Realization of Interoperability & Portability Among Open Clouds by using Agent's Mobility & Intelligence

Rabia Khan[1] and Amjad Mehmood[2]
[1,2]Institute of Information Technology, KUST, Indus Highway, Off Jarma, Kohat, KPK, Pakistan
[1]rabia.pk123@gmail.com, [2]amjadiit_kust@yahoo.com

***Abstract*– Cloud Computing has become most demanding utility or service for the current era, because of its high computing power, performance, cheapness, accessibility, scalability, and availability. But still it is in infancy stage, and has some pitfalls which are due to non-existence of standards. Interoperability and portability are the two among the major issues in Cloud Computing. Authors have pointed out these issues and how actually interoperability and portability issues would be encountered? Authors propose architecture to address these two issues with the collaboration of next emerging technology i.e., agents and XMPP protocol. As there is an architecture proposed before using agents but in this paper first time both features of an agent i.e. intelligence and mobility are used in some particular way. Mobility is for movement among different clouds, as agents are interoperable by default as per FIPA (Foundation of Intelligent Physical Agent), and intelligence is to take the wise decision by keeping number of attributes in the database i.e. workload per service on each machine, distance between the clouds and services available on each cloud to fix the above cited problems.**

***Keywords*–** Interoperability, Portability, Open Clouds, FIPA Agents and XMPP

I. INTRODUCTION

Cloud computing is the integration of many IT technologies like grid computing, cluster computing, utility computing, web 2.0, SOA and much more thus precise definition for cloud computing is harder to state. Cloud computing can be thought of as an infrastructure that provides storage, processing and applications as service. These services can be accessed over the internet by using some standard browser. Cloud computing is three layered service architecture [5], [8]:

SaaS: Applications delivered as service on top SaaS layer consumed directly by the user. SaaS provides fully functional software over the internet to user instead of letting him install software on his computer.

PaaS: Middleware provides runtime environment for application development to be run on infrastructure provided by the provider.

IaaS: Distributed resources connected via internet provide infrastructure for cloud services e.g. CPU, database, storage etc.

Cloud computing provides many advantages [5] listed as:

i). Customer demands services according to his/her needs and they are provisioned as per need
ii). Resources are added or removed as per demands
iii). Pay-as-you-go is the most important feature of cloud computing, user has to pay only for the services he uses
iv). Cloud computing completely is based on self service concept. Customer or service provider is responsible for the services he uses or provides, no administrator is available to configure the resources or provision/deprovision the resources.
v). Cloud is a collection of infinite resources; user can acquire resources according to need and then release them back to the pool after use.
vi). User need not to invest on any infrastructure, does not need to purchase any hardware, this reduces investment cost in hardware etc and many more as one can think of.

There are some concerns about cloud computing- as discussed by many researchers-which has to be addressed in order to make cloud computing concept spread world wide successfully. Concerns include [1], [5], [8]:

a). A service provided by one cloud may not follow the same rules on another cloud which locks the service in a single cloud and user has to follow different rules and regulations if he wants to use the same service of another cloud by abandoning previous cloud service thus interoperability among clouds is required which is yet to be established.
b). Huge amount of data upload on cloud may be difficult due to heavy use and shared nature of cloud.
c): Many CCSP (Cloud Computing Service Providers) promise to provide infinite scalability for customer but due to the fact that millions of users are now migrating to cloud computing so to cope up with user demands such promises are not fulfilled.
d). Since many systems have been crashed on cloud like Amazon so using only one CCSP services can result in a drawback as when a shutdown event happens on a cloud, the service disappears and user cannot find that service.
e). Because of absence of portability feature in cloud it is impossible for a user to move his application from one cloud to another; as a result user is locked in to a certain CCSP.
f). Applications can't scale from one CCSP to another due to lack of interoperability standards.

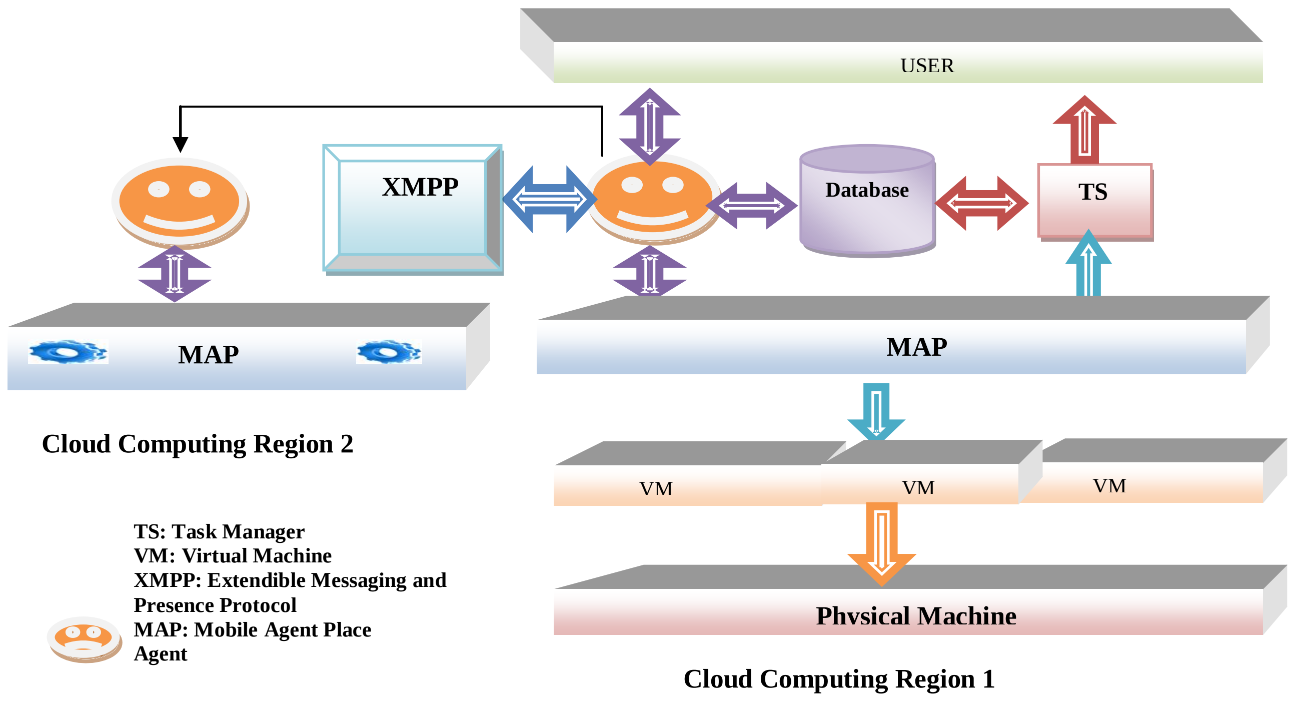

Fig. 1: Architecture of how realization of interoperability and portability will be accomplished

g). As it's the advantage of cloud computing user is free from investing in large hardware and expensive software, he just needs to put request and similar services and facilities will be provided to him through cloud. Same goes to service providers; he has to keep an eye on heavy demands by users but it all comes by cost to provider.

In this paper authors focused on interoperability and portability issues by using XMPP protocol: for discovering the nearby clouds, and providing same information to the agents: which is next state-of-art technology. It has features like mobility and intelligence: mobility is for interoperability and portability among the open clouds, intelligence for taking wise decision by calculating the work load on each MAP's (mobile agent place) virtual machine(s),services available on each cloud , estimating the distance between the clouds: by counting no of hops, and place the result in database. All those computations would not be done at runtime, in order to increase the performance of the cloud, however those parameters manipulation would be preprocessed after week or month and results would be stored in database. So after storage, when agent would be requested for the service, cloud would be more intelligent than before and would search its database, find the appropriate result and guide the operation of interoperability and portability accordingly as per parameters' conditions specified in paper.

Rest of the paper is organized as follows: Section II describes the relevant work done to address portability and interoperability among clouds. Section III shows the architecture of this paper. Section IV shows the mathematical analysis of this work. Section V is a case study and Section VI is conclusion.

II. RELEVANT WORK

Many researchers address portability and interoperability and proposed solutions. Zehua Zhang and Xuejie Zhang proposed MABOCCF (Mobile Agent Based Open Cloud Computing Federation) [1]. Their proposed architecture works as follows: User task is encapsulated in a mobile agent which runs on MAP (Mobile Agent Place). Mobile agent keeps an eye on resources of MAP and decides when to leave it to migrate to another MAP either on same CCSP or another if resources on particular MAP become scarce.

We take similar approach like MABOCCF but there are few differences:

1) MABOCCF was quite abstract and we propose all minute details.
2) MABOCCF uses only agent's mobility while we are using mobility plus intelligence
3) We achieve realization by using XMPP
4) MABOXFF proposed that one VM may have multiple MAPs while we propose that multiple VMs having same service providing features are joint together to work under one MAP that is named by the service it provides e.g., HTTP MAP etc. Here MAP serves the same purpose as hypervisor.

A V. Parameswaran and Asheesh Chaddha in their article Cloud Interoperability and Standardization [2] proposed.

Unified cloud interface/Cloud broker and Enterprise Cloud Orchestration Platform /Orchestration layer.

David Bernstein et al. in [3] proposed protocols based interoperability among clouds.

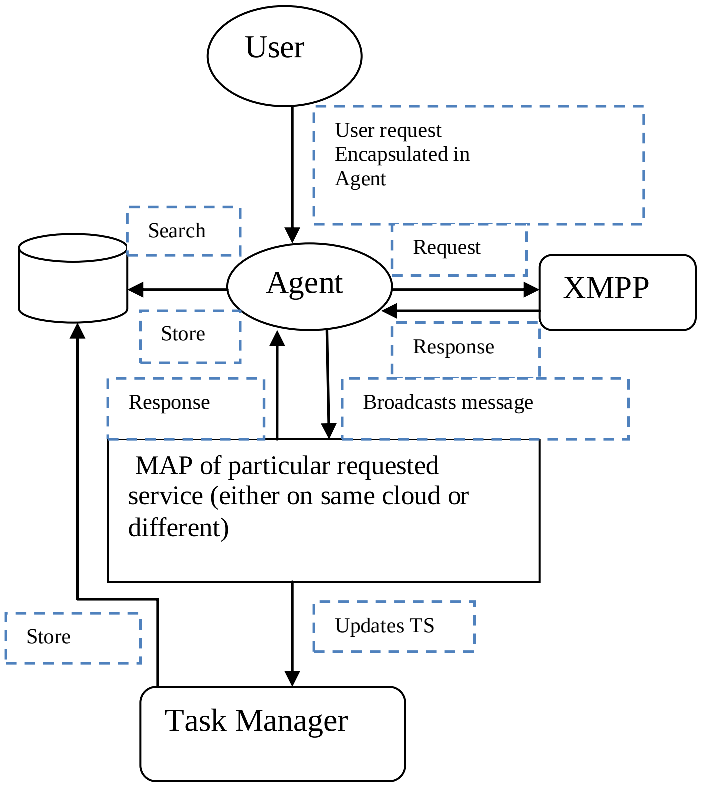
User
User request Encapsulated in Agent
Search
Request
Agent
XMPP
Store
Response
Response
Broadcasts message
MAP of particular requested service (either on same cloud or different)
Updates TS
Store
Task Manager

Fig. 2: Dataflow Diagram

Agent Table: 2

Cloud	Service	# of HOPs	MAP	% load	# of VMs
B	HTTP	30	HTTP-MAP	31.08%	03
C	HTTP	20	HTTP-MAP	90%	02

XMPP Table: 1

Service	Cloud
HTTP	B
HTTP	C

Cloud	MAP(specified Service)	VMs (in order)
B	HTTP-MAP	1,3,6
C	HTTP-MAP	2,5

III. ARCHITECTURE

We will take similar approach like MABOCCF [1]. User task is encapsulated in mobile and intelligent agents. Agents get registered to MAP (Mobile Agent Place) and runs there. Multiple Virtual Machines providing similar services are grouped together and a unified MAP is installed over them. Every MAP is named by the services underlying VMs provide e.g. HTTP MAP, SMTP MAP etc. Agent interacts with appropriate MAP keeping in view user requirements. Agent monitors resource condition on MAP on which it is running and when it finds resource scarcity it migrates to another MAP on another cloud having similar functionality ensuring portability. The architecture has some special components to integrate interoperability among different clouds. If an agent does not find any suitable MAP for its requirements, it has to move to another cloud. XMPP (Extensible Messaging and Presence Protocol) is a protocol used here which helps in discovery of clouds and services offered by clouds.

XMPP is a set of open XML technologies for presence and real time communication. XMPP interacts with agent to provide information about nearby clouds and their respective services. Agent saves those updates in database for future references. TS (Task Manager) get information about nearby clouds from database and accordingly update itself about the resources condition in nearby clouds. Each TS has more than one MAP and those MAPs are responsible for timely informing TS about the resources residing in them and the percentage of MAP busy for particular service thus TS is used for resource indexing. MAP also updates TS about agents who are registered there and agents who left MAP. TS is also helpful in authentication, security, billing, disaster management and fault tolerance. The architecture of this scenario is shown in Fig 1.

A. Working Mechanism

Agent requests XMPP server to investigate nearby clouds and their respective services. XMPP contacts other XMPP servers residing on nearby clouds and gets update and returns that information to the agent. Agent saves the information in database for the TS to update its status and for future references. Agent provokes XMPP time to time to get recent updates about nearby clouds. Upon receiving information from XMPP, agent would broadcast a message to nearby cloud having required services. Agents residing on MAP of different cloud would entertain the query by sending required information back.

Required information will include number of VMs that are working together to perform required service, percentage of service load on cloud and the appropriate MAP who can facilitate the requested service. Database entries are shown in Table 1, Table 2 and Table 3. When a user task has been arrived, two cases may result:

Case 1: Agent would search the database to see which cloud could better satisfy user request and a match is found.

Case 2: Agent search for appropriate match in database but could not find a match then it has to activate XMPP for updates to be stored in database.

Agent selects appropriate cloud matching user requirements and moves there. If more than one cloud has the required services then agent has to select the one having less workload of required service.

In a situation when two clouds with same workload of required service, the agent would select the cloud on the basis of minimum number of hops it has to travel and hop information is also stored in database.

Results are returned to the user either directly by agent or through TS. This mechanism is shown in Fig. 3.

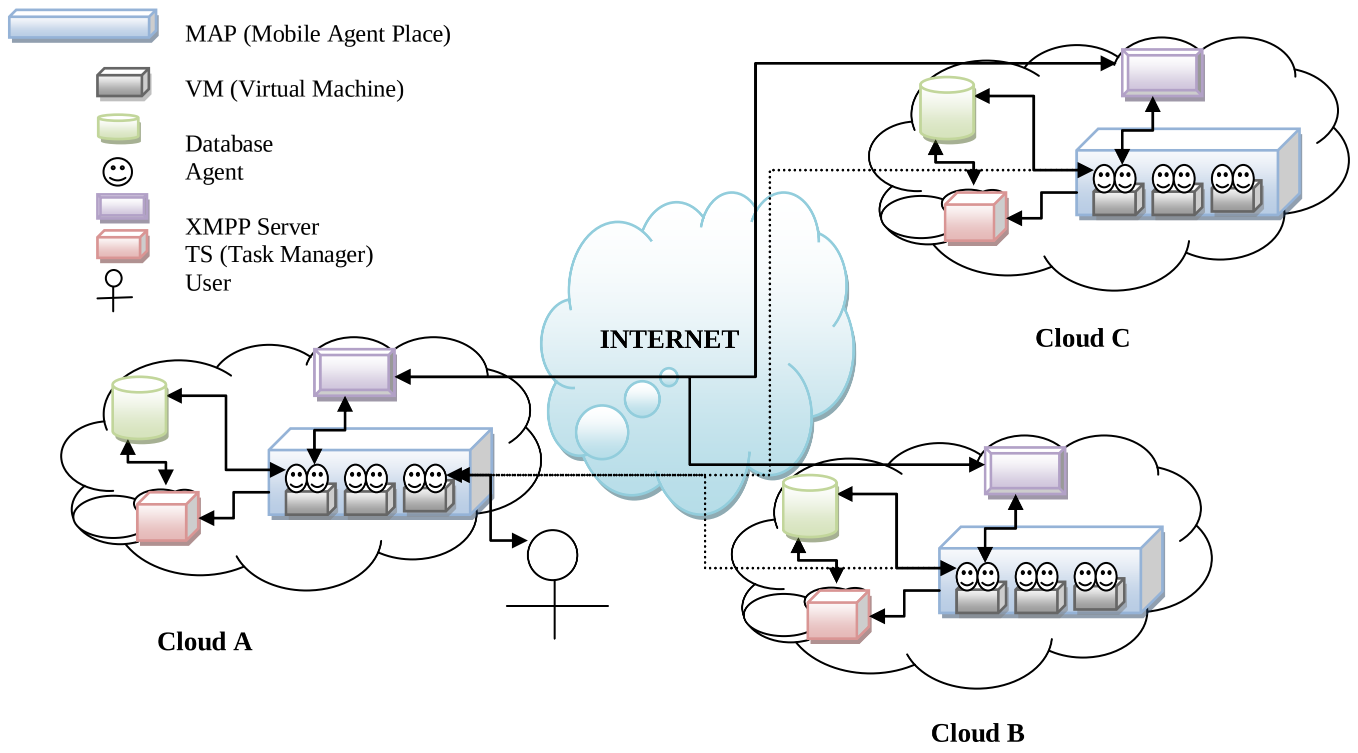

Fig. 3: Showing the scenario interoperability and portability by using agent's mobility and intelligence

Data flow Diagram of working mechanism is shown in Fig. 2. Four cases have been considered while constructing this architecture which are as follows:

Case I: Task delegation on single MAP among 2 agents

Case II: Migration and delegation of task among 2 agents from one MAP to another on same cloud.

Case III: Migration and delegation of task among 2 agents from one MAP to another on different clouds.

Case IV: Assignment of same task to more than 1 agent

Case I, II and III could be handled using technique given in [4] which shows two approaches for handling such issues.

One Phase Approach: An agent delegates its task to other agent; task is then decomposed into fragments (subtasks) which are then further delegated to other agents until the task is completed.

Two Phase Approach: Agents are first asked if they are capable of executing the task. If so then task is divided into fragments (subtasks) and handed over to agents who have approved task execution. Those agents will in turn request other agents for execution of subtasks and if request accepted, subtasks are handed over to them. This process continues until task is completed.

Case IV will arise race condition among agents. Agents will compete each other for resources needed to complete the task successfully.

IV. MATHEMATICAL ANALYSIS

For the purpose of this paper, we need to find out the percentage of services every MAP is to compute and provide this information to TS which then saves it in database.

Total CPU Time = Time CPU busy % * Task CPU burst [1] * number of processors ----------------------- (1)

([1] CPU requirement by a particular task as calculated by operating system)

% Time CPU busy can be calculated by the formula:
% Time CPU wait = [(Online Time – Wait Time /Online Time] * 100 ---------------------------------- (2)

Now to calculate the time captured by a particular task:
Captured CPU Time =Total CPU Time – Time CPU Busy --- (3)

Now to calculate service % workload on a particular MAP:
Service CPU % = (Captured CPU Time / task CPU burst) * 100 ……………………….………………… (4)

V. CASE STUDY

Let suppose we have two similar requests for HTTP service. Suppose two clouds B and C provide HTTP service. Which cloud must an agent select to forward service request?

Agent would first check database to dig out the required matching criteria. Criteria would be to select a cloud having less service CPU percentage that means that cloud would be having fewer loads for that particular service so a user request could be fulfilled early if sent to that cloud. So suppose Cloud B has single processor and CPU burst for the task is 15 seconds, using above equations to find out service percentage on cloud B.

Using equation (2):
Let suppose system is online from 15 seconds and waiting for task for 10 seconds, then % CPU busy becomes:
% CPU wait = [(15 – 10)/15] * 100
% CPU wait = 33.3%
Putting values in equations (1), (3) and (4)
Total CPU Time = 33.3% * 15 * 1
Total CPU time = 4.995 seconds
Captured CPU time = 4.995 – 0.333
Captured CPU time = 4.662 seconds

Service CPU % = (4.662 * 100) / 15
HTTP CPU % on cloud B= 31.08%
Now coming to cloud C. Let suppose it has a dual core system.
Let suppose the system is online from 15 seconds and waiting for task for 8 seconds.

Using equation (2):
CPU wait % = [(15 – 8) / 15] * 100
CPU wait % = 46.6667%
Putting values in equation (1), (3) and (4)
Total CPU time = 46.6667 % * 15 * 2
Total CPU time = 14.0 seconds
Captured CPU time = 14.0 – 0.466667
Captured CPU time = 13.5 seconds
HTTP CPU % = (13.5 * 100) / 15
HTTP CPU % on cloud C= 90%

Results show that HTTP service on cloud C is heavily loaded. So if an HTTP request is sent to cloud C, it would take too much time to be serviced, that's why better option is cloud B where HTTP workload less then cloud C and it will service the request faster.

Option is cloud B where HTTP workload less then cloud C and it will service the request faster.

VI. CONCLUSION

Cloud computing is yet in its infancy stage and no such standard exists which could set rules or privileges for cloud computing. Researchers, academia and organizations need to work collectively and set open standards acceptable to all. This effort is a step towards this milestone. Interoperability and portability could be achieved only if we adopt open mechanisms like discussed in this paper.

REFERENCES

[1]. Zehua Zhang and Xuejie Zhang"Realization of Open Cloud Computing Federation Based on Mobile Agent", IEEE International Conference on Intelligent Computing and Intelligent Systems, Publisher: IEEE, 2009, pp: 642- 646

[2]. A. V. Parameswaran and Asheesh Chaddha "Cloud Interoperability and Standardization" SET Labs Briefings VOL 7 NO 7, 2009.

[3]. David Bernstein, Erik Ludvigson, Krisna Sankar, Steve Diamond and Monique Morrow "Blueprint for the Intercloud Protocols and Formats for Cloud Computing Interoperability", The Fourth International Conference on Internet and Web Applications and Services, ICIW 2009, IEEE Computer Society. 2009, pp. 328–336

[4]. António Luís Lopes, Luís Miguel Botelho "Task Decomposition Coordinating Unstructured Multi Agent Systems", published in Proceedings of the International Conference on Complex, Intelligent and Software Intensive Systems, 2007, pp. 209- 214,.

[5]. Karl Scott "The Basics of Cloud Computing" White Paper November 2010, akaili systems inc.

[6]. Shufen Zhang, Shuai Zhang, Xuebin Chen and Shangzhuo Wu "Analysis and Research of Cloud Computing System Instance" 2010 Second International Conference on Future Networks, China, 2010, pp 88-92

[7]. Shuai Zhang, Shufen Zhang, Xuebin Chen and Xiuzhen Huo "Cloud Computing Research and Development Trend" 2010 Second International Conference on Future Networks, China, 2010, pp 93-97

[8]. H. Gilbert Miller and John Veiga "Cloud Computing: Will Commodity Services Benefit Users Log Term? " P u b l i s h e d by t h e IEEE Comp u t e r S o c i e t y, 2009 (vol. 11 no. 6), pp. 57-59.

[9]. Rich Maggiani "Cloud Computing Is Changing How We Communicate", 2009 IEEE International Professional Communication Conference USA, 2009, pp 1-4

[10]. G. Cabri, L. Leonardi, F. Zambonelli "MOBILE AGENT TECHNOLOGY: CURRENT TRENDS AND PERSPECTIVES"http://www.agentgroup.unimo.it/MOON/papers/pdf/aic,http://sirio.dsi.unimo.it/MOON/papers/aica98.ps.gz.

[11]. Sergiy Nikitin, Vagan Terziyan and Michal Nagy "MASTERING INTELLIGENT CLOUDS Engineering Intelligent Data Processing Services in the Cloud", ICINCO 2010, Proceedings of the 7th International Conference on Informatics in Control, Automation and Robotics, Volume 1, Portugal, 2010, pp 174-181.

[12]. "Open Cloud Manifesto," http://www.opencloudmanifesto.org/Openn%20Cloud%20Manifesto. pdf.

[13]. "Above the Clouds: A Berkeley View of Cloud Computing," Http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html.

[14]. "Interoperable Clouds", A White Paper from the Open Cloud Standards Incubator, Version 1.0.0, Status: DMTF Informational, Publication Date: 2009-11-11, Document Number: DSP-ISO101

[15]. Liang-Jie Zhang and Qun Zhou "CCOA: Cloud Computing Open Architecture", 2009 IEEE International Conference on Web Services, 2009, pp 607 – 616

[16]. Craig A. Lee Open Grid Forum and the Aerospace Corporation "A Perspective on Scientific Cloud Computing", HPDC'10 proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, 2010, pp 451-459

[17]. www.fipa.org visited on July 12,2011

[18]. www.xmpp.org visited on July 22, 2011