : A public cloud is available to anyone on the Internet by signing up. Any user can sign up to use the public cloud (e.g. Microsoft Windows Azure) [16].
Private : A private cloud is a proprietary cloud environment that only provides cloud services to a limited number of users [16].
Hybrid : A hybrid cloud provides services that run on a public cloud infrastructure, but limits access to it with a virtual private network (VPN) [16].

0.1.3 Issues

Although cloud computing seems attractive to many users, several issues can not been fully addressed. I will list the essential ones :
Business Continuity : Availability can represent a concern for cloud computing users. Actually, they let all their processing to the hands of an external provider. A disruption can affect services shared by many users on the network.
Data migration : There is no official standard between the providers and the storage APIs are for the moment mainly proprietary. Thus, data migration from one site to another can turn out to be difficult and complex.
Information security : Storing data in the cloud may expose the user to potential violation of privacy. This safety risk can be increased when the provider owing user's information reside in a different country, with different laws (cyber espionage, user profiling...).
Data transfer costs : Applications continue to become more data-intensive. This can lead to a costs issue (100-150$/terabyte).
Scalable Storage : It can be a drawback as well concerning persistent storage.
Debugging at large scale : Difficulty to remove errors in these large distributed systems.

0.2 Focus on a PaaS : Windows Azure Platform

0.2.1 Presentation

The Windows Azure Platform is a Microsoft cloud platform used to build, host and scale web applications through Microsoft datacenters. It belongs to the cloud layer PaaS seen in 0.1.2 page 4, by providing this platform. It is composed of three parts : Windows Azure providing a Windows-based environment for running applications and storing data on servers; SQL Azure providing data services in the cloud based on SQL Server and .NET Services providing distributed infrastructure services to cloud-based and on-premises applications. Each of these parts is detailed below.

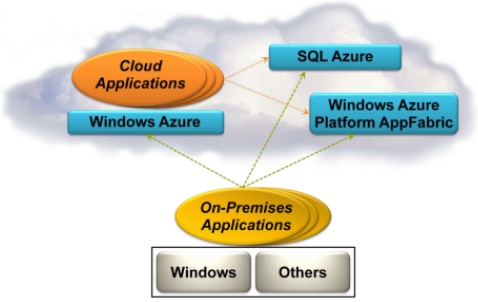


Figure 3: Windows Azure Platform [13]

0.2.2 Windows Azure

Windows Azure is in charge of computing, storaging, hosting, and managing capabilities on top of the fabric. It is connected to local applications with secure connectivity, messaging and identity management. It supports applications built on the .NET Framework or other frameworks in Windows languages like C#, Visual Basic, C ++ and Java using Visual Studio or another IDE. Web applications can be implemented using ASP.NET, WCF or PHP technologies. Windows Azure stores data in blobs (binary large objects), tables (non relationnal SQL), and queues for communication between the components of applications, via a REST architecture. Each application is assigned a configuration file which is modified by a XML-based description to set its behavior. The fabric controller is the software in charge of monitoring components in data centers such as physical resources, VMs and applications. This controller executes the configuration file and choose the more optimized location of the applications, that's to say, which physical servers will support the applications. The management of applications by the owner is made through a web portal thanks to its Windows Live ID.

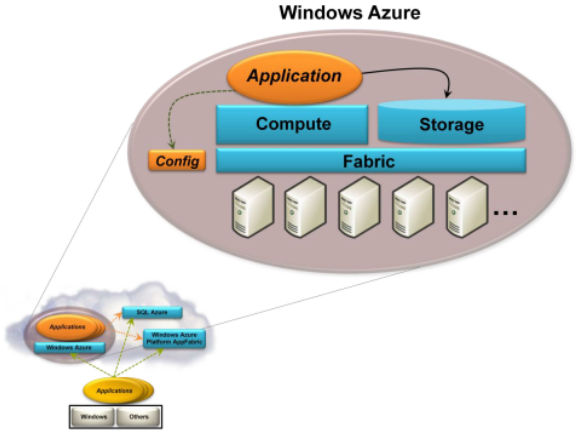


Figure 4: Windows Azure [13]

Web/Worker Roles An application has multiple instances, running a copy of the but creates applications using Web roles and/or Worker roles, and specifies how many instances of each role to run. Windows Azure creates a VM for each instance, then runs the application in those VMs. Web roles accept HTTP (or HTTPS) requests via Internet Information Services (IIS). A Web role can be implemented using ASP.NET, WCF, or another technology. Windows Azure provides built-in load balancing to spread requests across Web role instances that are part of the same application. Windows Azure also provides Worker roles. A Worker role is like a Web role instance but isn't hosted in IIS. Worker role instances can communicate with Web role instances with queues. A Web role instance can insert a work item in a queue, and a Worker role instance can remove and process this item. They can also communicate via direct connections via Windows Communication Foundation (WCF) or another technology.
Each VM, running Web or Worker roles, contains a fabric agent that allows the application to interact with the fabric controller. A choice of four VM sizes is offered to users: one core, two cores, four cores, and eight cores to scale applications performance (number of running instances defined by users in the configuration file). The fabric controller runs VMs, assign them to cores and runs the number of instances of the application. It is able to detect an instance failure and start a new one.

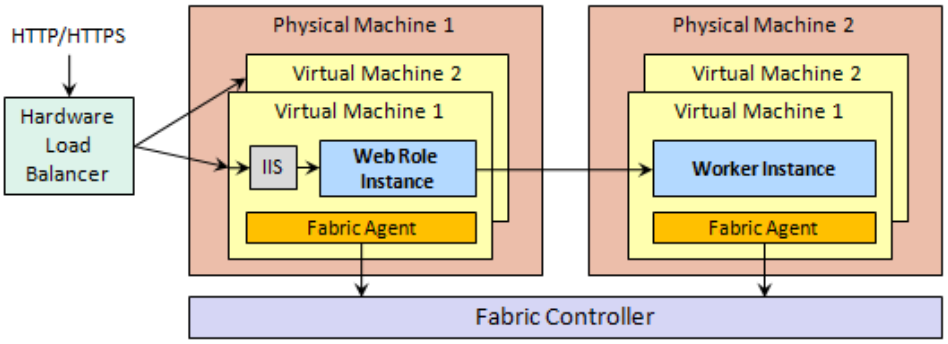


Figure 5: Web/Worker Roles [16]

Microsoft provides Visual Studio project templates for creating Windows Azure Web roles, Worker roles, and combinations of the two. The Windows Azure SDK contains a local version of Windows Azure environnment : Windows Azure Development Fabric (Windows Azure storage, fabric agent...), to deploy applications in the cloud after their local development. Still, Windows Azure provides other services like CPU, bandwidth or storage information.

Data Storage There are different ways to store information. The simplest one is to use blobs, present in containers for unstructured data. Large blobls can be divided into blocks for a more efficient transfer. When a failure occurs, retransmission can resume with the most recent block. Blobs can also have associated metadata as a description of the object. A content delivery network (CDN) is used to store frequently accessed data at locations closer to the applications that use it. Another way to use blobs is through Windows Azure XDrives, which can be mounted by a Web/Worker role instance. To work with data in a more fine-grained way, Windows Azure storage provides non relational tables (structured data): sets of entities with properties but without a defined schema. Properties can have various types (int, string, Bool, DateTime...). Table's data are accessed by ADO.NET Data Services or LINQ. As seen in 0.2.2 page 6, queues are used in communications between Web role instances and Worker role instances, and not to store data.

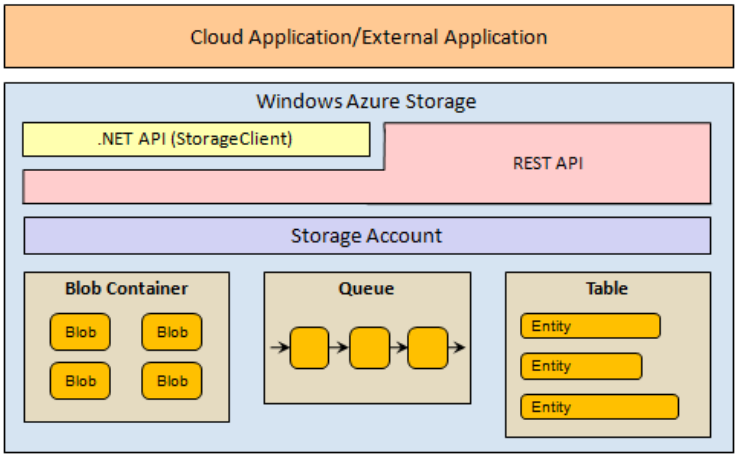


Figure 6: Data Storage [16]

Windows Azure storage is replicated three times. This replication allows fault tolerance, since losing a copy isn't fatal. All three Windows Azure storage styles use the conventions of REST to identify and expose data. Everything is named using URIs and accessed with standard HTTP operations. A .NET client can rely on ADO.NET Data Services and LINQ, but access to Windows Azure storage from, for example a Java application can just use standard REST. The Windows Azure platform charges independently for compute and storage resources. Data can be accessed from non Windows Azure applications.

0.2.3 SQL Azure

SQL Azure is responsible for storing and working with relational cloud data. It is composed of SQL Azure Database and "Huron" Data Sync. The first one, based on Microsoft SQL Server, offers indexes, views, stored procedures, triggers ...., via ADO.NET or other Windows data access interfaces. The possibility to use data in a local software is also given. The second one synchronizes relational data across these local DBMSs.

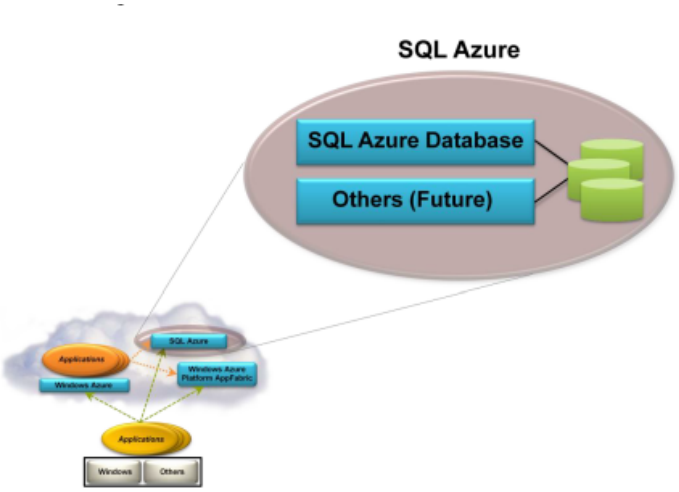


Figure 7: SQL Azure [13]

The application accesses data via a protocol called Tabular Data Stream (TDS). This is the same protocol used to access a local SQL Server database, and so a SQL Azure Database application can use any existing SQL Server client library. This includes ADO.NET, ODBC, and PHP.

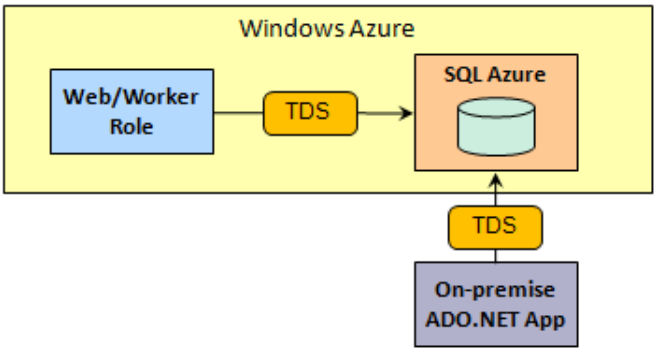


Figure 8: Accessing SQL Azure data [16]

Because SQL Azure Database looks like a traditionnal SQL Server system, standard tools can also be used (SQL Server Management Studio, SQL Server Integration Services, BCP). All data stored in SQL Azure Database is replicated three times as in Windows Azure storage. In its first release, the maximum size of a single database in SQL Azure Database is 10 gigabytes. An application whose data is within this limit can use just one database, while an application with more data will need to create multiple databases.

0.2.4 Windows Azure platform AppFabric

Windows Azure platform AppFabric provides cloud-based infrastructure services to connect distributed applications. It contains two main components : the Service Bus and the Access Control. The Service Bus exposes applications's services endpoints on the Internet or on-premises by their URI, used to locate them and access the services. The Access Control let a client application authenticate itself and provide a server application with identity information. Then, the server has to decide what this application is allowed to do.

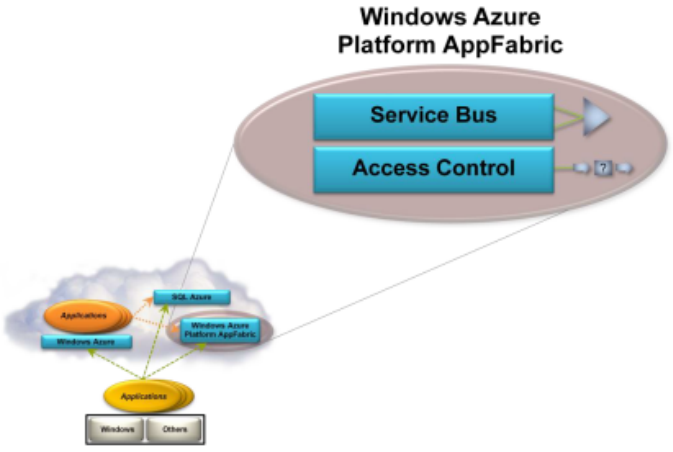


Figure 9: Windows Azure platform AppFabric [13]

Service Bus A problem may occur when a cloud user need to connect to services from other organizations. Actually, these organizations need to find user's services endpoints that they have to connect to. Another issue is for requests from softwares of these organizations to get through to the user's services and through their firewalls, whithout opening firewall ports. Service Bus addresses these issues.
Firstly, user's WCF service registers endpoints with Service Bus. For each registered endpoint, Service Bus exposes its own corresponding endpoint. Service Bus also assigns user's organization a URI root. This allows user's endpoints to be assigned specific, discoverable URIs. User's application must also open a connection with Service Bus for each endpoint it exposes. Service Bus holds this connection open, which solves two problems. First, NAT is no longer an issue, since traffic on the open connection with Service Bus will always be routed to the user's application. Second, because the connection was initiated from inside the firewall, there's no problem passing information back to the application via this connection, the firewall won't block this traffic. When a client running in the cloud or on-premises at some other organization wishes to access the user's services, it contacts the Service Bus registry to find the endpoint. This request uses the Atom Publishing Protocol, and it returns an AtomPub service document with references to the endpoints Service Bus exposes on behalf of the user's application. Once it has these, the client can invoke operations on the services exposed through these endpoints. For each request Service Bus receives, it invokes the corresponding operation in the endpoint exposed by user's WCF services. Clients can be built application that exposes its services via Servie Bus with WCF or other technologies, such as Java.

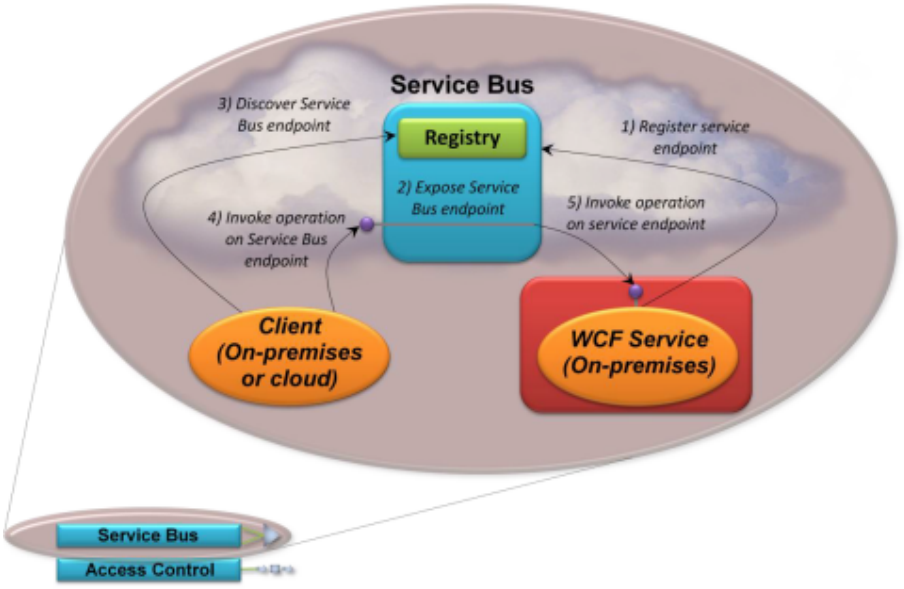


Figure 10: Service Bus [13]

Access Control Based on a user's identity information, an application might make decisions about what that user is allowed to do. The part that plays the Access Control component is conveying application identity information via REST-based services. To communicate with a particular server application, the client must first get a token that contains identity information about him. This information is expressed as one or more claims, descriptions. This token is issued by the Access Control server, and to get it, the client application must first authenticate itself. Once the client application has authenticated itself, the Access Control service creates another token containing identity information for this client. The format of the token is a set of human readable name/value pairs, each of which expresses some claim about this application. The server application for which this token is intended can define rules about how the token will be created. Then, it's sent back to the client application. This new token is signed using a key provided by the server application. The client sends this signed token to the server, which then validates the token's signature and uses the claims it contains.

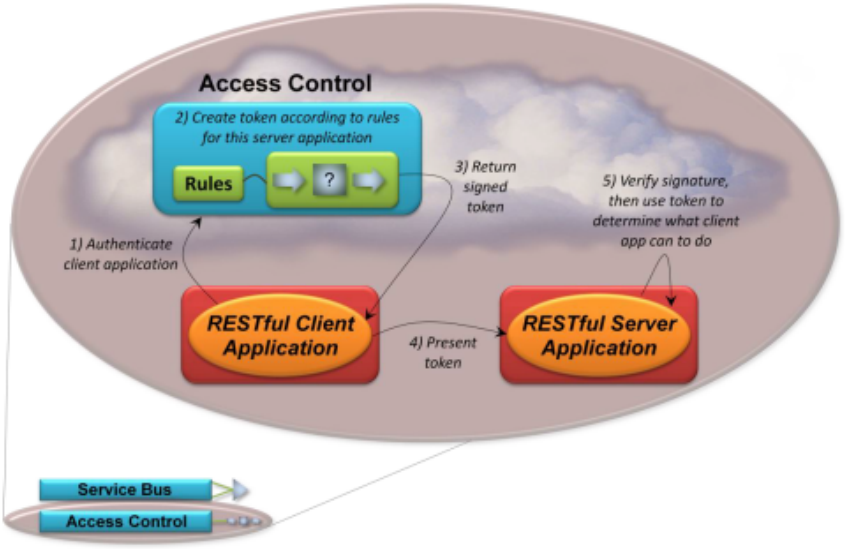


Figure 11: Access Control [13]

0.2.5 Windows Azure and other cloud solutions

This section studies the place of Windows Azure in the actual market. For the comparison, I chose the best known and used alternatives to Windows Azure : Google App Engine and Amazon EC2. Google App Engine, a PaaS like Windows Azure, offers all the scalability and infrastructure needed to enable development of web applications. Amazon EC2 provides basic cloud infrastructure (IaaS) such as computing resources, storage, communication queues, database to run virtual machines and applications. The comparison is summarized in the tabular below :

	Window Azure	**Google App Engine**	**Amazon EC2**
Release	private beta	public beta	commercially available
Service Model	PaaS	PaaS	IaaS
Application Model	Windows general-purpose	Web	General-purpose
Computing Model	Users provide .NET code that runs on Microsoft Common Language Runtime (CLR) VM according to users' environment specifications; Predefined roles of application instances	Users provide web application code (Python or Django) in predefined Google web application frameworks	OS Level on a Xen Virtual Machine ; Xen Virtual Machine images uploaded by users to the infrastructure; Predefined APIs to instantiate and manage them.
Storage Model	Azure application storage service and SQL Data Services	BigTable and MegaStore	Elastic Block Store, Simple Storage Service (S3) and SimpleDB
Development tools	Yes, integration into Visual Studio, support for any .NET languages,	Yes, have basic editing, local simulation, and deployment tools. Language selection limited to Python and Django. Application-level tools such as Google Web Toolkit (GWT) do not seem to have any integration with Google App Engine.	Not applicable. Amazon simply runs your virtual machines and does not care which development platform you are using on top of the base OS.
Automatic Scaling	Yes, based on application roles and a configuration file specified by users	Yes, transparent to users	Yes, change of the number of instances based on parameters that users specify
Queuing for machine communications	Yes, queues in Windows Azure storage	No	Yes, Simple Queue Service (SQS)
Integration with other services	So called .NET services: Access control services, workflow service, service bus ; Live Mesh ; Various Live services (contacts, mail, maps and so on.) ;	Yes, with existing Google services: authentication, mail, base, calendar, contacts, documents, pictures, spreadsheets, YouTube	No

Table 1: Cloud solutions comparison

Part II

SIP

0.3 Definition

SIP : Session Initiation Protocol SIP is a signalling protocol used for establishing sessions in an IP network [5]. It is the first protocol to enable multi-user sessions regardless of media content [4], which means scalability and adaptation to different architectures and deployment scenarios, independent of the transport layer (TCP, UDP or SCTP) It is now a specification of IETF.

IETF : Internet Engineering Task Force IETF is a Task Force consisting of over 80 working groups responsible for developing Internet standards. The IETF operates under the auspices of ISOC [14].

Functionnalities SIP is used for Internet conferencing, telephony, presence, events notification and instant messaging. Voice over IP community has adopted this protocol as its protocol of choice for signalling.
SIP is an application-layer control protocol that can establish, modify, and terminate these kinds of sessions. It can also invite participants to already existing sessions. It is able to determine the end system to be used for communication, the willingness of the called party to engage a communication, the media to be used, the establishment of session parameters, and session management like transfer, termination or update of sessions. This protocol handles name mapping, redirection of services regardless of the users network location (can translate from a user's name to its current network address).

SIP and other protocols SIP is a request-response protocol that looks like other Internet protocols such as HTTP and SMTP. Not depending on other protocols, it uses them, such as HTTP with URLs for addressing or SDP for conveying session information. To guarantee its proposed services, it works with Real-time Transport Protocol (RTP) for transportation of real-time data, Real-Time streaming protocol (RTSP) for control of streaming media delivered, and Session Description Protocol (SDP) for description of sessions. The overlapping protocols (H.323, MGCP, and MEGACO), sitting architecturally below SIP, can benefit from this inter-work.

0.4 SIP Signaling

Networks elements User Agents (UA) are essential logical entities to convey SIP information. They can play the part of client and server only for the duration of one transaction :
Server: A server is a network element that receives requests in order to service them and sends back responses to those requests. Examples of servers are proxies, user agent servers, redirect servers, and registrars [20].
Client: A client is any network element that sends SIP requests and receives SIP responses. Clients may or may not interact directly with a human user. User agent clients and proxies are clients [20].
User Agent Client (UAC): A user agent client is a logical entity that creates a new request, and then uses the client transaction state machinery to send it [20].
User Agent Server (UAS): A user agent server is a logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request [20].
Proxy, Proxy Server: An intermediary entity that acts as both a server and a client for the purpose of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity 'closer' to the targeted user. Proxies are also useful for enforcing policy (for example, making sure a user is allowed to make a call).
A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it [20].
Redirect Server: A redirect server is a user agent server that generates 3xx responses to requests it receives, directing the client to contact an alternate set of URIs [20].

Registrar: A registrar is a server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles [20].

SIP messages The syntax of SIP messages resembles that of HTTP. The request first line (method) and the header determine the nature of the call (services, addresses, and protocol features). The message body is independent of the SIP protocol and can contain anything.
Message: Data sent between SIP elements as part of the protocol. SIP messages are either requests or responses [20].
Request: A SIP message sent from a client to a server, for the purpose of invoking a particular operation [20].
Response: A SIP message sent from a server to a client, for indicating the status of a request sent from the client to the server [20].
Method: The method is the primary function that a request is meant to invoke on a server. The method is carried in the request message itself. Example methods are INVITE and BYE [20].

SIP Methods

The commands that SIP uses are called methods. SIP defines the following methods:

SIP Method	Description
INVITE	Invites a user to a call
ACK	Used to facilitate reliable message exchange for INVITEs
BYE	Terminates a connection between users or declines a call
CANCEL	Terminates a request, or search, for a user
OPTIONS	Solicits information about a server's capabilities
REGISTER	Registers a user's current location
INFO	Used for mid-session signalling

SIP responses
The following are SIP responses:

- 1xx Informational (e.g. 100 Trying, 180 Ringing)
- 2xx Successful (e.g. 200 OK, 202 Accepted)
- 3xx Redirection (e.g. 302 Moved Temporarily)
- 4xx Request Failure (e.g. 404 Not Found, 482 Loop Detected)
- 5xx Server Failure (e.g. 501 Not Implemented)
- 6xx Global Failure (e.g. 603 Decline)

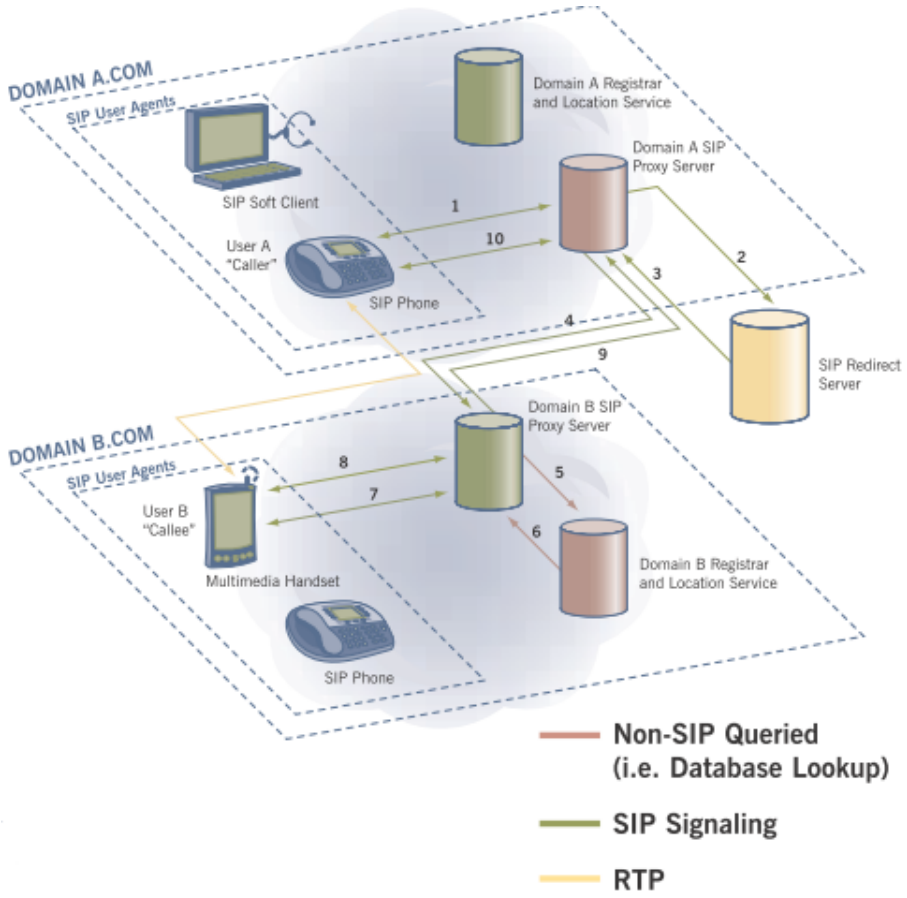
Figure 12: SIP Methods and Responses [3]

Header: A header is a component of a SIP request message that conveys information about the message. It is structured as a sequence of header fields [20]. The user agent identifies itself by a this text describing the software/hardware/product involved for the transaction. It's sent to the SIP server.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via: SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bK776asdhds
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710@pc33.atlanta.com
CSeq: 314159 INVITE
Contact: <sip:alice@pc33.atlanta.com>
Content-Type: application/sdp
Content-Length: 142
```

Figure 13: Example of a SIP header [20]

Signaling process The User Agents designate agents found in the SIP phones, softphones (VoIP software) of computers and PDAs or SIP gateways. In theory, sessions can be established directly between two user agents, two telephones for example. But this requires knowing the IP address of the callee. IP addresses are not the panacea : they may not be public (behind a NAT) or change and is much more complicated to remember than a URI (Uniform Resource Identifier). That's why a user is identified by a URI. A SIP URI has the following format :*sip:username:password@host:port*. For a secure transmission, sip is replaced by sips and SIP messages must be transported over Transport Layer Security (TLS).
The User Agents can then register with the Registrars servers to indicate their current location, that is to say, their IP address. These servers manage requests that contain associations of IP addresses and URIs, which will be stored in a database.
A SIP Proxy act as an intermediary between two user agents that don't know their locations (IP addresses). The proxy can query the database where are the desired associations to direct messages to the callee. It relays messages to establish, monitor and end the session. Once it is established, the data don't get through the proxy anymore. They are exchanged directly between User Agents.
An illustration is given to summarize the establishment of a SIP session between caller A and calle B, in distinct domains :



1. Call User B
2. Query "How do I get to User B, Domain B?"
3. Response "Address of Proxy Controller for Domain"
4. Call 'Proxied' to SIP Proxy for Domain B
5. Query "Where is User B?"
6. User B's Address
7. Proxied Call
8. Response
9. Response
10. Response
11. Multimedia Channel Established

Figure 14: Establishment of a SIP session in distinct domains [4]

0.5 SIP Servers

SIP Servers have been evoked in 0.4 page 13, as SIP network elements. We've seen they could be proxies, UAS, redirect servers or registrars servers as well. The next step will focus on these particular servers, able to exchange messages between SIP endpoints, authenticate users, register, manage user location and handle routing and security policies.

0.5.1 Classification

Definitions have been given in section 0.4 page 13.

Registrar Server This server manages location registration messages. REGISTER requests are generated by clients in order to establish or remove a mapping between their externally known SIP address(es) and the address(es) they wish to be contacted at. The REGISTER request can also be used to retrieve all the existing mappings saved for a specific address. The Registrar processes the REGISTER request for a specific set of domains. It uses a 'location service'-an abstract location database-in order to store and retrieve location information. The location service may run on a remote machine and may be contacted using any appropriate protocol (such as LDAP) [20].

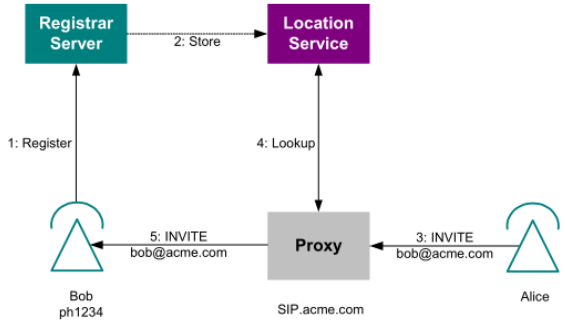


Figure 15: Registration Process [18]

Redirect Server This server returns simple 'contact this address' responses, i.e. 3xxx responses (figure 12). Accepting SIP request, it redirects the client to contact an alternate set of SIP addresses as Contact headers in the response message provided. There are different kinds of 3xx responses like Moved Permanently/Temporarily or Use Proxy. It doesn't create requests nor receives calls but just maps the address into new addresses to clients. These servers are characterized by high processing capacity (due to fewer messages to process), minimal state overhead, dependency on client device.

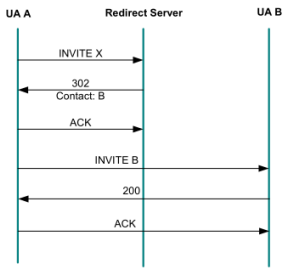


Figure 16: Request Redirection [18]

Proxy Server This server forwards, relays SIP requests and responses by providing dynamic associations of SIP endpoints. The proxy manages two types of transactions-server transactions to receive requests and return responses, and client transactions to send requests and receive responses.
Moreover, it can validate requests, authenticate users, fork requests, resolve addresses, cancel pending calls, Record-Route and Loose-Route, and detect and handle loops too. The versatility of SIP proxies allows the operator/system administrator to use the proxies for different purposes and in different locations in the network (such as edge proxy, core proxy and enterprise proxy).
A proxy server is designed to be mostly transparent to UAs. Proxy servers are allowed to change messages only in specific and limited ways.
These servers are characterized by flexibility (network service implementations), reliability (replication), scalability (partitioning), potential overload if not properly scaled.
Proxies can be 'stateful' or 'stateless'. In the first case, it maintains the client and server transaction state machines defined by this specification during the processing of a request whereas in the second case, it's the opposite, assuring client anonymity.
Stateful proxy processes transactions and not messages, maintaining the call context, whereas stateless proxy just relays individual messages. Stateful proxy processing consists on a UAS/UAC replication. Response provided by stateless proxy is not based on this model. Forking and TCP proxies require knowing state for reliability, advanced services as well for their execution, stateful proxy provides the fitting response. Stateless proxy has a restricted gateway access but a better processing capacity. Moreover, memory consumption, throughput, implementation complexity and underlying SIP stack complexity present stateful proxy drawbacks.
Before routing a request, a SIP Server (proxy or redirect) needs to validate the request to make sure it can actually proceed with processing this message by passing the following validity checks: reasonable syntax check, URI scheme check, max-forwards check, loop detection (optional), proxy-require and authentication.
Once a proxy has validated an incoming request and decided to forward it, it must determine the destination(s) to which the message is to be forwarded before sending the messages. The proxy does two types of address resolution: determining the target-set and DNS resolution.
The SIP server processes the following incoming requests: non-INVITE requests (such as BYE and REGISTER), INVITE and CANCEL.
Record-Routing is a SIP mechanism that allows SIP proxies to request being in the signaling path of all future requests that belong to this dialog. A proxy that does not Record-Route an INVITE message should not expect to receive any of the further requests. Proxies should normally Record-Route only requests that set up a dialog (currently INVITE and SUBSCRIBE). However, a proxy may add a Record-Route header to any SIP request if it so wishes. Record-Routing is used to:

- Route information pre/post-processing
- Rewriting Record-Route headers in responses
- Symmetric Record-Route
- Loose-Routing

After processing a request and building a target-set for it, the proxy may choose to forward the request to multiple addresses : that's the forking concept. Different ways of forking exist (parallel, sequential or both of them). When a UAC sends a request to a proxy server, the proxy server may decide to authenticate the originator before the request is processed to ensure that the originator of the request is an authorized user entitled to receive such services and that fields are not altered by a third party SIP.
A loop, a situation where a request that arrives at a proxy is forwarded, and later arrives back at the same proxy, is handled in two ways: Max Forwards (mandatory) or Loop detection (optional).
A spiral is a SIP request that is routed to a proxy, forwarded onwards, and arrives once again at that proxy, but with a different set of values in the fields that affect the routing decision.
Outbound proxy is a proxy that receives requests from a client regardless of the destination of the messages.

0.5.2 SIP Servers Policy

Different ways are offered to SIP servers for handling incoming messages. They are free to choose how dealing with the following issues :

- Where to route the request and based on which location
- Information source (location service/database/presence info/ other)
- Proxy/Redirect/Reject
- Stateful/Stateless forwarding
- Record-routing
- Forking-parallel/sequential/mixed/none
- Authentication
- Loop detection

Many factors must be taken into account, such as :

- Message destination address (Request-URI)
- Domain(s) managed by this server
- Type of request (method)
- Other message fields: From, To, Date, Priority ...
- External parameters, such as time of day

Therefore, the set of rules that govern proxy routing decision making is called Server Policy. Server Policies are typically set by the service providers or administrators that install and configure the SIP Servers. Because of the large number of input variables and output decisions that are available for Server Policies, SIP Servers are highly versatile and flexible elements. This is of key importance for deployment of SIP-based networks. Server Policy can be as complex or as simple as required by network topology, user profiles, traffic load, security risk levels, and other factors specific to the environment in which the server operates.

0.5.3 SIP servers comparison

With the world-wide adoption of SIP, numerous vendors, research organizations and academic institutions are choosing to develop their own SIP servers. Here is a list of several SIP servers, belonging to the wide range of various SIP servers projects, public or not. I chose the following variables : License(s), Programmating language(s), SIP URL, Operating System(s), Applications Domain, Target Market, Latest Release Version and Date, Classification Model of SIP Server, Involved Protocols, Transport Protocol, Requirement of a Service Record, Possibility of Extern Registrations (from other domains).

Org.	**Asterisk**	**FreeSwitch (Sofia-SIP)**	**Hotfoon**	**Kamailio/ OpenSIPS(** formerly OpenSER)	**Microsoft LC/OC Server**	**Prince of Songkhla Univ. Thailand**	**SER** (iptel.org/ Tekelec)	**SipX ECS** (formerly Pingtel)
Lic.	GPL/ Free soft.	Mozilla Public License/ Free soft. /Open source	Freeware	GPL/ Free soft.	Prop.	-	GPL/ Free soft.	L-GPL Open source
Lang.	mostly C, API via AGI (interface): any language able to communicate via the standard streams system or TCP sockets, (C, PHP, Python, Shell, C#, C++, Java, Tcl, Perl, Ruby)	mostly C, API : C/C++, Lua, Javascript/ ECMAScript, Python, PHP, Perl, Ruby, Java, .NET, more... +languages communicating via TCP sockets	-	mostly C, API: Perl, Java, Lua, Python	Microsoft SIP Processing Language scripts (MSPL) or a .NET Framework programming language, such as C# to customize functionalities	Java (JAIN-SIP)	mostly C, API: Perl, Java	mostly C++ and Java, C, Python, Ruby, Shell
URL	-	sofia/my_profile/ user@host Host can be a name or an IP address	sip:hotfoon.com	sip:test@ kamailio.net	-	sip:cnrsipserver. coe.psu.ac.th	sip:your.name@ iptel.org	sip:interop. pingtel.com
OS	Linux/BSD, Mac OS X, Solaris	Linux/BSD, Mac OS X, Solaris, Windows,	-	Linux/BSD, Solaris	Windows	Linux TLE and Mobile Windows	Debian, FreeBSD, Gentoo, NetBSD, OpenBSD, OpenSUSE, Solaris	Linux, FreeBSD
App.	VoIP Gateway, voicemail, basic accounting, conferencing, hot-desking, ENUM, IVR trees, call queuing, automated calls	Recording, Voicemail, Conferencing, RADIUS, IM Proxy, Streaming, Media gateway, Soft-PBX, IVR (modular)	Telephony service: Phone to Phone, PC to Phone, calls	Authentication, Diameter, RADIUS, ENUM, least-cost -routing and many others	Instant messaging, multiparty voice and video calling	Service for voice and video, set of SIP APIs (SIP Stack)	Authentication, Diameter, ENUM, and many others	softphone for VoIP PC desktops and laptops
Market	Enthusiasts, developers, enterprise users	Large soft -switch users, home PBX users, softphone users	-	SIP Service Providers	Enterprises	Research project	SIP Service Providers	Road warriors and telecommuter who want to access an office phone when they travel
Release	1.8.2.3, Jan. 2011	1.0.6, Ap. 2010	9.0, Mar. 2010	3.0.0, Jan. 2010	R2,2007/ R2,2010	- Ottendorf	2.0.0 Mar. 2011	4.4.0,
Class	Registrar	Proxy, Redirect, Registrar	Redirect	Proxy, Redirect, Registrar	Proxy, Redirect, Registrar	Proxy, Redirect; default is Redirect	Proxy, Redirect, Registrar	Proxy
Prot.| Encryp°.	SIP, H.323, IAX,MGCP| TLS, SRTP	SIP, IAX, H323, XMPP, NAT-PMP, STUN, SIMPLE, MRCP| TLS SRTP, ZRTP	LDP| MD5 based 128-bit authentication	SIP, XMPP |TLS	SIP, SIMPLE, XMPP|SRTP, TLS	SIP	SIP| TLS	SIP,IPsec |SSL
Transp.	TCP	By default TCP, UDP (TCP only to specify)	UDP	UDP,TCP, SCTP	TCP	UDP,TCP, SCTP	UDP,TCP	UDP,TCP
SRV	Yes	Yes	No	-	Yes	No	iptel.org	-
Ext. Regist°	Yes, config. file	Yes, XML config. file	Yes (password and registration required)	Yes, by setting the envnmt var. SIP_DOMAIN (SIP_DOMAIN= myserver.foobar.com) or configuring the default resource file for kamctlrc	Yes, add or remove several domains (Open Office Comm. Server 2007 in Properties, Global Properties)	Yes (no password or registration required)	No	Yes

Table 2: SIP servers comparison - [21], [1], org. websites

Part III

Windows Azure and SIP

0.6 SIP services already in Windows Azure ?

0.6.1 Windows Azure OS

SIP servers are defined as Internet servers, providing support for text, audio or video, communications.
We've seen in 0.5.3, page 20 a Microsoft SIP server : Microsoft Lync Server (previously Microsoft Office Communications Server). Operating System servers must be distinguished from SIP servers.
Let's have a closer look at Windows Azure roles. As seen in subsection 0.2.2, page 6, a running application executes multiple instances, managed by the fabric controller [10]. Since a VM is created by Windows Azure for each instance, we can classified VMs as roles. The operating system server running for Windows Azure through each VM is **Windows Server 2008 R2 VHD**. VHD means Virtual Hard Disk and is the hard disk of the VM (disk partitions, file system).

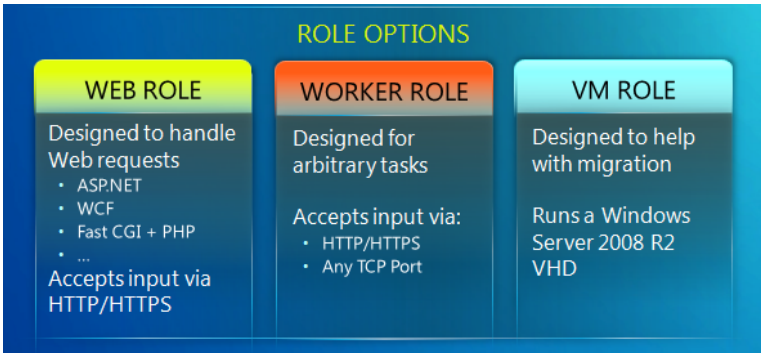


Figure 17: Role Options [10]

Thus, applications see a 64-bit Windows Server 2008 interface :

- A few things require accessing the Windows Azure Agent, e.g., logging
- A desktop replica of Windows Azure is provided for development called the Development Fabric [17].

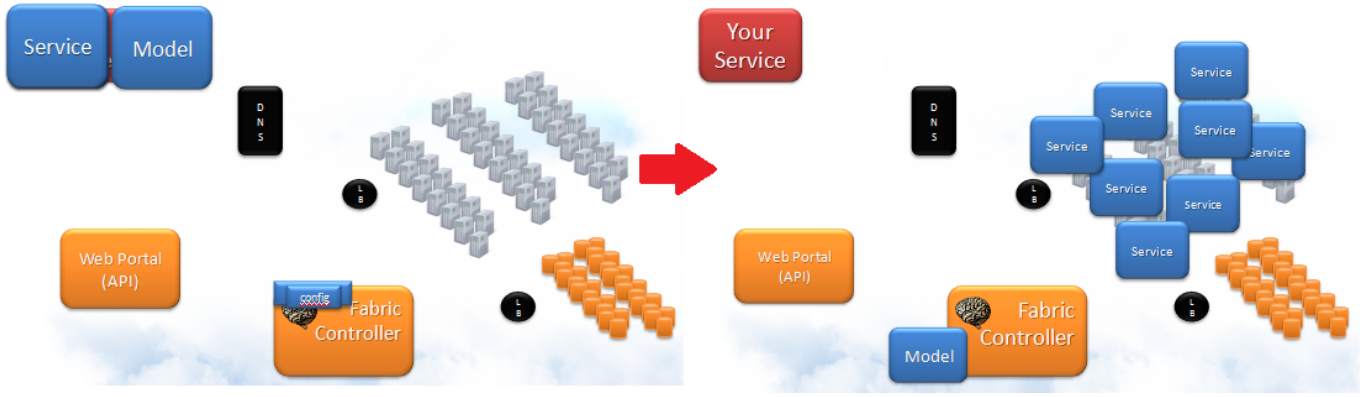


Figure 18: Service Hosting [17]

0.6.2 Windows Azure Services

Live Services The three main components of Windows Azure platform have been detailed in section 0.2, page 5 : Windows Azure, SQL Azure and Windows Azure platform AppFabric. Actually, Windows Azure Platform is made of Windows Azure and five services : SQL Azure, Windows Azure platform AppFabric ; **Live Services**, SharePoint Services and Dynamics CRM

Services must be added:

Figure 19: Windows Azure Services [17]

Live Services are APIs for developpers implementing cloud applications or websites, allowing them to integrate Windows Live Services (such as Gallery, Hotmail, Messenger...).

Windows Live Messenger is an instant messaging client. It uses the Microsoft Notification Protocol (MSNP) over TCP (and optionally over HTTP to deal with proxies) to connect to the .NET Messenger Service, a service offered on port 1863 of 'messenger.hotmail.com.' MSNP's drawback is security (no proper encryption). The corresponding servers are the .NET Messenger Service servers. But Windows Live Messenger (and MSN Messenger) are clients that do not support SIP (not to be confused with Windows Messenger). We can conclude **SIP services are not integrated in Windows Azure Platform.**

0.6.3 Other Microsoft Cloud SIP solutions

Other Microsoft Cloud solutions are compatible with SIP. Whereas Windows Azure Platform is a Platform as a Service, Microsoft's cloud computing offers Softwares as a Service too.
Windows Office Communications/Lync Servers have been evoked. An online client version, for the cloud, exists as Software as a Service : Microsoft Office Communications Online from the Business Productivity Online Standard Suite (BPOS), while Microsoft Lync. is part of the new Office 365. This time, SIP is supported for communications in these softwares.

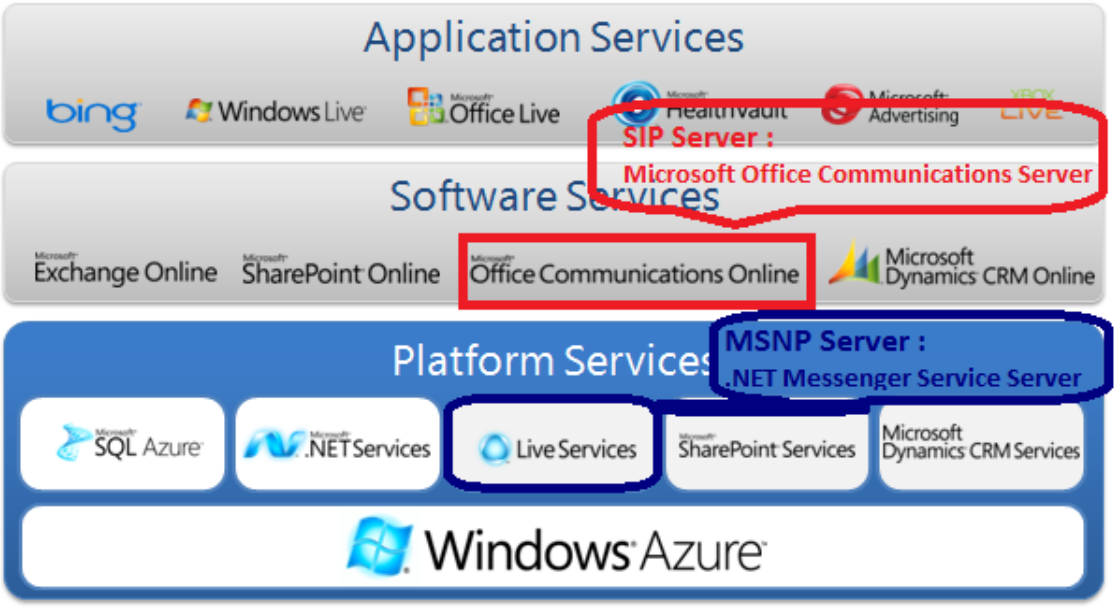


Figure 20: The Microsoft Cloud [17]

0.7 Windows Azure and SIP servers

This section represents a discussion about the way to make a SIP server communicate with Windows Azure Platform. The purpose of this part consisting on providing SIP implementations directions, further investigation has to be led to confirm the given proposals.

0.7.1 Transport protocol supported by Windows Azure

By default, TCP is supported by Windows Azure, whereas UDP is not. In February 2010, Windows Azure Platform was commercially lauched, and in October 2010, a few enhancements were added to the platform such as IT/development experience, role or admin tasks improvements. Among them, the creation of a new service : Windows Azure Connect, providing a secure network connectivity between on-premises and cloud. UDP and TCP are both supported. On the MSDN website, this possibility offered by the new network service is reported :

> When you deploy a VM role to Windows Azure, you are running an instance of Windows Server 2008 R2 in Windows Azure. Note also that a server instance running in Windows Azure is subject to certain limitations that an on-premises installation of Windows is not. Some network-related functionality is restricted; for example, in order to use the UDP protocol, you must also use Windows Azure Connect [2].

However, it said to be not suited for high-volume communications, Windows Azure Connect using a SSTP (SSL) intern connection ; TCP would be prefered regarding the performance/latency factor.

0.7.2 SIP service model

To implement a SIP service, or any hosted service with Windows Azure, it's necessary to create the service model, that's to say the service settings (ServiceConfiguration.cscfg) and the service properties (ServiceDefinition.csdef), like number of instances, role types and ports. Then, users can activate a remote access of role instances and they have to define endpoints of communication (external or internal). Finally, they must develop the codes of their applications and the defined roles.
An Azure role has a firewall/load-balancer in front of it. Therefore, when opening a port in a role, the port must be declared in the ServiceDefinition.csdef file in order to be seen from the outside. The following code from the ServiceDefinition.csdef file shows an example of a service, composed of Web, Worker and Virtual Role :

```
<?xml version="1.0" encoding="utf-8"?>
<ServiceDefinition name="MyServiceName" xmlns="http://schemas.microsoft.com/ServiceHosting/2008
<WebRole name="WebRole1" vmsize="Medium">
    <Sites>
        <Site name="Web">
            <Bindings>
                <Binding name="HttpIn" endpointName="HttpIn" />
            </Bindings>
        </Site>
    </Sites>
    <Endpoints>
        <InputEndpoint name="HttpIn" protocol="http" port="80" />
        <InternalEndpoint name="InternalHttpIn" protocol="http" />
    </Endpoints>
    <Certificates>
        <Certificate name="Certificate1" storeLocation="LocalMachine" storeName="My" />
    </Certificates>
    <Imports>
        <Import moduleName="Connect" />
        <Import moduleName="Diagnostics" />
        <Import moduleName="RemoteAccess" />
        <Import moduleName="RemoteForwarder" />
    </Imports>
    <LocalResources>
        <LocalStorage name="localStoreOne" sizeInMB="10" />
        <LocalStorage name="localStoreTwo" sizeInMB="10" cleanOnRoleRecycle="false" />
    </LocalResources>
    <Startup>
        <Task commandLine="Startup.cmd" executionContext="limited" taskType="simple" />
    </Startup>
</WebRole>

<WorkerRole name="WorkerRole1">
    <ConfigurationSettings>
        <Setting name="DiagnosticsConnectionString" />
    </ConfigurationSettings>
    <Imports>
        <Import moduleName="RemoteAccess" />
        <Import moduleName="RemoteForwarder" />
    </Imports>
    <Endpoints>
        <InputEndpoint name="Endpoint1" protocol="tcp" port="10000" />
        <InternalEndpoint name="Endpoint2" protocol="tcp" />
    </Endpoints>
</WorkerRole>

<VirtualMachineRole name="MachineRole" vmsize="Medium">
    <Imports>
        <Import moduleName="RemoteAccess" />
        <Import moduleName="RemoteForwarder" />
    </Imports>
</VirtualMachineRole>
</ServiceDefinition>
```

[8]
An InputEndpoint is an external connection, whereas an InternalEndpoint is an internal connection.
SIP ports are 5060 and 5061 for secured transmissions. As mentioned in 0.7.1, page 25, UDP is not supported by Windows Azure, without the Connect Service. We can assume that a SIP service would have a ServiceDefinition.csdef file that would contain :

```
<Endpoints>
    <InputEndpoint name="SIPEndpointname" protocol="tcp" port="5060(5061)" (certificate="SSL")
</Endpoints>
```

0.7.3 Microsoft Server as a SIP server ?

This part deal with the capability of configuring Microsoft Windows Server R2 2008 VHD, provided by the VM, as a SIP server. Indeed, users can specify the virtual server's behavior :

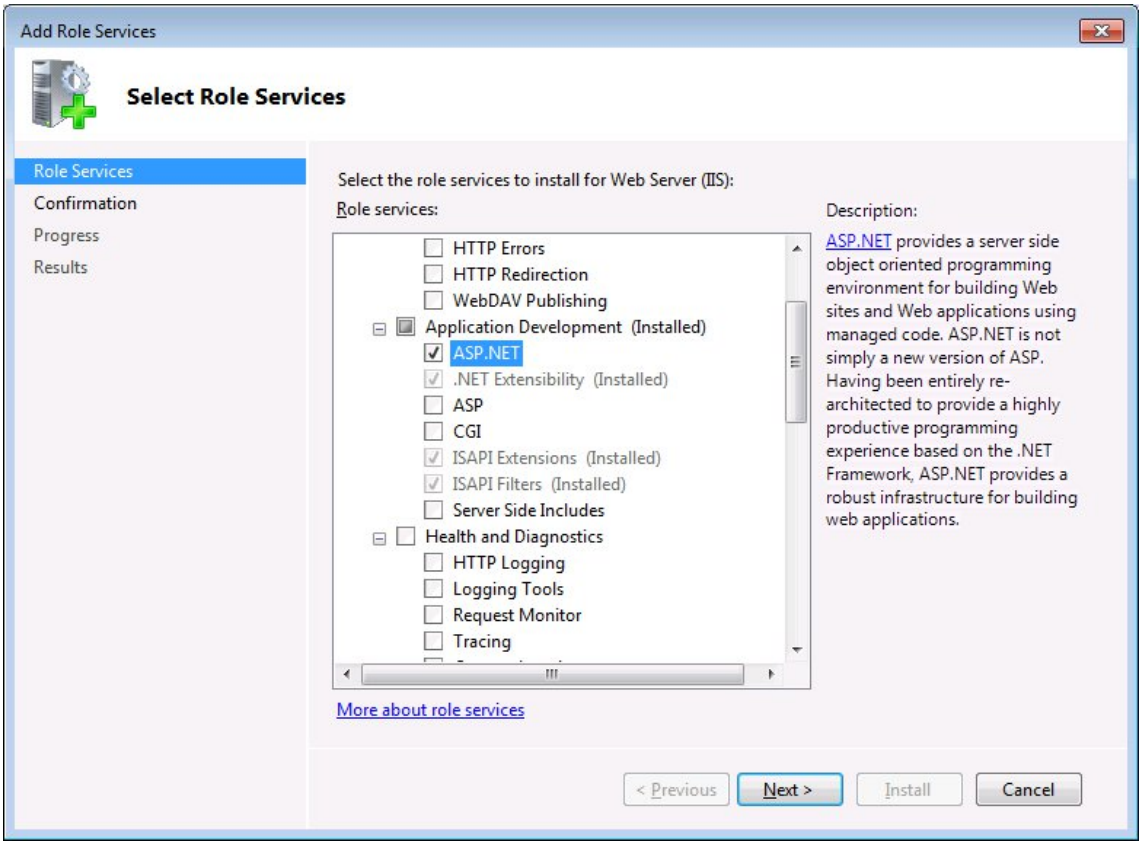


Figure 21: IIS - Server Features [6]

World Wide Web Service is often chosen, but other features can be used too, like FTP, as reported in Mayur Gondaliya's blog [15]. Maybe a SIP behavior option is available. If not, several extensions may be downloaded to set the kind of servers users want to have (from media streaming to web applications hosting), on the IIS (Internet Information Services) website: `http://www.iis.net/download`. The implementation of a SIP server on Windows Azure could resemble that of the FTP server, described by Mayur Gondaliya [15] : open SIP ports to the virtual server, enable a remote access and configure the SIP server role on the virtual server.

0.7.4 Another SIP server ?

A second implementation could enable a SIP service : instead of considering Microsoft Server as a SIP server, let's think about a non-Microsoft solution : we've seen many solutions available in part II.
Thanks to Windows Azure Connect, any local network can be connected to Windows Azure roles, without restriction for port numbers (UDP and TCP) through a secure IP connection (IPsec). Local machines can communicate bi-directionally over an IPv6 address that Connect provides. Furthermore, this connection will work through firewalls and network address translation (NAT). Windows Azure Connect enables using existing remote administration tools. Any Azure role (Web, Worker, or VM) can be connected to servers from an organization (SQL server for example), and networked resources (VMs for file, print, email, database access, Web communication, collaboration ...).

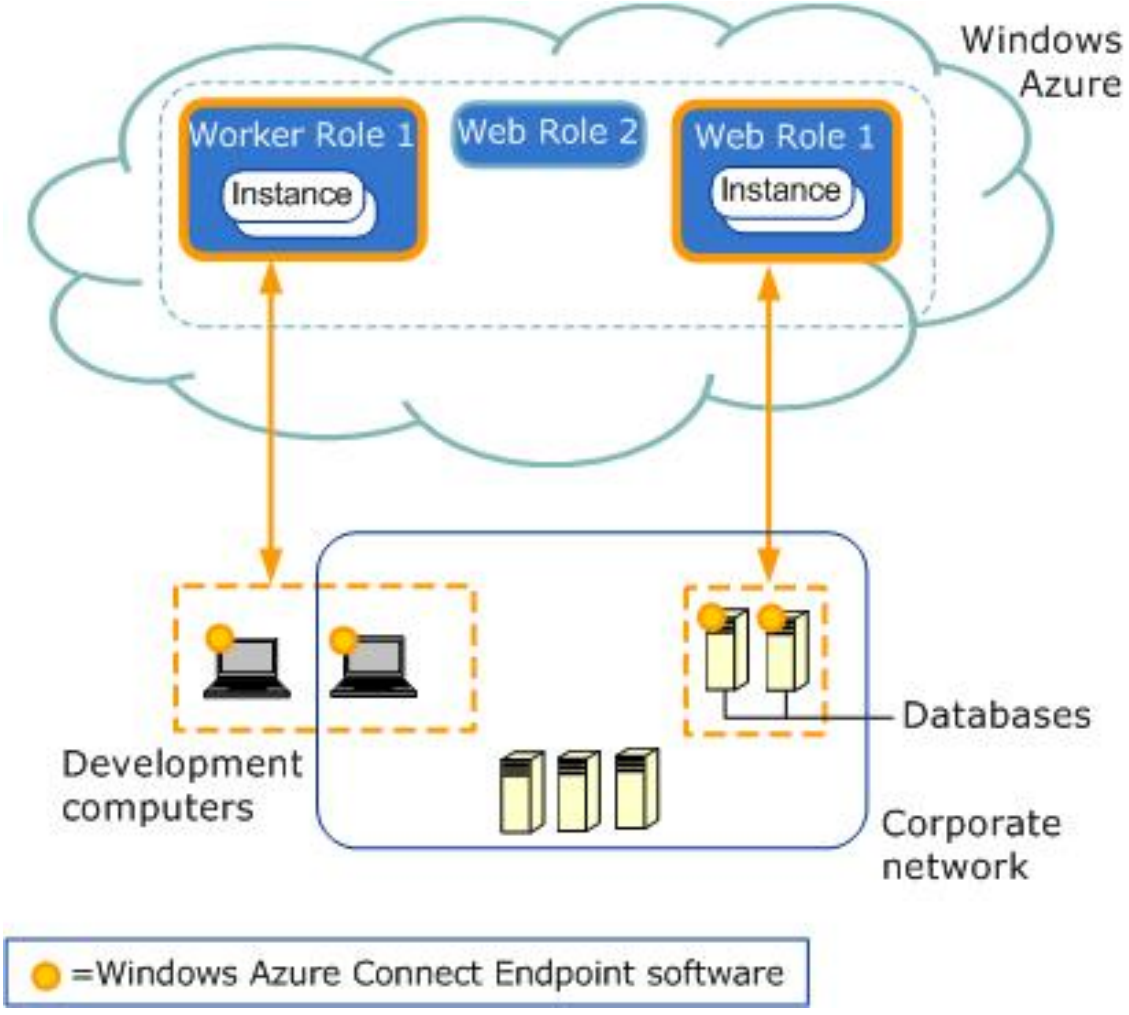


Figure 22: Example configuration in Windows Azure Connect [7]

That means a SIP service, made of any role, could establish a communication with any SIP server and its local network. Once local machines' firewalls checked, and client applications programmed and set up, they could be able to receive UDP or TCP traffic the applications are listening on. The connection installation process is explained on the MSDN website [9].
Note that the tag <Import moduleName="Connect" /> from the ServiceDefinition.csdef file (0.7.2, page 25) is automatically added at this stage. The following screenshot shows the local group 'My Server'. This group is made of only one member machine (server-2k8x64) but it's possible to add other local ressources. The role called 'SydneyCustomersWebRole' is connected to the group and able to communicate with it.

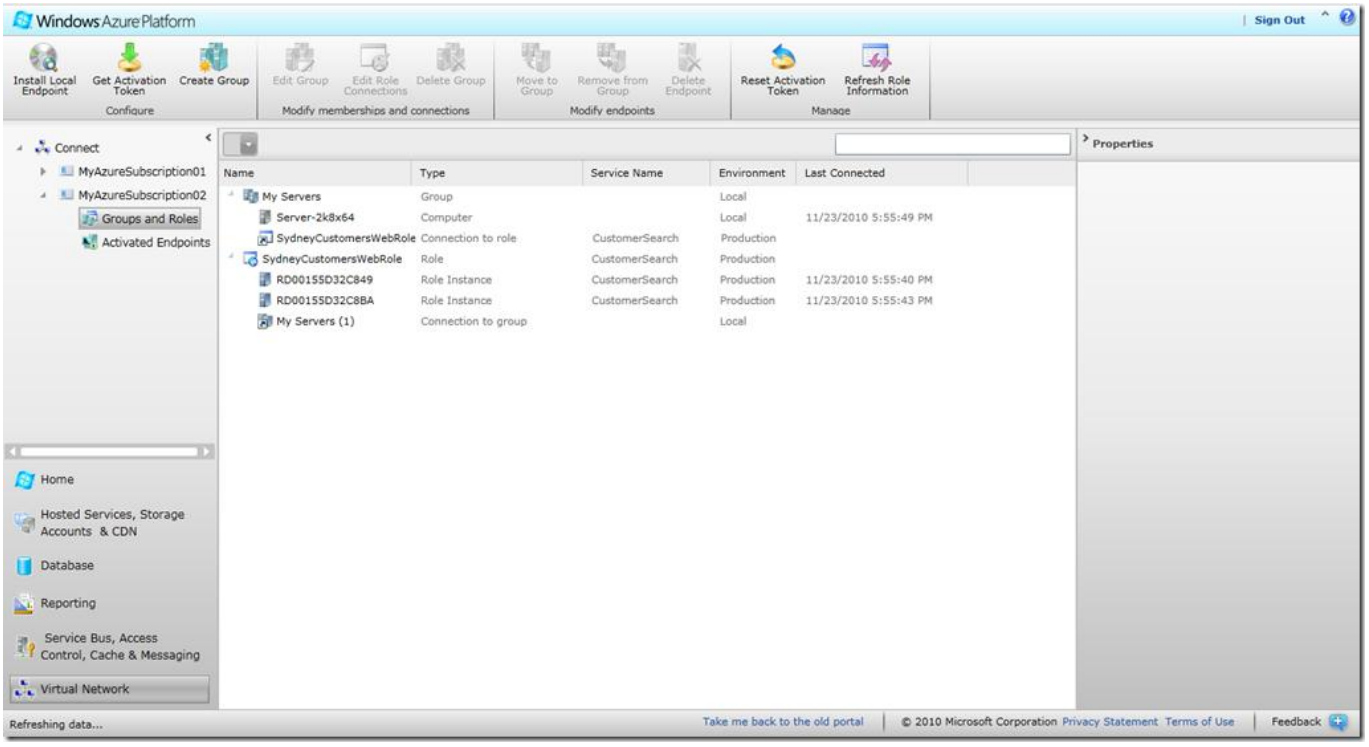


Figure 23: Local Network with Windows Azure Connect [9]

To conclude, the last solution, if turned out to be feasible, could let IT administrators choose SIP servers (regarding their own criteria : License, Price, Technical parameters...) ; the transport layer does not interfere with their choice. Moreover, they could continue to benefit from existing on-premises applications .

Bibliography

[1] Comparison of voip software - servers. http://en.wikipedia.org/wiki/Comparison_of_VoIP_software.

[2] Introduction to the vm role development process. http://msdn.microsoft.com/en-us/library/gg466229.aspx.

[3] Sip signaling. http://www.sipcenter.com/sip.nsf/html/SIP+Signaling.

[4] Understanding sip - today's hottest communications protocol comes of age. http://www.sipcenter.com.

[5] What is sip introduction, 2000-2011. http://www.sipcenter.com.

[6] Getting ready for azure development, June 2010. http://www.azurecloudpro.com/.

[7] Overview of windows azure connect, November 2010. http://msdn.microsoft.com/en-us/library/gg432997.aspx.

[8] Overview of setting up a hosted service for windows azure, April 2011. http://msdn.microsoft.com/en-us/library/hh124110.aspx#bk_Define.

[9] Tutorial: Setting up windows azure connect, May 2011. http://msdn.microsoft.com/en-us/library/gg508836.aspx#UsingTheVMRole.

[10] O. Allen and C. BTS. Windows azure platform technical overview, pricing and slas. https://skydrive.live.com.

[11] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia. Clearing the clouds away from the true potential and obstacles posed by this computing capability. *Communications of the ACM*, April 2010. VOL. 53 | n°. 4.

[12] L. Badger, T. Grance, R. Patt-Corner, and J. Voas. Draft cloud computing synopsis and recommendations, May 2011. http://csrc.nist.gov/publications/drafts/800-146/Draft-NIST-SP800-146.pdf.

[13] D. Chappell. Introducing the windows azure platform (whitepaper). *MICROSOFT CORPORATION*, December 2009.

[14] Cisco. *CCNA Exploration 4.0 - Network Fundamentals.*

[15] M. Gondaliya. Creating ftp server on windows azure, December 2010. http://mayur.gondaliya.com/microsoft/creating-ftp-server-on-windows-azure-493.html#respond.

[16] J. Heinzelreiter and W. Kurschl. *Cloud Computing Software Engineering Fundamentals.* Upper Austria University of Applied Sciences.

[17] K. Kumar. Overview of cloud computing and the windows azure platform: an academic perspective. http://azurepilot.com.

[18] R. Ltd. Sip server - technical overview. 2004. http://www.sipcenter.com.

[19] MSM. So what is the cloud? *MSM - Application Development*, March 2010. http://www.msmsoftware.com/2010/3/10/so-what-is-the-cloud.aspx.

[20] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. Sip: Session initiation protocol. *RFC 3261*, June 2002. http://www.ietf.org/rfc/rfc3261.txt.

[21] H. Schulzrinne. Listing of public sip servers. October 2008. http://www.cs.columbia.edu/sip/servers.html.

[22] Q. Zhang, L. Cheng, and R. Boutaba. Cloud computing: state-of-the-art and research challenges. *The Brazilian Computer Society 2010*, April 2010.